Detecting Illicit Data Leaks on Android Smartphones Using an Artificial Intelligence Models

Serge Lionel Nikiema[1(✉)], Aminata Sabane[2], Abdoul-Kader Kabore[3], Rodrique Kafando[1,2,3], and Tégawendé F. Bissyande[1,2,3]

[1] Centre d'Excellence en IA pour le Developpement (CITADEL), Ouagadougou, Burkina Faso
lionelsergepha@gmail.com
[2] Université Virtuelle du Burkina Faso, Ouagadougou, Burkina Faso
[3] Université Joseph Ki-Zerbo (UJKZ), Ouagadougou, Burkina Faso

Abstract. In today's digital landscape, hackers and espionage agents are increasingly targeting Android, the world's most prevalent mobile operating system. We introduce DeepDetector - a system based on artificial intelligence to recognize data thefts in Android. This model is based upon a large dataset comprising of clean and tainted network traffic trained using a Random Forest Classifier. DeepDetector scores high in two main areas as it achieves 82.9% accuracy for connection anomaly detection and 89.9% recall in connection anomaly detection whereas it gets 78.9% accuracy and 81.6 recall in terms of detection of under the system mounted with Raspberry Pi, automatic data collection, preparing of a dataset, training and testing of the model, as well as leak detection are ensured. In this regard, DeepDetector offers a viable way of enhancing Android user security.

Keywords: Android · mobile security · data leak · machine learning · anomaly detection

1 Introduction

Currently, Android smartphones hold 72% of the global mobile operating system market share [1]. At the same time, this high degree of adoption renders Android as a potential prey for malware seeking to steal user's information. Many solutions have been developed in the field of malware detection. But detecting data leaks remains a major challenge. Because it's hard to distinguish between information leaving smartphones that is important for the proper functioning of applications, and information leaving stolen by malicious users or malicious organization. This study proposes a solution to the problem of data leakage. There are three main detection methods: these are static, dynamic and hybrid analyses [2].

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 186–200, 2024.
https://doi.org/10.1007/978-3-031-63215-0_14

Yet, sophisticated static detection approaches also fail against modern malware. Dynamic techniques based on machine learning are however the most advanced approaches being considered as an emerging alternative.

The following describes DeepDetector that is a system for detecting illegal information leakage on Android phones. Instead, the proposed solution involves a machine learning algorithm that will be able to analyze in real time network data and detect any abnormalities suggesting unauthorized leakage of personal or confidential data. DeepDetector is a fully autonomous artificial intelligence system for detecting data leaks. This system incorporates automated processes. The process of collecting network traffic, extraction of pertinent features and their processing using the pre-trained ML model and also the visualization of results occur automatically and consecutively within the system. DeepDetector is a step forward in the detection of data theft.

Our work will be organized according to following arrangement. Then, describing the overall and specific structure of the system will be included in the next section. To begin with, the design of the dataset and training of this model with its rationalization of why the picked algorithm is appropriate will also be stated. Thirdly, a comparative study will be carried out and resultant findings highlighted. Thereafter, a conclusion with relevant recommendations/perspective shall be made.

2 State of the Art

The use of artificial intelligence (AI) and machine learning offers new perspectives for improving the detection and prevention of data leakage. This state of the art reviews the main AI approaches applied to data theft detection.

2.1 Detection Based on Access Analysis

One of the most widespread approaches is to monitor and analyze access to sensitive files and databases. AI techniques can be used to:

Detection of Abnormal Behavior

- Use of anomaly detection models based on unsupervised learning (clustering, isolation forest, etc.) to identify atypical accesses [2].
- Implementation of intelligent intrusion detection systems (IDS) combining behavioral analysis and machine learning techniques [3].

Prediction of Data Theft Risks

- Development of prediction models based on supervised learning (regression, random forests, neural networks, etc.) to identify at-risk user profiles [4].
- Use of deep learning techniques to improve prediction accuracy [5,6].

2.2 Detection Based on Network Traffic Analysis

Analysis of incoming and outgoing network traffic is also a promising approach to detecting data theft. AI techniques enable:

Identification of Suspicious Communications

- Use of anomaly detection models in network traffic (neural networks, support vector machines, etc.) to identify unusual activity [7].
- Development of intrusion detection systems based on deep learning capable of detecting data leaks [8,9].

Predictive Traffic Analysis

- Implementation of network traffic prediction models using machine learning (recurrent neural networks, Markov models, etc.) to anticipate abnormal behavior [10,11].
- Combining traffic analysis with other data sources (access logs, metadata, etc.) to improve detection [12].

2.3 Detection Based on Metadata Analysis

Exploiting file metadata (size, location, timestamps, etc.) Also represents an interesting avenue for detecting data theft. AI techniques enable:

Unusual Copy Detection

- Use of anomaly detection models based on unsupervised learning to identify atypical file copies [13,14].
- Development of systems to track changes and movements of sensitive files using machine learning techniques [5,15].

Data Theft Risk Prediction

- Design of predictive models using supervised learning to estimate the probability of a file being stolen based on its metadata [16].
- Integration of metadata analysis with other data sources for more accurate detection [12].

2.4 Challenge

Although AI-based detection systems offer many advantages, they face several challenges:

- Constant evolution of attack techniques requiring continuous adaptation of models [17,18]

- Management of false positives and the large volume of data to be analyzed [19]
- Interpretability and explicability of AI system decisions for decision-making [20]
- Integration with other security tools for global protection [12]

Future advances will focus on improving the adaptability, reliability and explainability of AI-based detection systems.

3 DeepDetector Description

Deepdetector is a project developed to solve the problem of data theft. The choice of the name deepdetector is based on the modular architecture of its operating system. In fact, the deepdetector project can be adapted to a specific type of detection or evolve according to training data or the performance of machine learning algorithms. In addition, it processes traffic by protocol type (TCP, UDP, DNS, etc.).

3.1 Hardware Architecture

The DeepDetector system was implemented on a nano computer called Raspberry PI 3 Model B with a powerful internal hardware configuration that includes 1.4 GHz 64 bit quad core ARM Cortex-A53 processor, 1 GB RAM, 802. Raspian is the operating system that powers this device. It has a 10.1-in. touch screen attached for interacting with the users.

3.2 DeepDetector General Software Architecture

The approach used by DeepDetector follows a 5-step main architecture as shown in Fig. 1:

1. Network traffic collection: Tshark detects Android's network traffic in real-time. The data sets include TCP connection logs and DNS queries. This data is in the native/.pcap format that cannot be used directly in its current form).
2. Data pre-processing: Unprocessed data is converted into the Zeek log format and then undergoes change to get the training dataset. Zeek program organizes data, which is contained in more than two logs that are specifically optimized for a better dataset preparation, such as conn.log and dns.log. Thereafter, they organize and prepare the connection and DNS query datasets then clean them, select features, vectorize and normalize.
3. Machine Learning model training and evaluation: Random Forest Classifier model is trained supervisely on developed data set.
4. Anomaly detection: In production, trained model analyzes Android network traffic for possible data leaks anomalies.
5. Visualization of results: in this step, you will observe the detection outcomes and draw conclusions from them.

3.3 Dataset

The aim of this work is to highlight the theft of smartphone user data. To achieve this, we need data from malicious traffic. Many datasets were found in our research, but the most relevant to us was the "Android Mischief Dataset". Why? A RAT (Remote Access Tronjan) virus is a malicious software designed to infect a target's device, thereby taking control or stealing confidential data. We assumed that the main purpose of a RAT is to steal data. However, Stratosphere University in Prague intentionally infected smartphones with 8 different RATs in order to harvest outgoing and incoming traffic. Our research has shown that this is the dataset best suited to our context. This is why we chose this dataset for our study.

DeepDetector's training dataset consists of network captures from:

- Healthy Android devices: Smartphone usage leads to normal traffic as well. To derive this "healthy" dataset, we had to go by exclusion involving using, emulating or virtualing a smartphone. Therefore, the most suitable method was the virtualization that offered minimal probability of a corrupted data.
- Devices infected with malware: traffic from one out of eight common Android RATs. A particular type of malware known as RAT (remote access Trojan) allows for the attacker to assume command and control the victim's device from afar. This takeover is primarily done for stealing of data. The Stratosphere Laboratory of the Czech Technical University, in Prague, created the Android Mischierf Dataset [12].
  The Android Mischief is a collection that comprises of network captures from different malware infected android apps. The current version of the dataset includes 8 packet captures from 8 Android RATs executed:
  - RAT01 - ANDROID TESTER V.6.4.6
  - RAT02 - DROIDJACK V4.4
  - RAT03 - HAWKSHAW
  - RAT04 - SPYMAX V2.0
  - RAT05 - ANDRORAT
  - RAT06 - SAEFKO ATTACK SYSTEMS V4.9

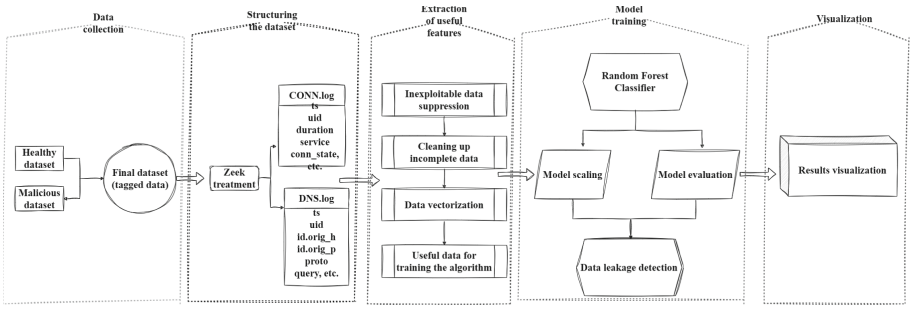


Fig. 1. System architecture.

- RAT07 - AHMYTH
- RAT08 - ANDRORAT COMMAND LINE

The dataset contains both positive (malicious traffic) and negative (normal traffic) samples as shown in Fig. 2. In total, over 500,000 samples were collected.

Also in the literature and in our research we didn't find any works exploiting the Android Mischief Dataset. So for the moment we can't objectively compare our work with others.

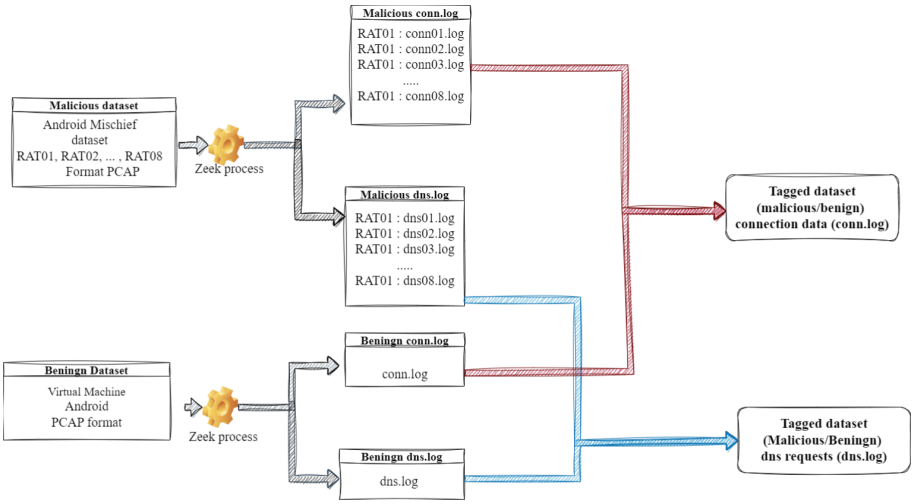


Fig. 2. Dataset Creation

3.4 Model Training

In this study, the data was trained on a single machine learning algorithm: Random Forest Classifier. First, it was chosen based on the following criteria:

- High performance: Random Forest usually has very high accuracy in malware detection and it has been reported that in a number of studies, the rate goes beyond 90%.
- Robustness to overfitting: Aggregation of multiple decision trees to form the Random Forest algorithm as well as shuffling the data during tree construction make Random Forests especially resistant to the overfitting issue. This helps in avoiding over-fitting with respect to the input information.
- Handling of imbalanced data: Malware is normally represented by the minority class in malware classification, while benign software constitutes the majority class. This is a class imbalance that is however handled by the Random Forests.

– Interpretability: In a sense Random Forest has some measure of interpretation through the tree rules as compared to pure neural network approaches. It enables one to determine the vital elements.
– Efficiency on large datasets: This makes Random Forest, an ideal tool for handling vast volumes of data collected from several Android apps, some carrying tens of thousands of attributes.

Afterwards, a general review showed it [21] suggested that random forest was best performing compared to the other methods depicted in Fig. 3.

The main objective was to set up a leak trace detection system that would operate in real time, be completely autonomous and, above all, be usable even by computer laymen. For this reason, greater emphasis was placed on automation. This is why we have selected a single machine-learning algorithm to achieve the first objectives. Looking ahead to future work, we intend to test new algorithms and build our own more precise dataset to improve system performance.

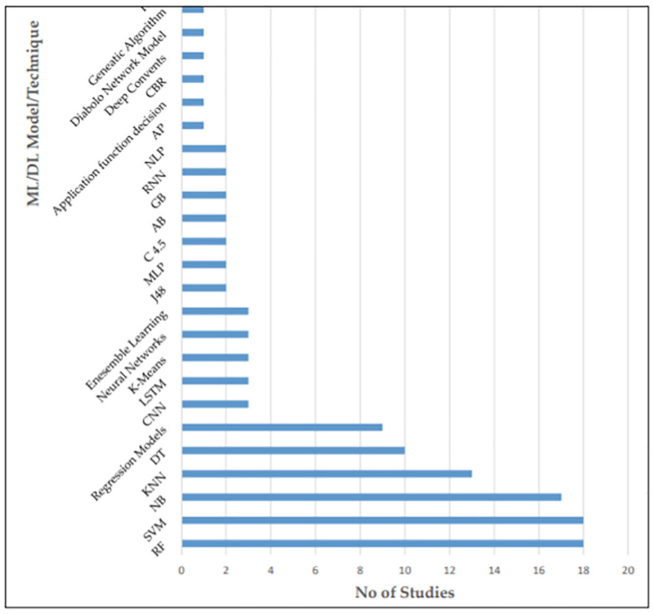


Fig. 3. Performance comparison of different algorithms.

The training set of DeepDetector's Random Forest Classifier consists of 80 % of the available dataset. This stage involves optimization of different parameters like number of decision trees. For this purpose, the rest of 20% datasets are kept as holdout for testing the performance of the model on unseen datasets. The Random Forest Classifier model was trained by optimizing the following hyperparameters via cross validation on the training dataset:

- Number of decision trees: Optimal number of trees to be tested between 10 and 500 is 100.
- Node split function: A comparison between Gini and Entropy has indicated that Gini performs better.
- Maximum tree depth: optimally with no specified depth limit, tested at least five and not more than twenty levels.
- Minimum number of samples per node: optimized between 1–10; it is best at 2
- Evaluation criterion: accuracy vs. Optimization of F1 score, better performance.

Table 1. Model evaluation results.

	Recall	Precision	F-score
Connection data	89.98%	82.90%	86.29%
DNS irregularities	81.69%	78.91%	80.27%

Zeek software separates recovered traffic into conn.log (connection logs) and dns.log (dns logs) as shown in Fig. 2. So the algorithm has been trained on both types of log separately.

Some key features required for effective detection of data leaks.

In the detection of data leaks using machine learning algorithms, it is possible to leverage a number of relevant network traffic parameters for anomaly identification and leak pattern recognition.

For our random forest model, we utilized some of these features collected from Zeek network logs in training our DeepDetector. As such, the model is designed to identify atypical combinations of these attributes in order to expose potential data compromise. Some of these features are:

- Traffic statistics per connection: Any volume of bytes sent or received that appears abnormal is suspect.
- Connection duration: Any unusual duration of these connections is deemed suspicious.
- DNS query sizes: Unusually large lengths of DNS queries are deemed uncommon and regarded as suspect ones.
- Source and destination IP addresses and ports: Other types of suspicious connections include normally closed ports.
- Timestamp of events: Timestamp consistency is important for leak detection.
- Geolocation: Such connections might be abnormal since they are not coming from their usual geographical origins.

The categorical information in our dataset was converted into numerical data so that it could be understood by the algorithm.

3.5 Source Code

The complete DeepDetector source code is available on GitHub at: https://github.com/Beninwende/Dataleak_detector_by_Machine_Learning. All Python scripts that collect data, pre-process it, train ML models, and detect anomalies are located in this directory, shown in Fig. 4. Additionally, it encompasses Jupyter notebooks utilized for first prototyping and descriptive data analytics. These notebooks describe in detail each of the key development steps: data preprocessing, training, tuning, testing and validation. The commented code is well structured. Lastly, the steps of running this system on Raspberry Pi are explained.

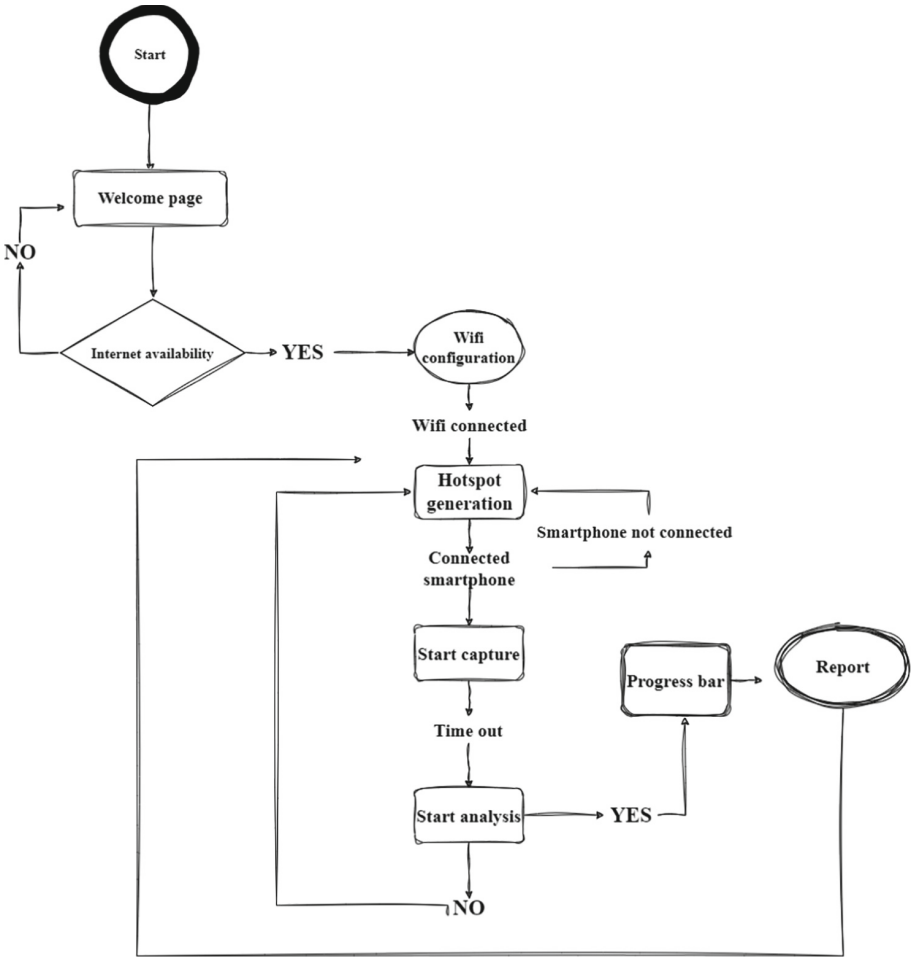


Fig. 4. System operation.

4 Comparative Study

This section makes a comparative study of different android malware detection methods using the dynamic approach based on artificial intelligence for the period 2017–2023. Several criteria are compared: use of the applied detection process, choice of the appropriate algorithm, model accuracy, and the advantages/disadvantages of each study. Data sets include DREBIN, Andro360, AMD, ML-Android and Android Mischief with up to 500,000 samples. Detectability depends on the methods used, with detection rates ranging from 83% to 99%. The performance of these methods is quite high. Nevertheless, each method has its drawbacks. See Table 2.

Table 2. Comparative study of Android malware detection techniques.

Detection approach	Selected ML Algorithm	Model Accuracy	Strengths	Limitations
Extracting DNS, HTTP, TCP, Origin based network features used by apps [2]	RF	98%	Works with different OS versions, Detects unknown malware, infected apps	Difficult to detect properly if encrypted
Using Dynamic permission analysis [22]	Simple Logistic	99.70%	High Accuracy	Problem of application blocking
Dynamically tracking execution behaviors of applications [23]	RF	96.70%	High accuracy and efficiency	Not detecting difference in some system calls of malware and benign apps
Extracting features and permissions from Android app [24]	RF	91.70%	High efficiency	No impact from features like HTTP, DNS, TCP/IP
Android Malware Detection Using Random Forest Based on API Call Sequences [25]	RF	97.40%	High Accuracy	DEREBIN dataset size limited to 120000 samples
Android Malware Detection with Deep CNN-LSTM Model [26]	CNN-LSTM	99.10%	High Accuracy	Andro360 dataset size limited to 360000 samples
Android Malware Detection System Using Isolation Forest [27]	Isolation Forest	94.20%	High Accuracy	AMD dataset size limited to 25000 samples
Android Malware Detection Based on Generative Adversarial Network [20]	GAN	96.80%	High Accuracy	ML-android dataset size limited to 25000 samples
Detecting illicit data leaks on Android (DeepDetector)	RF	83.34%	*Modular architecture	Does not identify leak source

DeepDetector uses a dynamic leak detection technique based on live traffic analysis. It applies machine learning algorithms (Random Forest trained on data from healthy and infected networks).

Its main strengths are:

- Dynamic real-time approach, Modifiable and extendible architecture involving the integration of new ML models.
- Supervised training

5 Results

The trained Random Forest model achieves 82.9% precision and 89.9% recall in classifying network connections, and 78.9% precision with 81.6% recall in detecting abnormal DNS requests as shown in the Table 1.

These results were obtained using the following criteria in building the decision trees:

- Number of trees: optimal at 100
- Split hyperparameter (data split): GINI (measure of impurity)
- Number of features considered at each split: optimal with sqrt
- Maximum tree depth: optimal with no depth limit
- Minimum number of samples per node: optimal at 2

These performances are obtained on the 20% test dataset of samples not used for training. They demonstrate the good generalization capabilities of the model to detect anomalies in the network traffic of unknown Android devices.

The results obtained are satisfactory, with an overall average of 83.32%. The results could have been better. However, our dataset is limited to just 500,000 samples. In the future, we'll be developing a larger, more accurate dataset.

Additionally, the DeepDetector system is fully automated as shown in the Fig. 4:

- Collect Android device traffic in real time via the Raspberry Pi configured as a WiFi access point.
- Generate the training dataset from the Zeek logs.
- Train the Random Forest model with hyperparameter optimization.
- Evaluate performance on the test dataset.
- Analyze traffic with the trained model and report detection results to the user in graphical interactive form.

This complete automation of data collection, processing and analysis steps is a key advantage of DeepDetector for practical field use (Fig. 5).

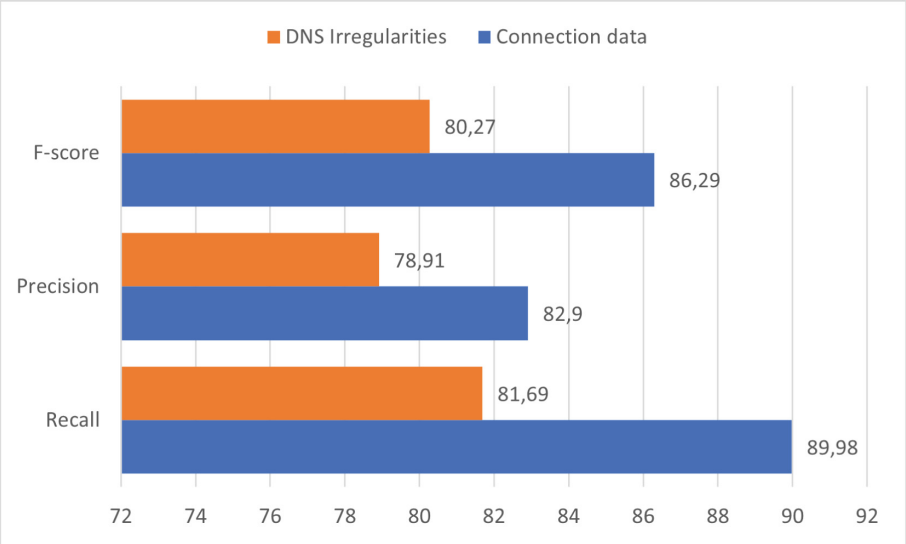


Fig. 5. Model evaluation results.

6 Discussion

This result is promising for the application of artificial intelligence methods, in particular supervised machine learning, in the detection of data leaks from Android smartphones.

A DeepDetector solution - free, easy to use and effective in strengthening user security and privacy in the face of the growing risk of malware.

A number of improvements can still be made to the system:

- Increase the size and variety of the training dataset to reinforce model robustness.
- Optimize model parameters and architecture to improve performance.
- Add new software sensors to enrich the data collected.
- Implement advanced techniques like deep learning.
- Deploy the system on a cloud platform for scaling.

On the other hand, as we pointed out in the section State of the art 2.4. Among the great challenges of artificial intelligence is the problem of interpretability and explicability [20] of model decisions. AI algorithms function like black boxes. They analyze a large amount of data and build patterns according to the data and the objectives we want to achieve. So it's hard to say precisely which features the algorithm has focused on, and to explain why the results are exactly as they are.

This work demonstrates the potential of artificial intelligence to protect mobile data, and paves the way for the development of even more effective data leakage detection systems in the future.

7 Conclusion

Here we present Deepdetector, an artificial intelligence-based data leak detection system that can be run solely on the Raspberry Pi. The results show promise for detecting anomalies in network traffic, with 80.1% precision, 86% recall and an F-score of 83%). In this respect, DeepDetector proves the ability of supervised machine learning methods to protect people's sensitive information from the growing danger posed by mobile malware.

Training the model with a larger and more varied dataset will improve results in future implementations. Another way of improving this collection is to combine the different methods cited in the state-of-the-art section.

DeepDetector has been developed in python, deployed and tested on a Raspberry PI 3 model B. The tests revealed that online betting sites, adult film sites and illegal film downloads gave the highest percentage of leakage trace detection. In essence, DeepDetector could eventually be considered an effective and available measure. Automatic detection of data leaks would further enhance user privacy.

DeepDetector's contribution is described in the conclusion, which summarizes the main findings. This conclusion also paves the way for future work to further improve the system.

Acknowledgement. This work was conducted as part of the Artificial Intelligence for Development in Africa (AI4D Africa) program, with the financial support of Canada's International Development Research Centre (IDRC) and the Swedish International Development Cooperation Agency (Sida).

References

1. Mobile Operating System Market Share Worldwide | Statcounter Global Stats. Statcounter Global Stats. https://gs.statcounter.com/os-market-share/mobile/worldwide. Accessed 14 Feb 2023
2. Senanayake, J., Kalutarage, H., Al-Kadri, M.O.: Android mobile malware detection using machine learning: a systematic review. Electronics **10**(13) (2021). https://doi.org/10.3390/electronics10131606
3. Hossain, M.S., Ochoa, M., Patterson, K., Boettiger, C.: Detecting and visualizing anomaly in network traffic. In: 2015 IEEE International Conference on Big Data (Big Data), pp. 1739–1748. IEEE (2015)
4. Eldardiry, H., Bart, E., Liu, J., Hanley, J., Price, B., Brdiczka, O.: Multi-instance multi-label learning for identifying security risks in corporate networks. In: Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security, pp. 135–146 (2013)
5. Shen, Y., Mariconti, E., Vervier, P.A., Stringhini, G.: Tiresias: predicting security events through deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 592–605 (2018)
6. Bon[u]klu, O., Okutan, A.: Predicting insider threat with Deep Learning. In: Proceedings of the 14th International Conference on Availability, Reliability and Security, pp. 1–10 (2019)

7. Almubayed, A., Hadi, A., Issa, T.B.: Detecting data exfiltration using neural networks. In: 2015 10th International Conference on Information Assurance and Security (IAS), pp. 26–31. IEEE (2015)
8. Li, Z., Qin, Z., Huang, K., Yang, X., Ye, S.: Intrusion detection using convolutional neural networks for representation learning. In: Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.-S.M. (eds.) ICONIP 2017. LNCS, vol. 10638, pp. 858–866. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70139-4_87
9. Patel, K., Patel, P., Patel, H.: Malware detection using machine learning and deep learning. In: 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), pp. 563–566. IEEE (2019)
10. Casas, P., Mazel, J., Owezarski, P.: Unsupervised network intrusion detection systems: detecting the unknown without knowledge. Comput. Commun. **35**(7), 772–783 (2012)
11. Rezaei, S., Liu, X.: Deep learning for encrypted traffic classification: an overview. IEEE Commun. Mag. **57**(5), 76–81 (2019)
12. Aljawarneh, S., Aldwairi, M., Yassein, M.B.: Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. J. Comput. Sci. **25**, 152–160 (2018)
13. Hoang, X.D., Choi, J.: A novel approach for Android malware detection using deep learning. In: 2016 18th International Conference on Advanced Communication Technology (ICACT), pp. 84–89. IEEE (2016)
14. Ryu, J.H., Baek, K., Hwang, J., Kim, P.J.: Detecting data exfiltration from the insider threat using threat tagging and nested context. Symmetry **10**(1), 22 (2018)
15. Cai, H., Sanfilippo, A., Glynn, E., Rathbun, L.C.: Insider threat detection by ontology-based semantic analysis of user behavior. In: Proceedings of the First Workshop on Misinformation and Misbehavior Mining on the Web, pp. 1–6 (2016)
16. Popic, V., Yang, T., Vukovic, V., Desai, N., Ahamad, M.: File upload security: new attack vectors and countermeasures. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, pp. 281–290 (2016)
17. Šajatović, M., Budiselić, E., Sušac, V.: A survey of honeypot deployment for detection of cyber attacks. In: 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), vol. 2020, pp. 1636–1641. IEEE (2020)
18. Feng, X., Zheng, Z., Cai, Z., Li, D., Li, J.: Defending against new malware with shared knowledge. In: 2014 IEEE International Conference on Communications (ICC), pp. 853–858. IEEE (2014)
19. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. (CSUR) **41**(3), 1–58 (2009)
20. Lipton, Z.C.: The mythos of model interpretability. Queue **16**(3), 31–57 (2018)
21. Android Mischief Dataset. Stratosphere IPS. https://www.stratosphereips.org/android-mischief-dataset. Accessed 29 Oct 2023
22. Garg, S., Peddoju, S.K., Sarje, A.K.: Network-based detection of Android malicious apps. Int. J. Inf. Secur. **16**, 385–400 (2017)
23. Sikder, A.K., Aksu, H., Uluagac, A.S.: 6thSense: a context-aware sensor-based attack detector for smart devices. In: Proceedings of the 26th USENIX Security Symposium, Vancouver, BC, Canada, pp. 397–414 (2017)
24. Salehi, M., Amini, M., Crispo, B.: Detecting malicious applications using system services request behavior. In: Proceedings of the 16th EAI International Conference on Mobile Ubiquitous System Computing, Networking Services, Houston, TX, USA, pp. 200–209 (2019)

25. Thangavelooa, R., Jinga, W.W., Lenga, C.K., Abdullaha, J.: DATDroid: dynamic analysis technique in android malware detection. Int. J. Adv. Sci. Eng. Inf. Technol. **10**, 536–541 (2020)
26. Lee, J., Park, S., Jung, J.: Detecting malicious behavior in Android apps through analyzing inter-app information flows. Expert Syst. Appl. **189**, 116124 (2022)
27. Zhang, H., Chan, P.P., Cheung, N.M.: Android malware detection based on generative adversarial network. Neural Comput. Appl. (2023)

Enhancing Predictive Process Monitoring with Conformal Prediction

Fotios Skouvas[1], Harris Papadopoulos[2(✉)], and Andreas S. Andreou[1]

[1] Cyprus University of Technology, Limassol, Cyprus
[2] Frederick University, Nicosia, Cyprus
h.papadopoulos@frederick.ac.cy

Abstract. This paper introduces a framework thatintegrates Conformal Prediction (CP) with Predictive Process Monitoring (PPM) to enhance prediction accuracy and reliability by producing prediction intervals with a guaranteed coverage rate. The approach followed fills a significant gap in current research as it provides an effective technique for assessing prediction uncertainty, which is vital for making well-informed decisions in various business sectors. Comprehensive experimental research conducted on various datasets demonstrates the framework's ability and effectiveness in providing accurate and reliable predictions of the remaining time required for the completion of a process trace. This work highlights the significance of measuring uncertainty in predictions, providing a substantial contribution to the areas of PPM and CP. It also offers a solid and trustworthy approach for integrating uncertainty quantification into process mining predictive models that contributes to significantly enhanced decision-support.

Keywords: predictive process monitoring · conformal prediction · remaining time · prediction intervals

1 Introduction

In the current digital age businesses create and use enormous amounts of data, requiring novel methods for effective management and analysis. Predictive Process Monitoring (PPM) is a crucial technology that helps enterprises anticipate and improve their operational results. Process mining is a crucial analytical method that leverages past data stored in event logs to provide deep insights into business processes. This field has been greatly improved because of the recent progress in Data Science, Predictive Analytics, and Artificial Intelligence, which are driven by a data-focused method for solving business problems. PPM represents a major advancement in anticipating future consequences of corporate processes with exceptional accuracy. PPM has diverse applications across several sectors including manufacturing, healthcare, finance, and logistics, improving data-driven decision-making and operational efficiency [1, 2].

The use of machine learning methods in PPM has advanced the field into an entirely novel phase, allowing for predictions regarding process execution outcomes, timetables, and upcoming steps. Despite these developments, the use of PPM in real-world scenarios

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 201–214, 2024.
https://doi.org/10.1007/978-3-031-63215-0_15

is hindered by the difficulty in defining predictions. This limitation emphasizes the need for approaches that are capable of accurately predicting outcomes while also offering an acceptable measure of confidence in their predictions. Conformal prediction provides a theoretical framework for producing prediction intervals with guaranteed coverage, improving the reliability and trustworthiness of PPM outcomes [3, 4].

Research into generating models for prediction intervals remains limited despite substantial work on predicting the remaining time required for the completion of a process [5]. Moreover, there are no studies on the incorporation of conformal prediction techniques and the benefits they provide to PPM. This research aims to fill the current research gap by investigating how conformal prediction techniques can enhance decision-making and reliability in PPM. It will also evaluate the effectiveness of a framework that integrates conformal prediction theory, PPM techniques, and machine learning models to predict the remaining time of processes using real-world datasets through thorough experiments and detailed analysis.

Specifically, this work integrates Conformal Prediction with PPM methods evaluating its combination with three advanced machine learning models: Random Forest Regressor (RFR), Extreme Gradient Boosting (XGBoost), and Bayesian Neural Networks (BNN). Detailed analysis of experimental results suggests that the proposed framework advances PPM and shows that conformal prediction can redefine predictive accuracy and decision-making reliability in business process monitoring.

The rest of the paper is structured as follows: Sect. 2 covers process mining, predictive process monitoring, and conformal prediction basics. Section 3 provides an overview of the related literature, while Sect. 4 details the proposed framework. Section 5 presents the experimental setup, the metrics used, and the results obtained, discussing and comparing the different models employed and demonstrating the framework's applicability. Finally, Sect. 6 presents the conclusions and recommends future research to improve PPM decision-making and reliability.

2 Technical Background

The technical background of this paper focuses on foundational aspects and methodologies relevant to PPM, machine learning models, and conformal prediction, highlighting their significance in enhancing the accuracy and reliability of PPM predictions.

2.1 Predictive Process Monitoring

PPM is a set of techniques that uses event records to provide actionable insights into the future behavior of business processes, integrating business process mining and predictive analytics [5, 6]. By combining event data with process models, process mining methods contribute to various business fields by providing insights that permit identifying bottlenecks and errors, predicting and analyzing performance, identifying compliance issues, and automating or avoiding repeated labor [7, 8]. The key elements of business process mining include:

Event logs: These are the most important elements of process mining since they include all the historical data that was produced by different information systems while

a process was being executed [8]. Event logs are the main source of information used for analysis within the context of a process mining framework. Table 1 provides an example of an event log, which contains information about two process instances (Trace ID 1 and 2), each with a sequence of events (A, B, C, and D), timestamps, and resources (User 1, User 2, User 3, and User 4).

Table 1. Example of event log

Trace ID	Events	Timestamp	Resource
1	A	2023-01-01 08:00:00	User 1
1	B	2023-01-01 09:00:00	User 2
1	C	2023-01-01 10:00:00	User 3
2	A	2023-01-01 08:30:00	User 1
2	B	2023-01-01 09:30:00	User 2
2	D	2023-01-01 10:30:00	User 4

Traces: Individual executions of a business process are represented by traces (or cases) which are sequential lists of events. Each entry in an event log is a record of a single execution of a process, complete with information regarding the time, date, and resources used. Process performance, conformity, and variance may all be studied over time and between instances with the use of information provided by traces [7]. By definition, a trace is a non-empty sequence $\sigma = [e_1, ..., e_n]$ of events such that $\forall i \in [1...n]$, $e_i \in E$, where E is the universe of all events, and $\forall i, j \in [1...n] e_i.c = e_j.c$. In other words, each and every event that is included in a trace refers to the same case c [6].

Events: The execution of individual actions inside a process instance is represented by events, which are the fundamental components of traces and serve as their building blocks. each event includes details on the activity that was carried out, the resource that was used (such as a person or a system), a timestamp that indicates when the event took place, as well as other data properties that are pertinent to the process. understanding the dynamics of the process execution, identifying bottlenecks, and discovering deviations from the intended behavior of the process all need events as a foundational component [7]. By definition, an event is a tuple $(a, c, t, (d_1; v_1), \ldots, (d_m; v_m))$ where a is the activity name, c is the case id, t is the timestamp and $(d_1; v_1), \ldots, (d_m; v_m)$, where $m \geq 0$, are the event or case attributes and their values respectively [6].

2.2 Machine Learning Models

Machine learning algorithms, such as Extreme Gradient Boosting (XGBoost), Random Forest Regressor (RFR), and Bayesian Neural Networks (BNNs), play pivotal roles in PPM due to their high efficiency in predictive modeling. XGBoost [9] utilizes gradient boosting to improve the accuracy of models by employing a series of decision trees. This approach incorporates regularization and sparsity-aware techniques to ensure scalability

and address the issue of overfitting. Consequently, XGBoost is particularly well-suited for managing extensive datasets in industries, such as banking, healthcare, and retail. RFR [10], through an ensemble of decision trees [11] and random feature selection, reduces tree correlation and boosts predictive reliability, offering insights into feature impacts on predictions and excelling in managing vast data volumes across various fields. This approach provides valuable insights into the influence of features on predictions and excels in handling large amounts of data in diverse domains. BNNs [12, 13] combine Bayesian inference with neural networks, utilizing probability distributions for weights to address prediction uncertainty and provide inherent regularization. This approach improves performance, even when dealing with limited data, and enhances the comprehensibility of decisions. Together, these algorithms highlight the progress in PPM by providing strong, adaptable, and effective tools for predictive modeling in various fields.

2.3 Conformal Prediction

Conformal Prediction (CP) [4, 14] is a novel framework for quantifying the uncertainty of machine learning models. In particular, CP can be used to transform the predictions of any underlying model to prediction regions with a guaranteed coverage of $(1 - \delta)$, for any prespecified significance level δ. Importantly this guarantee relies solely on the assumption that the data points are generated independently from the same probability distribution (i.i.d.). Even though this is a rather strong assumption, it is much weaker than the typical assumption of a particular data generating distribution. Additionally, it can be combined with techniques, such as [15], for detecting exchangeability violations and recalibrating accordingly. This distribution-free nature of CP in combination with its versatility in being combined with any machine learning approach, render it a powerful tool for a vast range of applications.

CP comes in two main versions: Full CP (FCP) [4], or Transductive CP, and Split CP (SCP) [14], or Inductive CP. While the former achieves higher statistical efficiency, it is rather computationally inefficient and therefore unsuitable for large datasets or computationally demanding models. SCP on the other hand, provides a huge computational efficiency improvement at the cost of sacrificing some statistical efficiency by using distinct parts of the data for training the underlying model and for constructing its prediction regions. This ability to handle extensive datasets is the reason for which this study follows the SCP version of the framework.

To define the SCP process, consider a set of training examples $\{(x_1, y_1), \ldots, (x_l, y_l)\}$ where $x_i \in \mathbb{R}^d$ represents the vector of attributes for example i, and $y_i \in \mathbb{R}$ is the corresponding label. The objective is to provide a prediction region for the label y_{l+g} for a new example x_{l+g}, based on the assumption that all pairs (x_i, y_i) are i.i.d. SCP performs the following steps: *(i) Splitting the Dataset:* The training set is divided into: the *proper training set* with m examples and the *calibration set* with $q = l - m$ examples. *(ii) Training the Underlying Algorithm:* Use the proper training set to train the underlying algorithm and generate a prediction rule R. *(iii) Calculating Nonconformity Scores:* For each pair (x_i, y_i), $i = m+1, \ldots, m+q$ in the calibration set, calculate the nonconformity score a_i based on the degree of disagreement between the predicted $R(x_i)$ and actual y_i labels; the function employed for calculating the nonconformity scores is called the

nonconformity measure. (iv) Calculating New Nonconformities: For a new example x_{l+g} compute its prediction $R(x_{l+g})$ and calculate the nonconformity score $\alpha_{l+g}^{\tilde{y}}$ for every candidate label $\tilde{y}$ as the degree of disagreement between $R(x_{l+g})$ and $\tilde{y}$; ; using the same nonconformity measure with the previous step. *(v) Calculating p-values:* The nonconformity score of each assumed label $\tilde{y}$ is compared to the calibration scores to determine how unlikely it is:

$$p(\tilde{y}) = \frac{\#\left\{i = m+1, \ldots, m+q, l+g : \alpha_i \geq \alpha_{l+g}^{\tilde{y}}\right\}}{q+1}. \tag{1}$$

This comparison yields $p(\tilde{y})$, which is a valid p-value for the null hypothesis that $y_{l+g} = \tilde{y}$. . *(vi) Constructing Prediction Regions:* For a given significance level δ, produce the prediction region $\mathbb{C}^{\delta}(\mathrm{x_{l+g}}) = \left\{\tilde{\mathrm{y}}\colon\ \mathrm{p}(\tilde{\mathrm{y}}) > \delta\right\}$, which has a guaranteed $(1-\delta)$ coverage, for a proof see [4].

Of course, for regression it is impossible to explicitly consider every possible value $\tilde{\mathrm{y}} \in \mathrm{R}$, but given a particular nonconformity measure we can work backwards to calculate the prediction region $\mathbb{C}^{\delta}$ efficiently. In particular, in this work we consider the normalized nonconformity measure [14, 16]:

$$\alpha_i = \left|\frac{y_i - R(x_i)}{d_i}\right|, \tag{2}$$

where $\mathrm{d_i}$ is the expected difficulty of x_i for the prediction rule R. Given this nonconformity measure, SCP calculates the prediction region $\mathbb{C}^{\delta}$, „ which is a prediction interval in this case, by performing steps *(i)–(iii)* defined above and then proceeding as follows: *(iv) Sort Calibration Nonconformity Scores:* Sort the nonconformity scores $\alpha_{\mathrm{i}},\ \mathrm{i} = \mathrm{m}+1, \ldots, \mathrm{m}+\mathrm{q}$ of the calibration examples in descending order, obtaining the sequence $\alpha_{(\mathrm{m}+1)}, \ldots, \alpha_{(\mathrm{m}+\mathrm{q})}$. .. *(v) Construct New Example Prediction Regions*: For a new example x_{l+g} compute its prediction $R(x_{l+g})$ and expected difficulty $\mathrm{d_{l+g}}$, and output the prediction interval:

$$\mathbb{C}^{\delta}(x_{l+g}) = \left(R(x_{l+g}) - \alpha_{(m+s)}d_{l+g}, R(x_{l+g}) + \alpha_{(m+s)}d_{l+g}\right), \tag{3}$$

where $\mathrm{s} = \lfloor \delta(\mathrm{q}+1) \rfloor$. This is exactly the same as $\{\ \mathrm{y}:\ \mathrm{p}(\ \mathrm{y}) > \delta\}$ [16, 17].

3 Literature Review

Integrating PPM with conformal prediction using any of the existing machine learning models such as XGBoost, RFR, and BNNs to offer accurate prediction intervals for the remaining time of a process trace at a predetermined confidence level is an important breakthrough for handling uncertainties in process results. Traditional machine learning models like Bayesian methods, ensemble methods, and direct interval estimation techniques have played a crucial role in prediction outcomes, such as the remaining time of a trace. These models frequently generate point estimates without effectively considering the uncertainty present in these predictions. Conformal prediction methods

can convert a point predictor into an interval estimator, producing accurate prediction regions with a predetermined confidence level [1]. This integration intends to utilize the capabilities of machine learning models like XGBoost, RFR, and BNNs to capture intricate data patterns. It incorporates conformal prediction to measure the uncertainty of these predictions, enabling better-informed decision-making in process management.

Wang and Cao's approach [18] for predictive process monitoring utilizes interval-based remaining time prediction and LSTM networks for forecasting and trace clustering to improve accuracy. Nevertheless, although this method includes the measurement of uncertainty using LSTM's ability to handle sequential data, it does not provide a guarantee for the prediction intervals. Current literature on PPM examines different approaches, which include traditional process mining methods to advanced machine learning models, in order to predict future process behaviors. Research conducted by Di Francescomarino et al. and Teinemaa et al. [5, 6] examines and contrasts various PPM techniques, emphasizing their suitability and effectiveness in different situations. In their study, Maggi et al. [2] examine the progression of predictive monitoring within business processes, highlighting the shift towards outcome-oriented and resource-efficient methodologies. Reinkemeyer and Park et al. [19, 20] have recently conducted research that emphasizes the combination of process mining with execution and the optimization of resource allocation. These studies demonstrate the field's advancement towards more comprehensive and applicable solutions. Nevertheless, these studies frequently fail to consider the element of uncertainty, which is essential for generating more dependable forecasts.

This paper presents a novel framework that utilizes conformal prediction to generate prediction intervals for the remaining time of business process traces with a predetermined confidence level, using advanced machine learning models. This innovative approach successfully addresses the limitations of conventional machine learning methods by providing more comprehensive predictions that adjust for uncertainty. The proposed framework stands out for its ability to adapt to a variety of business process behaviors and complexities, providing a strong and flexible tool for navigating the unpredictable environments in modern business operations.

4 Research Methodology

This section describes the overall research approach for combining PPM with conformal prediction. Specifically, this work examines the complex nature of predictive process monitoring by comparing the performance of three sophisticated machine learning models (XGBoost, RFR, and BNNs) used as the underlying models of conformal prediction applied on three separate datasets (BPIC 2012, BPIC 2017, and Helpdesk). Our approach utilizes a systematic method that includes data collection, data preprocessing, implementation of PPM algorithms with machine learning models and conformal prediction methodologies, evaluation of model performance, and comparative analysis of the results. Essentially, this study seeks to enhance comprehension of the advantages, difficulties, and practical uses linked to integrating conformal prediction methods in the field of PPM.

Figure 1 represents the preprocessing phase of the BPIC 2012 dataset, which includes the calculation and the standardization of the features. BPIC 2012 contains Dutch financial institution loan application event logs. It provides an extensive overview of the

process involved in various loan application activities, including timestamps and associated process attributes. Initially, the dataframe is organized according to case and timestamp, enabling the calculation of time differences among events. An essential preprocessing step involves determining the total length of each case and incorporating the variable 'elapsed_time' to indicate the time that has passed since the beginning of a case. The variable 'event_count' represents the position of an event within its trace. The 'requested_amount' of each trace, which denotes the entire loan amount that the applicant intends to obtain from the financial institution, is added to all the events of the trace. In the proposed framework, the prediction target is represented by the variable 'remaining_time', which is defined as the disparity between the total duration and the elapsed time. 'elapsed_time', 'event_count' and 'requested_amount' features are used as inputs for predicting the 'remaining_time'. To ensure all features are comparably scaled, the 'elapsed_time' and 'requested_amount' features are standardized.

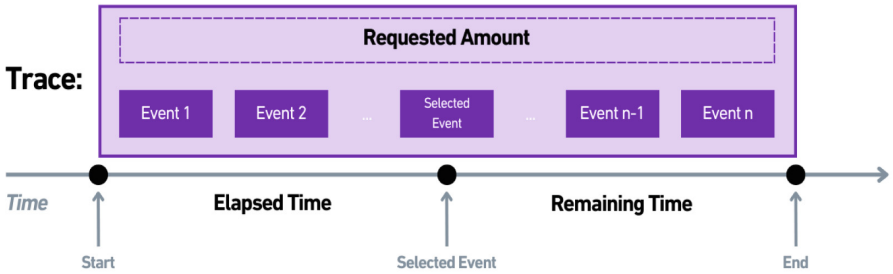


Fig. 1. Remaining time prediction of BPIC 2012 trace.

Figure 2 presents a comprehensive overview of the proposed framework. Following the completion of preprocessing, the dataset is divided into separate proper training and calibration sets. The underlying machine learning model is trained on the proper training set generating the model *R*. This model is used both for calculating the predictions of the calibration and test examples and for generating a secondary linear model for the calculation of their d_i's in our nonconformity measure definition (2). In particular, the definition of d_i used in this study is $d_i = \exp(\mu_i)$, where exp is the exponential function and μ_i is the prediction of the secondary linear model for $\ln(|R(x_i) - y_i|)$. This secondary linear model is trained on the training set, but with target values set to the logarithm of the residuals of *R*, i.e. it predicts the expected error of *R*. The use of the logarithmic scale ensures that the estimate is always positive. After training the underlying and the secondary models, these are used for calculating the nonconformity scores for all calibration examples with (2). The prediction interval for each test example is then generated following the efficient process defined in Sect. 2.3.

The implementation of SCP was based on the MAPIE library [21] using MapieRegressor with the ResidualNormalisedScore. Both the underlying and secondary models were trained on the proper training data as described above and provided as fitted models to MapieRegressor setting prefit = True. The resulting prediction intervals can provide

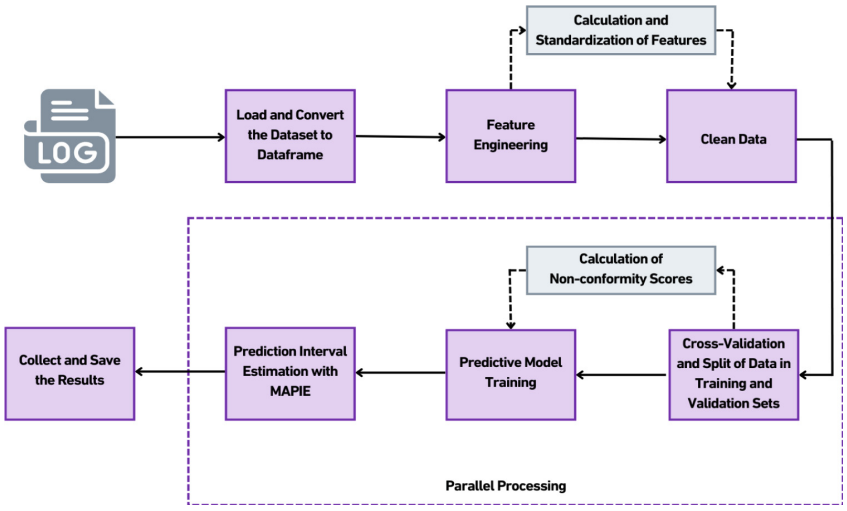


Fig. 2. Overview of the framework

end-users with a reliable quantification of the degree of uncertainty involved in each individual case enabling informed decisions.

5 Experimental Results

This section provides a comprehensive analysis of the primary experimental results obtained from the implementation of the proposed framework, which combines conformal prediction with PPM techniques. The performance of the proposed framework was evaluated using a standard 5-fold cross-validation process. The calibration set for each fold was created by selecting randomly 20% of the training events. This methodology guarantees that every segment of the dataset is methodically employed for both training and validation, enabling a comprehensive assessment of the model's efficacy on various datasets. An in-depth analysis of three commonly used machine learning models was conducted: XGBoost, RFR, and BNNs. This evaluation was performed on three different datasets: BPIC 2012, BPIC 2017, and Helpdesk. The study examines different metrics evaluating the empirical coverage, statistical efficiency and computational efficiency of each SCP, thereby offering a comprehensive comprehension of their individual merits and limitations.

5.1 Experimental Setup

Datasets: The robustness, efficiency, and validity of the generated models and insights are all directly influenced by the quality and relevance of the selected dataset. This study utilizes three datasets to validate the proposed models: BPIC 2012 [22], BPIC 2017 [23], and Helpdesk [24]. The BPIC 2012 dataset, produced by the banking industry, centers on the process of personal loan application within a Dutch financial institution.

It provides a comprehensive account of the various steps required in the approval or rejection of loan applications. The BPIC 2017 refers to the loan application procedure of a financial institution in the Netherlands as well but mean-while enables the potential for multiple offers to be made per application. The Helpdesk dataset, sourced from the IT sector, encompasses the procedures involved in managing and resolving customer support tickets within the helpdesk system of an Italian software company. Each dataset offers a unique perspective on different real-world processes, ranging from loan origination to IT support services, providing a rich basis for evaluating the generalizability and effectiveness of the proposed PPM framework. Table 2 provides a basic statistical analysis of the datasets and the number of events allocated for each fold in the training and test sets.

Table 2. Basic statistics of datasets

Dataset	Trace count	Event count	Training events	Test events
BPIC 2012	13087	262200	209760	52440
BPIC 2017	31509	1202267	961817	240453
Helpdesk	4580	21348	17078	4270

5.2 Evaluation Metrics

The evaluation metrics that have been utilized for evaluating the three SCPs (with the three different underlying models) are as follows:

Empirical Coverage Rate (ECR) measures the percentage of actual values that are included within the prediction intervals:

$$ECR^{\delta} = \frac{1}{n}\sum_{i=1}^{n} \#\left\{y_i \in \complement^{\delta}(x_i)\right\}, \tag{4}$$

where n is the number of test examples.

Mean Prediction Interval Width (MPIW) determines the average width of the prediction intervals and consequently provides insights into the model's interval accuracy:

$$MPIW^{\delta} = \frac{1}{n}\sum_{i=1}^{n}\left(U_i^{\delta} - L_i^{\delta}\right), \tag{5}$$

where $\complement^{\delta}(x_i) = \left(U_i^{\delta}, L_i^{\delta}\right)$.

Execution Time: refers to the duration it takes for the algorithm to complete its entire process, under similar experimental conditions.

Fig. 3. Empirical Coverage Rate across BPIC 2012, BPIC 2017 and Helpdesk datasets at confidence levels ranging from 90% to 99%.

5.3 Empirical Coverage

Figure 3 displays the empirical coverage for the XGBoost, RFR, and BNNs models on the BPIC2012, BPIC2017, and Helpdesk datasets. The confidence levels range from 90% to 99%. The closeness of the coverage rates to the diagonal serves as an indicator of the degree of alignment with the nominal confidence levels. The data points consistently exhibit a strong alignment with the diagonal line across all three models and datasets, indicating the empirical validity and reliability of the resulting prediction intervals. This confirms that the i.i.d. assumption holds for the data in question. The minor deviations from the diagonal are due to statistical fluctuations and will diminish as the number of test instances increases.

5.4 Statistical Efficiency

In Fig. 4, the MPIW appears as the percentage of the label range of each dataset it represents for the XGBoost, RFR, and BNNs underlying models. The three SCPs were tested at confidence levels of 90%, 95%, and 99% on the BPIC 2012, BPIC 2017, and Helpdesk datasets. The figure shows each SCP's ability to adjust its prediction intervals based on the desired level of confidence, essential for businesses that rely on accurate predictions for strategic operational decisions. As the confidence level increases, prediction intervals become wider so as to achieve the required coverage.

Figure 5 demonstrates a comparison between the MPIW and the mean of the actual remaining time across different event positions for the XGBoost model, with a confidence

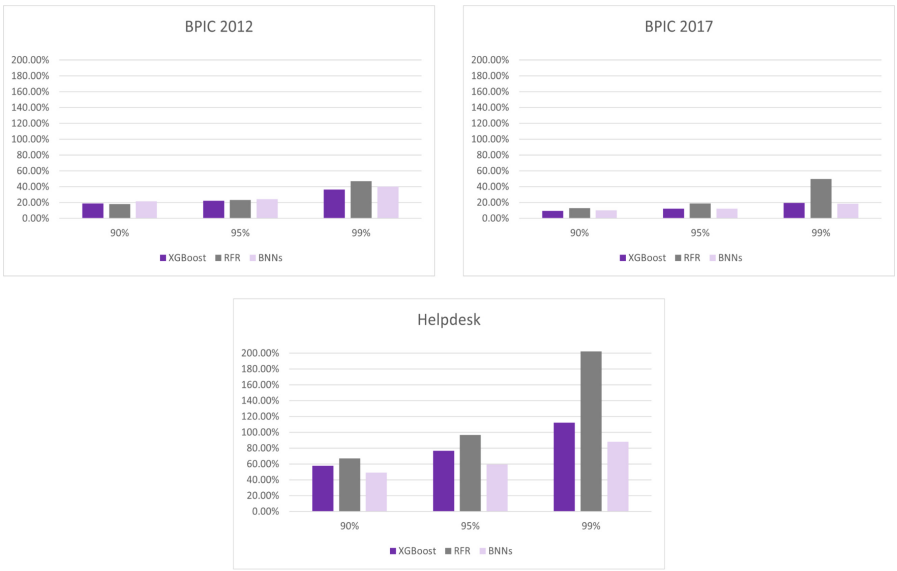


Fig. 4. MPIW as percentage of maximum actual width

level of 90% (a = 0.10). This comparison highlights how effectively the framework's predictive ability may adjust throughout a business process's lifecycle. The MPIW shows a consistent decrease across all datasets, suggesting that the intervals are becoming more precise. This pattern corresponds to the expected behavior, where the remaining time decreases as the process moves ahead. The outcomes indicate that the model modifies its prediction intervals to reflect uncertainty based on the events happening throughout a process.

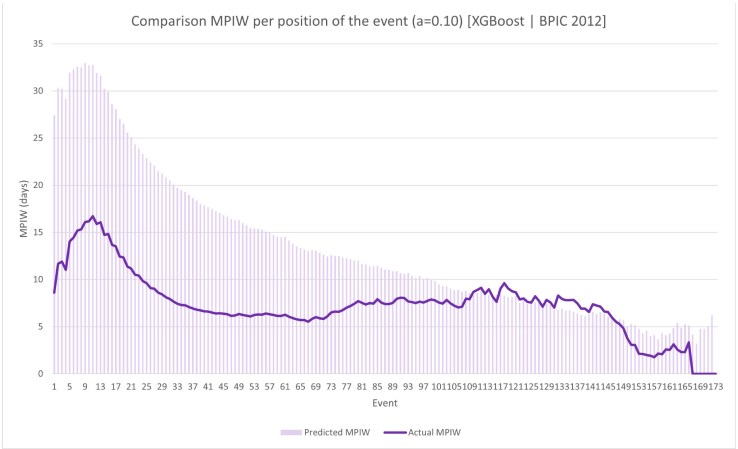


Fig. 5. Comparison of MPIW (XGBoost) per position of the event (confidence level 90%)

5.5 Computational Efficiency

Figure 6 displays a comparison of the execution time, measured in seconds, for the three predictive machine learning models utilized in the proposed framework. Evaluating models is essential in order to understand the relationship between prediction accuracy and computational demands. XGBoost outperforms the other models in computational efficiency. The short execution time of XGBoost indicates that it is a very efficient model for fast-generating predictions, which is beneficial for almost real-time predictive process monitoring.

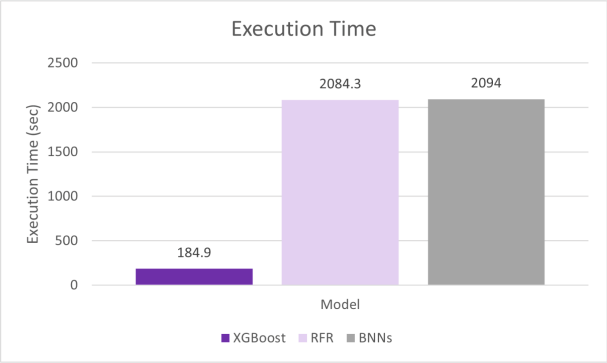


Fig. 6. Execution Time per model (BPIC 2012)

5.6 General Discussion

The comprehensive experimental analysis presented in Figs. 3, 4, 5, 6 and 7 confirms the effectiveness of combining conformal prediction with PPM approaches in machine learning models on multiple datasets. The empirical evidence demonstrates the precision and reliability of the models, with XGBoost standing out for its computational efficiency, making it suitable for almost real-time applications.

When evaluating the results of our model, it is crucial to compare them with previous studies in the field. The literature review for this study identified a notable deficiency in research results that correspond with our approach. Several studies provide techniques for creating prediction intervals for the remaining period of activities, but none of them are customized for a certain level of confidence, a unique feature of our research. In comparison to the interval-based remaining time prediction method proposed by Wang and Cao [18], our analysis indicates that there is no statistically significant difference in the MPIW produced by the two methodologies. Nevertheless, a crucial differentiation arises with respect to the reliability of the prediction intervals. Although Wang and Cao's approach utilizes LSTM networks to produce prediction interval bounds, it does not have an inherent mechanism to guarantee the accuracy of these intervals. On the other hand, the proposed methodology not only aligns with the MPIW but also exceeds it by offering a predetermined level of confidence, thereby ensuring that the generated intervals are not simply estimations but statistically guaranteed predictions.

Although this work offers significant insights into integrating conformal prediction to PPM models, there are some aspects that require further investigation. The first aspect is the effectiveness of the framework in different scenarios and with different kinds of event logs. Another aspect is the examination of the effectiveness of CP techniques in classification scenarios. The performance characteristics of regression and classification models vary greatly due to the differences between the two types of problems.

6 Conclusions

This paper proposed an innovative framework that integrates CP techniques with PPM using advanced machine learning models such as XGBoost, RFR, and BNNs. This novel method represents a notable contribution in the field of PPM by quantifying the uncertainty of predictions at predefined confidence levels. CP methods enable the generation of prediction intervals that are both meaningful and statistically valid, filling a crucial gap in existing methodologies. The framework's novelty is its capability to offer dependable prediction intervals for the remaining time of a process trace at a specified confidence level. This capability is crucial to businesses and decision-makers who rely on accurate and trustworthy predictions to enhance operations and minimize risks. A thorough experimental investigation was conducted on three different datasets, BPIC 2012, BPIC 2017, and Helpdesk, and illustrated the efficacy of our method in generating accurate and reliable predictions.

Future research will concentrate on further examining the aspects mentioned in the previous section. This will be performed by exploring innovative computational methods that can decrease the execution time of conformal prediction while maintaining the accuracy and dependability of the predictions. Moreover, it is essential to validate the generalizability and efficiency of our methodology by applying it to a wider variety of datasets and business processes in varied scenarios. One potential area for future research involves combining our system with real-time data streams to improve its effectiveness in dynamic and fast-paced business environments.

References

1. van der Aalst, W.M.P., Pesic, M., Song, M.: Beyond process mining: from the past to present and future. In: Pernici, B. (eds.) Advanced Information Systems Engineering. CAiSE 2010. Lecture Notes in Computer Science, vol. 6051. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13094-6_5
2. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Jarke, M., et al. Advanced Information Systems Engineering. CAiSE 2014. Lecture Notes in Computer Science, vol. 8484. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07881-6_31
3. Shafer, G., Vovk, V.: A tutorial on conformal prediction. J. Mach. Learn. Res. 371–421 (2008)
4. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic Learning in a Random World. Second edition. Springer (2022)
5. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Milani, F.: Predictive Process Monitoring Methods: Which One Suits Me Best? In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNCS, vol. 11080, pp. 462–479. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_27

6. Teinemaa, I., Dumas, M., La Rosa, M., Maggi, F.M.: Outcome-oriented predictive process monitoring: review and benchmark. In: ACM Transactions on Knowledge Discovery from Data (TKDD), pp. 1–57 (2019)
7. Van der Aalst, W., Pesic, M.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011)
8. Van der Aalst, W., Pesic, M.: Process Mining: Data Science in Action. Springer, Heidelberg (2016)
9. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. Association for Computing Machinery, New York (2016)
10. Breiman, L.: Random Forests. Machine Learning (2001)
11. Rudd, J.M., Ray, H.G.: An empirical study of downstream analysis effects of model preprocessing choices. Open J. Stat. 735–809 (2020)
12. Neal, R.: Bayesian Learning for Neural Networks. Springer, New York (1996)
13. Jospin, L.V., Laga, H., Boussaid, F., Buntine, W., Bennamoun, M..: Hands-on Bayesian neural networks—a tutorial for deep learning users. IEEE Comput. Intell. Mag. 29–48 (2022)
14. Papadopoulos, H., Gammerman, A., Vovk, V.: Normalized nonconformity measures for regression conformal prediction. In: Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications (AIA 2008), pp. 64–69. ACTA Press (2008)
15. Papadopoulos, H., Haralambous, H.: Reliable prediction intervals with regression neural networks. Neural Netw. **24**(8), 842–851 (2011)
16. Eliades C., Papadopoulos H.: A conformal martingales ensemble approach for addressing concept drift. In: Proceedings of the Twelfth Symposium on Conformal and Probabilistic Prediction with Applications, PMLR 204, pp. 328–346 (2023)
17. Papadopoulos, H., Proedrou, K., Vovk, V., Gammerman, A.: Inductive confidence machines for regression. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) Machine Learning: ECML 2002. ECML 2002. Lecture Notes in Computer Science, vol. 2430. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36755-1_29
18. Wang, C., Cao, J.: Interval-based remaining time prediction for business processes. In: Hacid, H., Kao, O., Mecella, M., Moha, N., Paik, Hy. (eds.) Service-Oriented Computing. ICSOC 2021. Lecture Notes in Computer Science, vol. 13121. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-91431-8_3
19. Reinkemeyer, L.: Status and future of process mining: from process discovery to process execution. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook. Lecture Notes in Business Information Processing, vol. 448. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08848-3_13
20. Park, G., Song, M.: Optimizing resource allocation based on predictive process monitoring. IEEE Access **11**, 38309–38323 (2023)
21. Taquet, V., Blot, V., Morzadec, T., Lacombe, L., Brunel, N.J.: MAPIE: an open-source library for distribution-free uncertainty quantification. ArXiv, abs/2207.12274 (2022)
22. Van Dongen, B.F: Bpi challenge 2012. Eindhoven University of Technology (2012). https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b
23. Van Dongen, B.F: Bpi challenge 2017. Eindhoven University of Technology (2017). https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
24. Polato, M.: Help desk event log. University of Padova (2017). https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb

Improved NO_2 Prediction Using Machine Learning Algorithms

Chukwuemeka Jaja-Wachuku, Lorenzo Garbagna(✉), Lakshmi Babu Saheer, and Mahdi Maktab Dar Oghaz

Anglia Ruskin University, Cambridge CB1 1PT, UK
lg673@pgr.aru.ac.uk

Abstract. Improved air pollution management approaches are required to ensure better air quality and tackle climate change. The ability to accurately forecast air quality, particularly the concentration of NO_2 in the air, is crucial especially for urban settings due to direct health implications. Various machine and deep learning models have been used for air quality prediction. However, the application of these approaches on NO_2 concentration levels, focusing on specific cities and specific climatic conditions, has been investigated on a limited scale. In this study, the performance of commonly used algorithms, such as Random Forest Regressor, Light Gradient-Boosting Machine (LightGBM), Extreme Gradient Boosting (XGBoost), and Long Short-Term Memory (LSTM) networks for predicting NO_2 concentration levels based on time series and meteorological data including climatic conditions is assessed. Further, ensemble modeling techniques are evaluated as a voting regressor using Random Forest, LightGBM, and XGBoost as base models for improving the prediction of NO_2 concentration levels. Each model was evaluated using cross-validated (5-fold) Mean Absolution Error and Root Mean Square Error metrics with LSTM emerging as the best-performing model.

Keywords: Air quality modeling · NO_2 predictions · Machine learning · Deep learning · ensemble models

1 Introduction

The escalating issue of air pollution has become a significant concern for environmental scientists, policymakers, and the general public due to its profound implications for human health, climate change, and biodiversity [15]. Air pollution, a complex mixture of solid particles and gases in the air, originates from various sources such as industry, transport, and agriculture and can lead to severe health problems, including respiratory diseases, cardiovascular conditions, and premature death [17]. 4.2 million premature deaths globally are linked to ambient air pollution, primarily from heart disease, stroke, chronic obstructive pulmonary disease, lung cancer, and acute respiratory infections in children [20]. This alarming statistic underscores the urgency of addressing air pollution and improving air quality.

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 215–225, 2024.
https://doi.org/10.1007/978-3-031-63215-0_16

One of the key pollutants monitored in urban areas is NO_2 which is emitted by the burning of fossil fuels and could lead to further generation of ground-level Ozone, which is a greenhouse gas responsible for climate change. Apart from contributing to climate change, and being detrimental to biodiversity, this pollutant is a major threat to human health and there are regulatory standards around the maximum allowed values for this gas. This research aims to investigate the effectiveness of ensemble machine learning and deep learning methods for accurately predicting NO_2 concentration levels in the UK City of Cambridge. The study will integrate historical NO_2 concentration levels and weather data to create a more robust and generalisable forecasting model. This research seeks to fill gaps, such as the under-utilisation of ensemble techniques in air pollution studies, and the comparison of classical ML algorithms against DL techniques for NO_2 concentration levels predictions, which has a limited focus in the research space when compared to other pollutants. It also aims to enhance our understanding of how local factors influence air pollution, thereby helping the City of Cambridge's climate planning efforts. The research will rely on specific machine learning models, including Random Forest Regressor, Light Gradient Boosting Machine Regressor (Light GBM), Extreme Gradient Boosting Regressor (XGBoost), and Long Short-Term Memory (LSTM) Neural Networks.

2 Related Work

Applying machine learning and deep learning techniques in air pollution analysis and air quality prediction has seen diverse approaches. Regression-based models, such as Support Vector Regression and Random Forest, are commonly used due to their ability to handle non-linear relationships and interpretability. These models have been particularly effective in short-term air quality prediction, capturing temporal variations in pollutant concentrations. On the other hand, artificial neural networks (ANNs), including Multi-layer Perceptrons (MLPs) and more complex deep learning models like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, have been increasingly popular [8]. These models can handle complex spatial-temporal patterns, making them suitable for high-resolution air pollution mapping and long-term air quality prediction. Comparing the results and findings from studies conducted in different locations or contexts reveals exciting insights. While the specific findings may vary due to local factors, such as pollution sources, meteorological conditions, and data availability, the general trend is that machine learning and deep learning models often outperform traditional deterministic models with regard to prediction accuracy and computational efficiency. However, the performance of different machine learning and deep learning techniques can vary significantly across studies. For instance, one study might find that a Random Forest model performs best for $PM_{2.5}$ prediction, while another study might report the superior performance of an LSTM network [21]. These differences can be attributed to various factors, including the characteristics of the air pollution patterns, the data quality and granularity, and the specific objectives of the study.

Traditional statistical models like ARIMA are still popular in forecasting pollutants like PM_{10} [3] or CO_2 [4]. However, machine learning or deep learning approaches are being actively investigated as more powerful prediction techniques. Li et al. [10] investigated the effectiveness of machine learning algorithms in predicting air quality. The researchers focused on the daily mean concentrations of six critical pollutants as features and used the Air Quality Index (AQI) as the outcome variable. Multiple regression models were tested, and the study found that Random Forest Regression (RFR) and Gradient Boosting Regression (GBR) exhibited excellent performance in terms of predictive accuracy, robustness to noise, and generalizability. Similarly, Aditya et al. [1] used logistic regression and autoregression techniques to classify polluted air samples and forecast $PM_{2.5}$ levels accurately. The researchers suggested that advanced knowledge of $PM_{2.5}$ levels could guide efforts to keep this pollutant within safe limits. Their findings confirmed the efficacy of machine learning models in air quality detection and $PM_{2.5}$ forecasting.

Kamińska [9] utlised a combination of two Random Forest models to predict NO_2 concentration levels from traffic flow and meteorological information by leveraging the bagging method and random feature selection. When compared to a classical RF implementation, the study showed an increase in R^2 value from 0.60 to 0.82 for the constructed method, demonstrating the influence of traffic flows on pollutant concentration levels.

Liu et al. [12] compared land-use statistical and machine learning models for NO_2 forecasting across Switzerland at a spatial resolution of 100×100 m. Temporal resolution and data size influenced model performance: linear regression-based models outperformed other machine learning methods on long-term predictions, such as monthly or annual, while the latter proved better capabilities for shorter time-frame forecasting such as daily predictions. XGBoost and LightGBM performed better than the other implementations but showed constant overfitting and steep changes in patterns for concentration predictions. Chi et al. [7] also implemented XGBoost for ground-level NO_2 concentration prediction in China, which showed a reliable method of forecasting and presented geographical and seasonal pollutant concentration variations. Al et al. [2] implemented ARIMA, SARIMA, NAR-NN and LSTM to investigate NO_2 predictions in the UAE for specific locations and its nearby monitoring stations and the whole area. The study showcased how accuracy can be affected by human activities and how different urban and rural areas might influence predictions both negatively and positively. Depending on the experiment, the optimal method varied, but machine learning implementations, LSTM and NAR-NN, presented the best overall results. Although methods for NO_2 forecast have shown promising results, research is mainly focused on predicting values for large areas and different terrains, while there is still a need for further focus on urban areas and how NO_2 concentration levels are affected by them considering weather and meteorological factors including microclimates.

The application of Long Short-Term Memory (LSTM) algorithms for air quality prediction has gained significant attention in recent years. Park et al. [14]

looked to forecast PM_{10} concentrations in Seoul, Korea, by leveraging LSTM algorithms. A moving average technique smoothed the data to mitigate the noisiness in the original dataset, which was then transformed into 30-day sequence sets for LSTM training. The study found that LSTM significantly outperformed linear regression models, improving RMSE rates by 500%. Despite the substantial gains, the study was limited to univariate PM_{10} data. Thus, there is a need for subsequent research incorporating additional variables like climate and meteorological data.

Tao et al. [18] implemented an ensemble method based on GRU, transformer and XGBoost for NO_2 prediction for the whole China region. Leveraging meteorological and spatial information, the study compared the ensemble model to each single architecture for three different time periods: 1-hr, 3-hr, and 24-hr. The ensemble model outperformed the other in every scenario and highlighted the high impact of external features such as O_3 and time factor for NO_2 forecasting accuracy.

Li et al. [11] developed a deep neural network model with multi-task learning (MTL-DBN-DNN) for air quality prediction. This model was pre-trained by a deep belief network (DBN), and its architecture was designed to allow for simultaneous predictions using shared information. Interestingly, each output unit in the model was connected only to a subset of the last hidden layer units in the DBN. This unique design feature helped to avoid the pitfalls commonly associated with fully connected networks and led to improved prediction accuracy. The model was tested on 12-h advance forecasts of $PM_{2.5}$, SO_2, and NO_2 concentrations, revealing its ability to capture commonalities and differences across multiple air quality variables. The proposed research looks at predicting NO_2 values for the city of Cambridge based on the data from an air quality monitoring station considering a large range of weather-related features. Recursive feature elimination (RFE) based feature elimination technique will be used to understand the key factors that contribute to predicting this pollutant accurately. Experiments are performed with multiple machine learning and deep learning models including ensemble models combining these.

3 Data and Methods

The air pollution dataset used in this research was sourced from the Air Quality England database [5], which stores air pollution data recorded by the Cambridge City Council. The dataset includes hourly measurements of NO_2 levels from a particular air quality monitoring station in the city from January 2020 to July 2023. The weather data was sourced from a weather data provider, Visual Crossing [19]. Weather information includes hourly data on 21 different variables, such as temperature, wind speed, wind direction, UV index and humidity. Sparse features like precipitation types, wind gust, risk levels and station details were removed from the dataset due to high number of null values. After pre-processing data, features with high missing values or low relevance to the analysis were removed. Missing values in the remaining features were replaced with the mean of each respective feature.

After cleaning and preprocessing the NO_2 and the Cambridge Weather datasets, the datasets were merged based on the 'datetime' index. Additional features, 'hour', 'day', and 'month', were also created and relevant features were selected using the Recursive Feature Elimination (RFE) technique. RFE identifies the attributes that contribute the most towards the accuracy of the model by recursively reducing the dimension of the dataset by evaluating each parameter. RFE has been implemented with Random Forest (RF): while the model performance is being evaluated, an RF regressor is generating metrics to quantify each parameter's significance and the lowest performing ones are eliminated from the data. RFE also improves model performance on NO_2 predictions by addressing interdependencies and multicollinearity. A total of 34 features were selected: hour, day, month and multiple weather variables were selected, as well as specific values from the weather dataset columns: conditions, such as rain, overcast, fog, snow, and wind. A complete list of all features selected can be found in Fig. 1. The importance of variables in relationship to each ML model can be found in the Appendix section.

```
['temp', 'feelslike', 'dew', 'humidity', 'precip', 'precipprob', 'snow', 'snowdepth', 'windspeed', 'winddir', '
sealevelpressure', 'cloudcover', 'visibility', 'solarradiation', 'solarenergy', 'uvindex', 'hour', 'day', 'mont
h', 'conditions_Overcast', 'conditions_Partially cloudy', 'conditions_Rain', 'conditions_Rain, Overcast', 'cond
itions_Rain, Partially cloudy', 'conditions_Snow, Rain, Overcast', 'conditions_Snow, Rain, Partially cloudy', '
icon_clear-night', 'icon_cloudy', 'icon_fog', 'icon_partly-cloudy-day', 'icon_partly-cloudy-night', 'icon_rain'
, 'icon_snow', 'icon_wind']
```

Fig. 1. RFE Feature Selected

The methodology is summarised in Fig. 2. The data was then divided into training and testing sets. As this is a time series dataset, the data sequence had to be preserved. Thus, the last week of hourly data served as the testing set, while the remaining data comprised the training set. Before training the model, the feature values in both the training and testing sets underwent scaling. A Min-Max scaling technique was applied to standardise the data. This step was required to ensure that the model did not give disproportionate importance to data with higher numerical value ranges over data with lower ranges. All models were trained and tested using this training and testing data. However, due to the need to create sequences for the LSTM model, sequences were generated from this training dataset specifically to train the LSTM model. The models were evaluated using RMSE and MAE metrics.

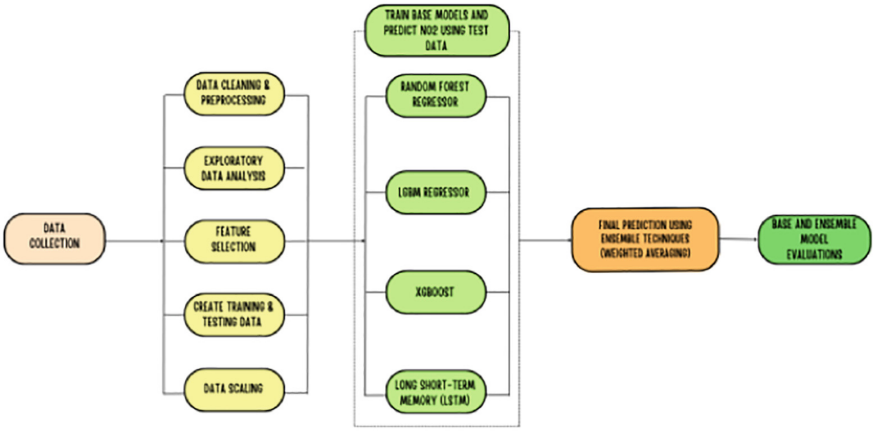


Fig. 2. Methodology Diagram

4 Results and Discussion

All models were evaluated individually based on their predicted outputs using (5-fold) cross-validated RMSE and MAE scores: the outputs included hourly predictions of NO_2 values for the entire last week of the dataset. The evaluation metrics for each model were compared to select the best machine-learning technique for predicting NO_2 concentration levels in Cambridge. Figure 3 and Fig. 4 summarise the results of each evaluation metric comparing the models to each other.

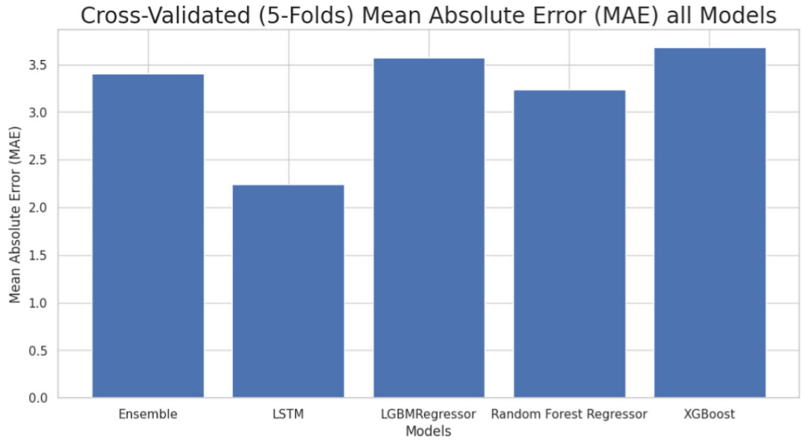


Fig. 3. MAE Scores

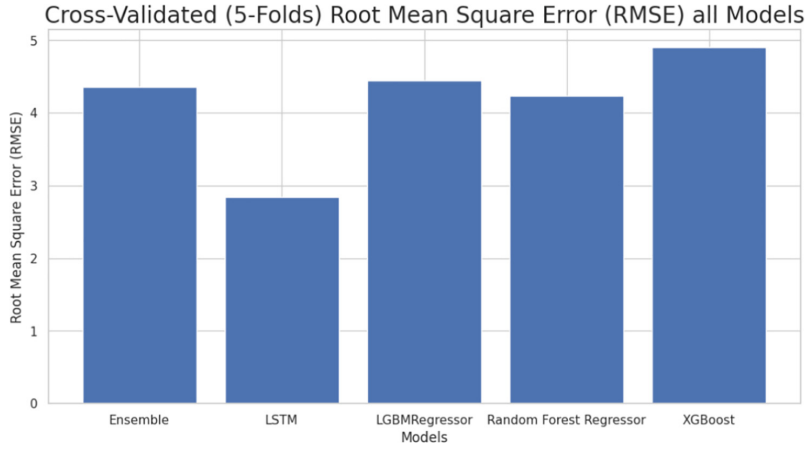


Fig. 4. RMSE Scores

Model performance was calculated using RMSE and MAE metrics. The LSTM model produced a better prediction of NO_2 concentrations (RMSE = 2.847, MAE = 2.24) than the Ensemble Model (RMSE = 4.359, MAE = 3.405), Light GBM Model (RMSE = 4.446, MAE = 3.570), Random Forest Model (RMSE = 4.233, MAE = 3.235), and XGBoost Model (RMSE = 4.902, MAE = 3.679). A sample prediction result is shown in Fig. 5. It can be observed that LSTM model most closely follows the ground truth pollutant values compared to other models.

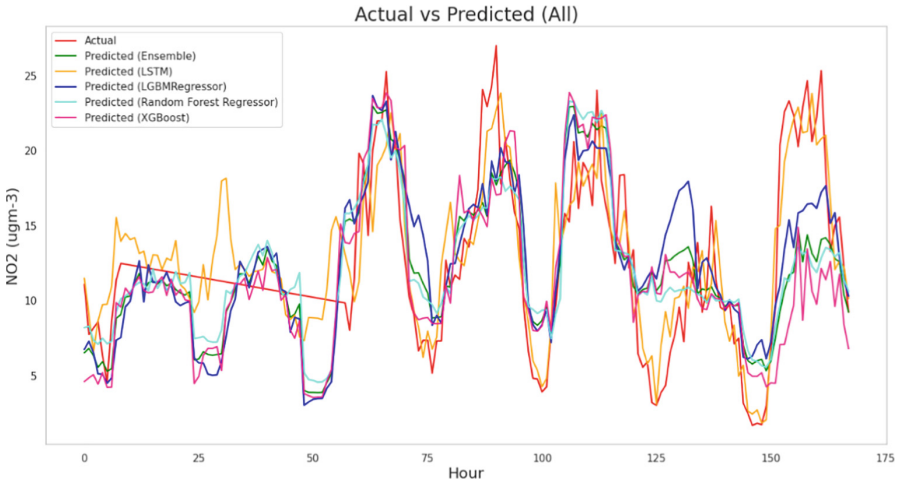


Fig. 5. NO2 Predictions Vs Actual (All Models)

The LSTM model outperformed other models, confirming the findings of [13], where LSTM was successfully applied to predict $PM_{2.5}$ concentration. LSTM outperformed the Random Forest model because of its intrinsic ability to understand long-term dependencies. It is essential to highlight that NO_2 concentration levels vary as time goes by, increasing complexity and thus leading to poor predictive performance from the Random Forest Model. For LSTM, however, this is where its distinctive ability to learn long-term dependencies gives better performance [16]. The accuracy of the LSTM network remained relatively stable, even as complexity increased. The LSTM model also outperformed both the XGBoost and the LightGBM models. Unlike the LightGBM model, which utilises feature variables from identical timeframes as the target for forecasting, LSTM uses feature variables within a window period for target variable prediction [6]. Thus, when significant changes occur in a short period, the LightGBM model may miss this, while the LSTM model captures this due to the window period, thus reducing prediction errors. Lastly, even the ensemble model comprised of the Random Forest, XGBoost, and LightGBM Models; could not outperform the LSTM model.

5 Conclusion and Further Work

This research focused on using machine-learning techniques to predict NO_2 concentration levels, specifically within the City of Cambridge, and found the LSTM model to have superior performance over other models. The ability of LSTM to understand long-term dependencies and utilising feature variables within a window period for target variable prediction makes this the best-performing model. The usage of climatic data and longer-term data within a small geographical region further evidences LSTM as the most effective model for NO_2 concentration prediction. The research also looks are feature elimination techniques to identify the most relevant features that could accurately model the pollutant. However, the LSTM model is resource-intensive and requires significant computing power for tasks involving a large dataset. As the next step, it would be useful to further reduce the complexity of modeling longitudinal data and look at possibilities of spatiotemporal modeling of the pollutants rather than just temporal modeling to maximise the effectiveness of the handful of air quality sensors in a city.

6 Appendix

(See Figs. 6, 7 and 8)

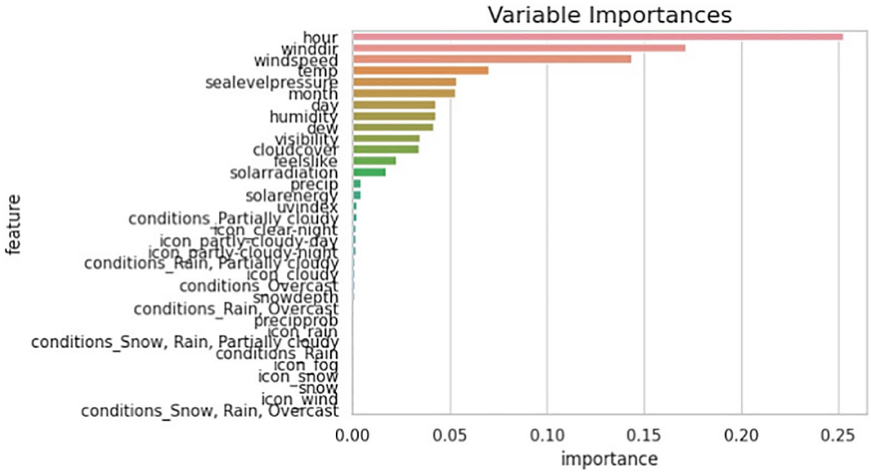


Fig. 6. RF Feature Importance

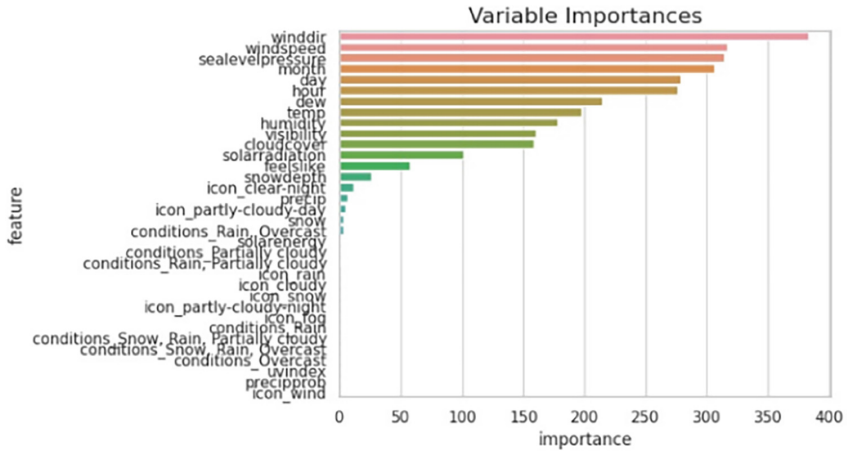


Fig. 7. LGBM Feature Importance

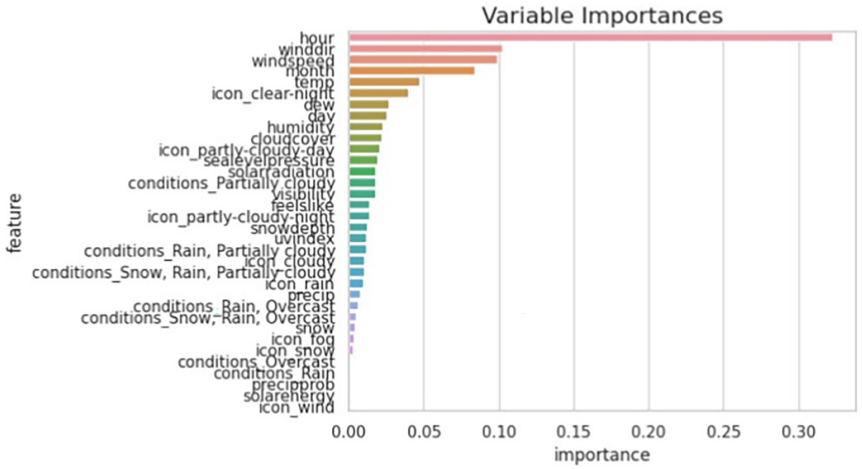


Fig. 8. XGBoost Feature Importance

References

1. Aditya, C., Deshmukh, C.R., Nayana, D., Vidyavastu, P.G.: Detection and prediction of air pollution using machine learning models. Int. J. Eng. Trends Technol. **59**(4), 204–207 (2018)
2. Al Yammahi, A., Aung, Z.: Forecasting the concentration of no2 using statistical and machine learning methods: a case study in the uae. Heliyon **9**(2) (2023)
3. Aladağ, E.: Forecasting of particulate matter with a hybrid arima model based on wavelet transformation and seasonal adjustment. Urban Climate **39**, 100930 (2021)
4. Alam, T., AlArjani, A.: Forecasting co 2 emissions in Saudi Arabia using artificial neural network, holt-winters exponential smoothing, and autoregressive integrated moving average models. In: 2021 International Conference on Technology and Policy in Energy and Electric Power (ICT-PEP), pp. 125–129. IEEE (2021)
5. AQE: Air pollution measurements (2023). https://www.cambridge.gov.uk/air-pollution-measurements
6. Bian, L., Qin, X., Zhang, C., Guo, P., Wu, H.: Application, interpretability and prediction of machine learning method combined with lstm and lightgbm-a case study for runoff simulation in an arid area. J. Hydrol. **625**, 130091 (2023)
7. Chi, Y., et al.: Machine learning-based estimation of ground-level no2 concentrations over china. Sci. Total Environ. **807**, 150721 (2022)
8. Galvão, S.L.J., et al.: Particulate matter forecasting using different deep neural network topologies and wavelets for feature augmentation. Atmosphere **13**(9), 1451 (2022)
9. Kamińska, J.A.: A random forest partition model for predicting no2 concentrations from traffic flow and meteorological conditions. Sci. Total Environ. **651**, 475–483 (2019)
10. Li, C., Li, Y., Bao, Y.: Research on air quality prediction based on machine learning. In: 2021 2nd International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI), pp. 77–81. IEEE (2021)

11. Li, J., Shao, X., Sun, R.: A dbn-based deep neural network model with multitask learning for online air quality prediction. J. Control Sci. Eng. **2019** (2019)
12. Liu, T.L., Flückiger, B., de Hoogh, K.: A comparison of statistical and machine-learning approaches for spatiotemporal modeling of nitrogen dioxide across switzerland. Atmos. Pollut. Res. **13**(12), 101611 (2022)
13. Ma, J., Li, Z., Cheng, J.C., Ding, Y., Lin, C., Xu, Z.: Air quality prediction at new stations using spatially transferred bi-directional long short-term memory network. Sci. Total Environ. **705**, 135771 (2020)
14. Park, J.H., Yoo, S.J., Kim, K.J., Gu, Y.H., Lee, K.H., Son, U.H.: Pm10 density forecast model using long short term memory. In: 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 576–581. IEEE (2017)
15. Pörtner, H.O., et al.: Ipbes-ipcc co-sponsored workshop report on biodiversity and climate change. IPBES IPCC **10** (2021)
16. Reuß, F., Greimeister-Pfeil, I., Vreugdenhil, M., Wagner, W.: Comparison of long short-term memory networks and random forest for sentinel-1 time series based large scale crop classification. Remote Sens. **13**(24), 5000 (2021)
17. Sharma, S.B., Jain, S., Khirwadkar, P., Kulkarni, S.: The effects of air pollution on the environment and human health. Indian J. Res. Pharmacy Biotechnol. **1**(3), 391–396 (2013)
18. Tao, C., et al.: Time-sensitive prediction of no2 concentration in china using an ensemble machine learning model from multi-source data. J. Environ. Sci. **137**, 30–40 (2024)
19. VisualCrossing: Weather data & api (2023). https://www.visualcrossing.com
20. Vohra, K., Vodonos, A., Schwartz, J., Marais, E.A., Sulprizio, M.P., Mickley, L.J.: Global mortality from outdoor fine particle pollution generated by fossil fuel combustion: results from geos-chem. Environ. Res. **195**, 110754 (2021)
21. Zamani Joharestani, M., Cao, C., Ni, X., Bashir, B., Talebiesfandarani, S.: Pm2. 5 prediction based on random forest, xgboost, and deep learning using multisource remote sensing data. Atmosphere **10**(7), 373 (2019)

Improving Agricultural Image Classification by Mining Images

Wei Zhou, Aoyang Liu, and Yongqiang Ma(✉)

National Key Laboratory of Human-Machine Hybrid Augmented Intelligence, National Engineering Research Center of Visual Information and Applications, Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, Shaanxi, China

musayq@xjtu.edu.cn

Abstract. The task of agricultural image classification has always been a popular topic in agricultural research. Both traditional and deep learning-based methods have emerged to address this task. However, as these methods have expanded, they have become more reliant on data and require additional external information to improve performance. In reality, agricultural images often have low quality and lack annotations, and it is challenging to obtain clear external prior knowledge and semantic information. Therefore, we aim to improve image classification using only the simplest agricultural image dataset, which consists of images and their corresponding class labels. By leveraging the information inherent in the images themselves, we seek to obtain prior knowledge and semantic information to enhance image classification performance, and we use Class Activation Mapping to illustrate the results and the improvement. Furthermore, we enhance the feature extraction process by utilizing it. We conducted experiments on four agriculture-related datasets, using Residual Neural Network as our baseline. The results show that our method achieves improvements in both Top-1 accuracy and Mean Average Precision metrics.

Keywords: Agricultural Image Classification · Semantic Information · Feature Extraction

1 Introduction

In the field of agriculture, image classification tasks are primarily used to assist farmers in identifying plants [1] and detecting diseases [2], enabling them to choose more suitable agricultural practices and improve overall agricultural production. There are various methods applied to agricultural image classification. Traditional approaches involve manually extracting image features and using them as input to classifiers. With the advent of Convolutional Neural Networks (CNNs), deep learning-based methods [3–6] have greatly advanced the development of agricultural image classification. The introduction of attention mechanisms, such as the Transformer-based [7] Vision Transformer (ViT) [8], has

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 226–237, 2024.
https://doi.org/10.1007/978-3-031-63215-0_17

further improved the performance in this domain. Additionally, the emergence of large-scale models and vision-language models [9,10] has expanded the scope of agricultural image classification beyond solely analyzing images, incorporating multiple modalities such as crop description texts and videos.

To further enhance classification accuracy, integrating prior knowledge [11] and domain-specific information into agricultural image classification tasks is a promising strategy. Prior knowledge can be leveraged from knowledge graphs [12,13], probability distributions [14], or languages [15,16]. Semantic information is often obtained through techniques like semantic segmentation [17–20] and object detection [21–23].

While these methods indeed improve the accuracy of image classification, they typically require high-quality, fine-grained annotated, and large-scale agricultural image datasets. Moreover, acquiring prior knowledge and semantic information related to agriculture often involves complex processes such as knowledge collection, annotation, and graph construction, which significantly increases the overall difficulty of the work. Another challenge is that the abundance of image data in agricultural production may not always meet the requirements for high-quality or sufficient additional prior or semantic information. This raises the question of whether it is feasible to improve image classification by utilizing the inherent information conveyed by agricultural images themselves [24], rather than solely relying on high-quality and abundant data.

Our approach leverages the inherent information present in agricultural images, obtained through traditional feature extraction techniques used in image classification tasks. And, we utilize the semantic relationship between these features and their corresponding labels to improve the effectiveness of image classification. By incorporating the raw image features as prior knowledge input for the network model, we benefit from the rich information contained within the images. Moreover, it is evident that the extracted features from the same image and their associated categories exhibit strong semantic correlations. This training methodology effectively captures and utilizes the semantic information embedded in the images. We will present experimental results using Residual Neural Network (ResNet) [5] as the baseline model on four agriculture-related datasets. Additionally, we will explore the application of Class Activation Mapping (CAM) [25] to identify the salient regions of the image that the model focuses on during the classification task, providing valuable insights into the results. Furthermore, we will conduct experiments to enhance the feature extraction step using CAM.

Our contributions can be summarized in three aspects:

- We emphasize the utilization of intrinsic image features for agricultural image classification tasks, without relying on external information. Additionally, we introduce explicit semantic information related to the images.
- We employ CAM to visualize the model's focus points using different methods, and we explore the integration of CAM to achieve an implicit attention effect.
- We conducted experiments on four datasets using the ResNet model.

2 Related Works

ResNet. ResNet was proposed by K.He et al. [5] at Microsoft Research in 2015. This network successfully addresses the issue of gradient vanishing during training of deep neural networks, enabling effective training of much deeper networks. The key idea behind ResNet is the incorporation of residual blocks (ResNet units), which allow the network to learn the residual of the previous network output rather than the entire output. This design enables each layer of the neural network to focus on learning the residual between the input and output, simplifying the learning process and improving training efficiency. In this paper, we utilize ResNet as the foundational network model.

Traditional Image Feature Extraction Methods. Traditional methods for image feature extraction are diverse and well-established. For color features, commonly used techniques include color histograms, color moments, and color aggregation vectors [26]. For texture analysis, methods like the Gray-Level Co-occurrence Matrix (GLCM) and Local Binary Patterns (LBP) are popular. Shape features often involve Hough transforms and Fourier shape descriptors. In this paper, we employ color moments, LBP, and Hough transforms to extract features.

Class Activation Map. CAM [25] is a tool used for visualizing CNNs. It applies Global Average Pooling (GAP) operation at the final output layer. For each feature map in the last convolutional layer, the average response over the entire image is obtained through GAP. These average responses are then multiplied with the class probabilities before the softmax layer, resulting in the contribution of each feature map towards the final prediction. By weighting and superimposing these contributions, CAM generates a heat map, allowing clear visualization for the image regions focused on by the network.

3 Method

3.1 Using Prior Knowledge from Images

Color, texture, and shape are the most informative features that capture the content in agricultural images. Therefore, we focus on extracting these three aspects of features from agricultural images.

Color Moments. Color moments provide a description of the global color distribution in an image. We collect the (R, G, B) information for the entire image by scanning it. The first, second, and third-order moments for each color channel can be calculated using the following formulas:

$$\mu_i = \frac{1}{N} \sum_{j=1}^{N} p_{i,j}, \tag{1}$$

$$\sigma_i = \left(\frac{1}{N} \sum_{j=1}^{N} \left(p_{i,j} - \mu_i \right)^2 \right)^{\frac{1}{2}}, \tag{2}$$

$$s_i = \left(\frac{1}{N} \sum_{j=1}^{N} \left(p_{i,j} - \mu_i \right)^3 \right)^{\frac{1}{3}}, \tag{3}$$

where μ, σ, and s represent the first-, second-, and third-order moments, respectively. $p_{i,j}$ represents the value of the color component i (R, G, or B) for the j-th pixel. The total number of pixels is denoted by N.

The color moment information of an image is represented as a 9-dimensional vector, and we normalize the first-order, second-order, and third-order moments according to the following method. To capture local variations, we divide the image into a 4×4 grid, resulting in 16 sections. For each section, we calculate the color moments and arrange them in a fixed order. This process generates a 144-dimensional vector (16 segments $\times$ 9 dimensions per segment), serving as the color feature.

$$\mu_n = \frac{\mu}{255}, \sigma_n = \frac{\sigma}{255 \times 255}, s_n = \frac{s}{s_{max}}, \tag{4}$$

where s_{max} represents the globally maximum third-order moment, which is obtained through pre-computation.

Local Binary Patterns. LBP are used to capture the texture features of images. We employ the original LBP method, which compares the grayscale value of each pixel with its eight neighboring pixels within a specified range, resulting in a descriptor for that pixel. This computation is performed for all pixels in the image, and the occurrences of the 256 different pixel descriptors are counted and plotted as a histogram. The horizontal axis of the histogram represents the 256 types of pixel descriptors, while the vertical axis records the frequency of each descriptor in the image. For each neighboring point, a pixel point p undergoes the following computation:

$$b = \begin{cases} 1, p_t - p \geq 0 \\ 0, p_t - p \leq 0, \end{cases} \tag{5}$$

where p_t is an adjacent pixel point around pixel point p, with a total number of 8. By using Eq. 5, eight values are obtained for point p. These values are then arranged in a fixed order to form an eight-bit binary number, which represents the descriptor of the pixel point p.

Now we have a 256-dimensional vector that records the frequency of each value. To normalize this vector, we apply the following formula:

$$\boldsymbol{l_n} = \frac{\boldsymbol{l}}{P}, \tag{6}$$

where $\boldsymbol{l_n}$ represents the normalized feature vector, $\boldsymbol{l}$ is the feature vector before normalization, and P represents the number of pixel points involved in the calculation.

Hough Transform. Hough Transform is a powerful technique used in image processing to detect and recognize geometric shapes. In our approach, we utilize the Hough Transform to extract circles from the image, capturing their coordinates and radii.

To create the shape feature, we randomly select 112 circles from the image and collect their radii. We randomly arrange these values of radii, concatenate them, and normalize the resulting vector. If the number of extractable circles is fewer than 112, we supplement the remaining positions with circles of radius 0 until we reach a total of 112 circles. To normalize the vector, we divide each element by the maximum radius allowed during the circle detection process.

Integration of Prior Knowledge. We concatenate the three feature vectors.

$$\boldsymbol{v} = cat\left(\boldsymbol{c_n} + \boldsymbol{l_n} + \boldsymbol{h_n}\right), \tag{7}$$

where $\boldsymbol{c_n}$, $\boldsymbol{l_n}$, $\boldsymbol{h_n}$ represent the color, texture, and shape feature vectors of the image, respectively. *cat* represents the operation of horizontally concatenating vectors. Figure 1 illustrates the process from feature extraction to vector $\boldsymbol{v}$. Then, we concatenate it with the output of the flatten layer in ResNet and feed the combined vector into the Fully Connected (FC) layer.

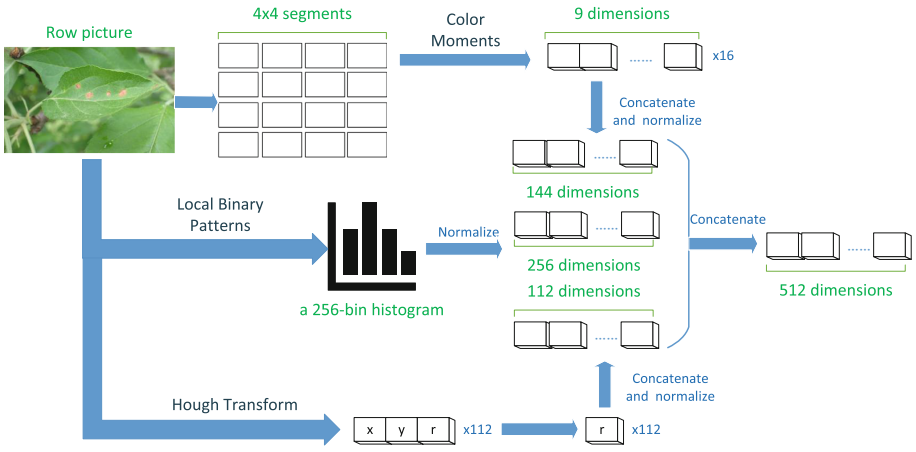


Fig. 1. The entire process of extracting prior feature vectors.

3.2 Using Semantic Correlation Information

During the neural network parameter learning process, the image labels can be seen as a form of description for the image information. There is a clear semantic correlation between the label information and the image feature information, as the labels often summarize the key components that the neural network needs to learn from the image. To establish semantic associations between the image feature vectors and the image labels, we use a 512-dimensional vector for each image, obtained by summing the dimensions of the color, texture, and shape features (144 + 256 + 112). To align the dimensions of this vector with the number of categories, we pass it through two FC layers. Subsequently, we calculate the cross-entropy loss by comparing the transformed vector with the predicted categories generated by the model.

$$L_{feature}\left(v_n, class\right) = -\log\left(\frac{\exp\left(v_n\left[class\right]\right)}{\sum_{i=0}^{N-1}\exp\left(v_n\left[i\right]\right)}\right), \tag{8}$$

where *class* refers to the category that the corresponding image of the vector belongs to, and N represents the total number of categories. The total loss function is modified to:

$$L_{total} = L_{model} + L_{feature}. \tag{9}$$

At this point, the image features have established a semantic connection with the image labels, which will contribute to the classification of images.

3.3 Using CAM to Demonstrate and Improve Agricultural Image Classification

CAM allows us to visually identify the regions of interest in an image based on the predictions of the model. To leverage this capability, we utilized a pretrained model along with CAM to identify these attentive regions. We then extract feature vectors specifically from these regions, use them for secondary training, and estimate the effectiveness of this method.

4 Experiments

4.1 Datasets and Training Details

Datasets. We utilized four datasets in our study. The datasets and implementation details are as follows:

- **Appleleaf9** [27]. Appleleaf9 dataset comprises approximately 15,000 images, covering eight different patterns of apple leaf diseases as well as a healthy state.

- **Tomatoleaf-A** and **Tomatoleaf-B**. The "Tomato Disease Multiple Sources" [28] collection on Kaggle comprises over 30,000 images of tomato leaf pathologies, encompassing a total of 11 distinct categories. We obtained two subsets from this extensive collection, namely Tomatoleaf-A and Tomatoleaf-B, with approximately 5,000 images and 10,000 images respectively.
- **Plantvillage**. Plantvillage, which is part of the PlantVillage dataset [29], consisting of over 20,000 images, covers 12 types of crop leaf diseases and healthy states, encompassing crops such as peppers, potatoes, and tomatoes.

We split each dataset into two parts, 90 percent for training and the remaining 10 percent for validation.

Training Details. We used ResNet50 and ResNet101 as the baseline models for our experiments. The Stochastic Gradient Descent (SDG) optimizer was selected with the following parameters: learning rate of 0.001, momentum of 0.9, and weight decay of 0.0005. During training, we utilized a batch size of 8, while for validation, the batch size was set to 4.

4.2 Results with Prior Knowledge from Images

By adjusting the input length of the FC layer in ResNet50 and ResNet101, we integrated prior image features into the model during training. The training

Table 1. Mean Average Precision (mAP) and Top-1 Accuracy (Top-1 acc) without/with prior knowledge on four datasets.

Datasets	Methods	mAP	Top-1 acc
Tomatoleaf-A	ResNet50	92.2	92.0
	ResNet50 + PK	92.5	92.6
	ResNet101	92.2	92.6
	ResNet101 + PK	94.5	94.6
Tomatoleaf-B	ResNet50	94.0	93.9
	ResNet50 + PK	94.7	94.5
	ResNet101	95.0	94.9
	ResNet101 + PK	95.7	95.7
Appleleaf	ResNet50	96.0	97.0
	ResNet50 + PK	94.1	97.3
	ResNet101	96.6	97.8
	ResNet101 + PK	96.7	97.9
Plantvillage	ResNet50	99.4	99.3
	ResNet50 + PK	98.6	99.1
	ResNet101	99.5	99.5
	ResNet101 + PK	99.6	99.7

results for the four datasets under this configuration are presented in Table 1. In the upcoming table, we use "PK" as an abbreviation for "prior knowledge".

We noticed that incorporating prior image features leads to improvements in both Top-1 acc and mAP, which demonstrate that our method is useful. However, it is important to note that as the dataset size increases, the baseline results of ResNet50 and ResNet101 also improve accordingly. In some cases, we observed a plateau or even a decline in performance. This could be attributed to overfitting when the dataset is already of high quality and sufficient quantity.

4.3 Results with Semantic Information

We employed two FC layers to reduce the dimensionality of the 512-dimensional feature vector. The weights of these two FC layers were initialized using the Xavier uniform distribution. For the optimization process, we employed the Adam optimizer with a learning rate set to 0.005. The training results for the four datasets after incorporating semantic information into the model are presented in Table 2. As evident from the table, our method of incorporating semantic information outperforms the method that solely relies on image features. In the upcoming table, we use "SI" as an abbreviation for "semantic information".

Table 2. Mean Average Precision (mAP) and Top-1 Accuracy (Top-1 acc) with prior knowledge and semantic information on four datasets.

Datasets	Methods	mAP	Top-1 acc
Tomatoleaf-A	ResNet50 + PK + SI	94.1	93.8
	ResNet101 + PK + SI	94.8	95.4
Tomatoleaf-B	ResNet50 + PK + SI	94.6	94.8
	ResNet101 + PK + SI	96.4	96.3
Appleleaf	ResNet50 + PK + SI	94.8	97.5
	ResNet101 + PK + SI	97.3	98.0
Plantvillage	ResNet50 + PK + SI	99.3	99.4
	ResNet101 + PK + SI	99.8	99.8

4.4 Using CAM to Display and Optimize

Figure 2 showcases the heat maps generated by the CAM for different methods with ResNet50 in some examples. Notably, when incorporating prior knowledge and semantic correlation information, it becomes evident that the model pays more attention to the diseased areas of the leaves. This observation highlights an important point: as the model receives more informative information, it naturally directs its focus towards the relevant regions. This aligns with the fundamental

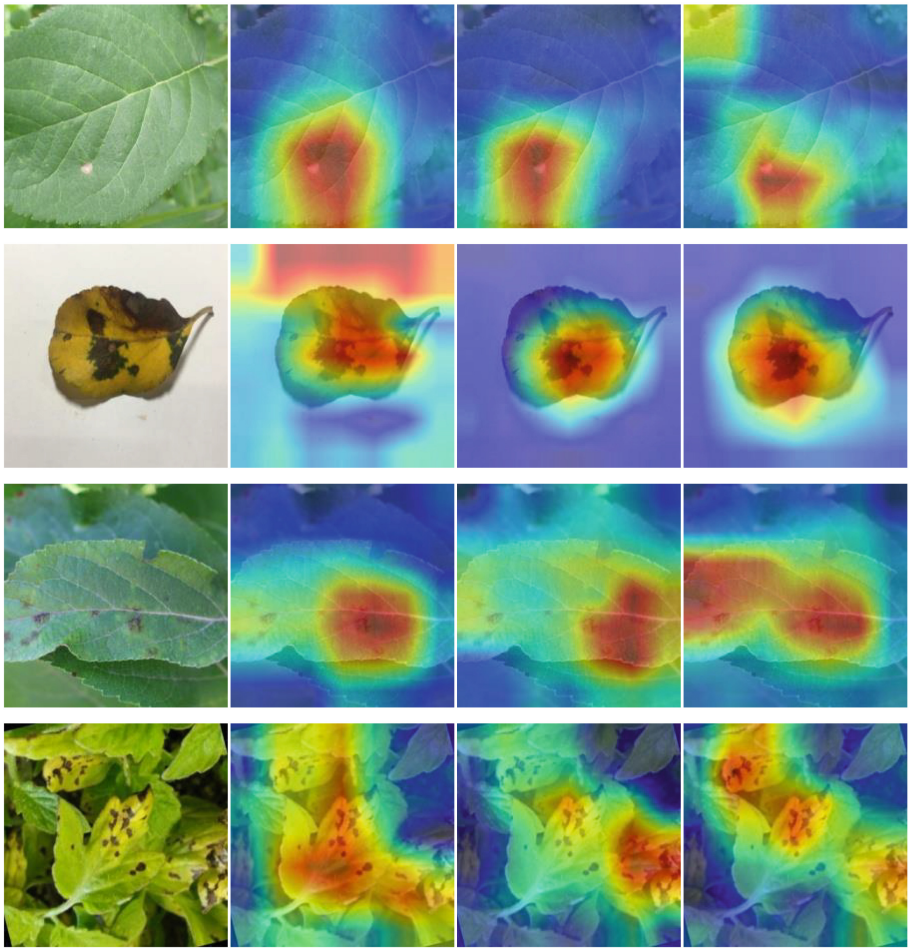


Fig. 2. Some examples of heat maps generated by CAM in different methods. For each line, the figures represent row image, the heat maps using CAM after training with ResNet50, ResNet50+PK, and ResNet50+PK+SI, respectively.

concept of the attention mechanism, which emphasizes the significance of guiding the model's attention to the most informative parts of the input.

Figure 3 showcases regions of interest extracted using CAM. Table 3 presents the results obtained by incorporating CAM-guided feature extraction into our training pipeline. By using CAM, we guide the model to focus on the diseased areas of the leaves. We enable the model to capture more accurate features and improve the classification performance. The results indicates the potential benefits of this method.

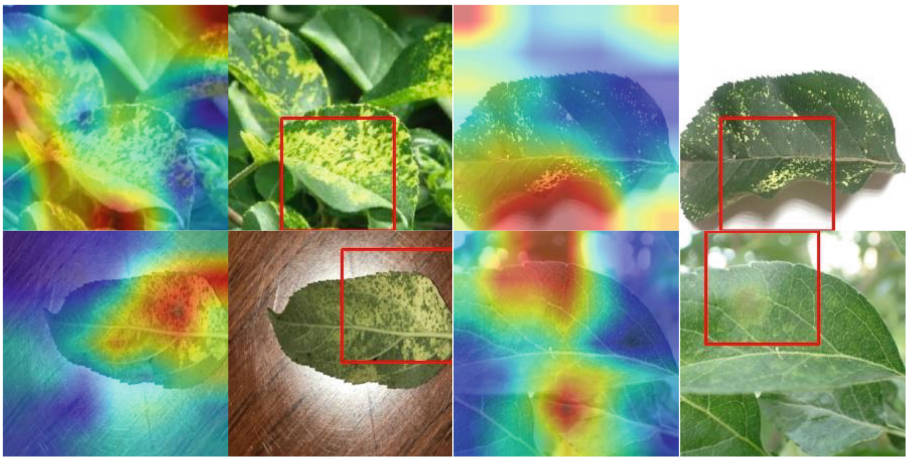


Fig. 3. The results of attentive regions extracted by pretrained models using CAM. Among them, the left side of each set of figures represents the images generated after CAM processing, while the right side showcases the attentive regions.

Table 3. Mean Average Precision (mAP) and Top-1 Accuracy (Top-1 acc) with prior knowledge using CAM on four datasets.

Datasets	Methods	mAP	Top-1 acc
Tomatoleaf-A	ResNet50 + CAM + PK	92.3	92.6
	ResNet101 + CAM + PK	95.3	95.5
Tomatoleaf-B	ResNet50 + CAM + PK	94.7	94.4
	ResNet101 + CAM + PK	95.7	96.2
Appleleaf	ResNet50 + CAM + PK	95.7	97.3
	ResNet101 + CAM + PK	96.3	97.9
Plantvillage	ResNet50 + CAM + PK	99.3	99.5
	ResNet101 + CAM + PK	99.5	99.5

5 Conclusion

In this paper, we propose a novel method to enhance agricultural image classification by leveraging the inherent information of image data as prior knowledge. By incorporating prior knowledge extracted from the raw images and semantic information obtained from the classification labels into the task, we have improved the performance of agricultural image classification, particularly when performing on the datasets without fine-grained annotations or external information. Furthermore, we introduce CAM to explain the effectiveness of our approach, and utilize CAM to extract attentive regions for prior knowledge, instead of using the global features of the entire image. Our methods have demonstrated promising results on four agricultural-related datasets. As part of future work,

we plan to investigate the impact and improvement of fragmented and blurred information on agricultural image classification tasks, which will provide a better exploration and a further understanding on the concept of inherent image information.

Acknowledgments. This work was supported by the STI2030-Major Projects (NO. 2021ZD0113604) and National Natural Science Foundation of China (NO. 62088102).

References

1. Batchuluun, G., Hong, J.S., Wahid, A., Park, K.R.: Plant image classification with nonlinear motion deblurring based on deep learning. Mathematics **11**(18), 4011 (2023)
2. Peng, H., Xu, H., Zhou, Z.: Crop pest image classification based on improved densely connected convolutional network. Front. Plant Sci. **14**, 1133060 (2023)
3. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)
4. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
6. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)
7. Vaswani, A., et al.: Attention is all you need. Advances in neural information processing systems **30** (2017)
8. Dosovitskiy, A., et al.: An image is worth 16×16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
9. Gan, Z., Li, L., Li, C., Wang, L., Liu, Z., Gao, J., et al.: Vision-language pre-training: Basics, recent advances, and future trends. Found. Trends Comput. Graph. Vis. **14**(3–4), 163–352 (2022)
10. Li, F., et al.: Vision-language intelligence: tasks, representation learning, and large models. arXiv preprint arXiv:2203.01922 (2022)
11. Roychowdhury, S., Diligenti, M., Gori, M.: Image classification using deep learning and prior knowledge. In: Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence (2018)
12. Zhang, D., et al.: Knowledge graph-based image classification refinement. IEEE Access **7**, 57678–57690 (2019)
13. Marino, K., Salakhutdinov, R., Gupta, A.: The more you know: using knowledge graphs for image classification. arXiv preprint arXiv:1612.04844 (2016)
14. McCann, S., Lowe, D.G.: Local naive bayes nearest neighbor for image classification. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3650–3656. IEEE (2012)
15. He, X., Peng, Y.: Fine-grained image classification via combining vision and language. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5994–6002 (2017)

16. Zhang, Y., Zhang, M., Li, W., Wang, S., Tao, R.: Language-aware domain generalization network for cross-scene hyperspectral image classification. IEEE Trans. Geosci. Remote Sens. **61**, 1–12 (2023)
17. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
18. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, pp. 234–241. Springer (2015)
19. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(12), 2481–2495 (2017)
20. Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 833–851. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_49
21. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
22. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)
23. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: SSD: single shot MultiBox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
24. Kabbai, L., Abdellaoui, M., Douik, A.: Image classification by combining local and global features. Vis. Comput. **35**, 679–693 (2019)
25. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2921–2929 (2016)
26. Tian, D.P.: A review on image feature extraction and representation techniques. Int. J. Multimed. Ubiquitous Eng. **8**(4), 385–396 (2013)
27. Yang, Q., Duan, S., Wang, L.: Efficient identification of apple leaf diseases in the wild using convolutional neural networks. Agronomy **12**(11), 2784 (2022). https://doi.org/10.3390/agronomy12112784
28. Khan, Q.: Tomato Disease Multiple Sources. https://www.kaggle.com/datasets/cookiefinder/tomato-disease-multiple-sources
29. Ali, A.: PlantVillage dataset. https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset/data

Learning-Based Short-Term Energy Consumption Forecasting

Hatem Haddad(✉), Feres Jerbi, and Issam Smaali

Wattnow, La Goulette, Tunisia
{hatem.haddad,feres.jerbi,issam}@wattnow.io
https://wattnow.io

Abstract. Development of reliable methods is essential to understand building energy consumption. Traditional statistical models showed drawbacks to express non-linear predictions. The recent artificial intelligence methods are more suitable to study the non-linear correlation between the consumed data, meteorological data, and other features. To address these challenges, we evaluate and compare the performances of ten learning-based models on four energy consumption datasets. The proposed framework includes four preprocessing steps namely, outliers and missing data processing, resampling processing, data normalization and features reduction. The results revealed the importance of the preprocessing steps having a high impact on the forecast performances. In addition, finding results showed that performances drop when resampling the original data values and performances increase when reducing the features by applying Pearson Correlation Coefficient. Based on four evaluation metrics (MAE, MAPE, R-Squared and Pbias), forecast results revealed that the applied models achieved high forecasting performances. Moreover, Machine learning models achieved slightly better performances, especially ensemble models such as ERTR and XGBOOST, outperforming Deep learning models and Hybrid models.

Keywords: Energy consumption · Artificial intelligence · Short-term forecasting · Time series · Learning-based models

1 Introduction

The ongoing worldwide expansion of smart metering infrastructure development has created opportunities for forecasting energy consumption, improving the economy and the stability of the power system operation [1]. Generally speaking, traditional statistical models, built by simple regression functions, are not effective to express non-linear predictions. However, the recent artificial intelligence methods can obtain higher prediction performances in most real-world applications and have been widely applied to the prediction of building energy consumption [2]. Indeed, artificial intelligence algorithms are more suitable to study the non-linear correlation between the consumed data, meteorological data, and

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 238–251, 2024.
https://doi.org/10.1007/978-3-031-63215-0_18

other features such as the working hours and business days. They have advantages at expressing complex non-linear relations creating new learning-based models for demand-side management with an increased level of automation to reach reliable energy consumption forecasting.

Based on the time scale of the prediction, energy consumption forecasting can be divided into four categories: long-term (a year or more), medium-term (between one week and one year), short-term (from one hour to one week), and very short-term prediction (from a few minutes to less than one hour) [1]. In this paper, we evaluate short-term (one Week-ahead) data-driven models to predict energy consumption for buildings based on historical consumption data and meteorological data. In summary, the novelty of our work is presented below in the consequent points.

- Rather than analyzing energy consumption patterns through visual analysis techniques as described by [3], we investigate the forecasting of energy consumption by training and deploying various Machine Learning (ML) models, Deep Learning (DL) models, and Hybrid models, avoiding traditional statistical models that rely on simple regression functions, which often overlook external factors and do not explicitly account for lag as a crucial element.
- In real applications, outlier values are usually caused by power outage or a hardware/software failure of the measurement device. These data may lead to a biased analysis and can reduce the accuracy of the learning-based models. For this reason, we evaluate how the data preprocessing should handle outliers (including anomalous data) and missing values before applying the forecasting models.
- Data may be gathered at different frequency (e.g. 1-min frequency or 10-min frequency) leading to a significant difference in time steps. We assess how data resampling affects the prediction of energy consumption.
- Historical data and external data (e.g. meteorological data) might include a large number of features. We evaluate how feature reduction impacts forecasting energy consumption.

In this regard, this research highly contributes to the knowledge gap and undertakes a longitudinal assessment for determining the best Learning-based models, the most suitable preprocessing steps and the useful features to forecast short-term energy consumption.

2 Related Works

Earlier attempts to predict energy consumption were based on statistical models [4] such as: the Autoregressive Moving Average (ARMA), the Autoregressive Integrated Moving Average (ARIMA), and the seasonal model SARIMA. Authors in [5] conducted empirical studies and showed that deep learning-based algorithms, such as the Long Short-Term Memory (LSTM), outperformed the ARIMA model. Indeed, results showed that LSTM-based algorithm improved the prediction by 85% on average compared to ARIMA. This finding was explained by the fact that LSTM uses an iterative optimization algorithm with the goal

of finding the best results by tuning model parameters with respect to the gradient descent. In addition, Bidirectional Long Short-Term Memory (BiLSTM) model provided better predictions compared against ARIMA and LSTM [6]. More recently, applied to the energy consumption prediction on a daily basis, LSTM was found to be more prominent in comparison to ARIMA and SARIMA [7]. In [8], authors conducted a comparison of the ARIMA forecasting algorithms with ML models and DL models on different areas of time series analysis and forecasting. Results showed the superiority of ML models and DL models except in cases of small datasets characterized by a limited range of values or a limited time-span. This can be explained by the inability of ARIMA models to capture the non-linear relationships between the energy consumption and other factors that influence the consumption, such as meteorological data and occupancy [9]. Therefore, time series models variations and extensions (ARIMA, VARIMA, SARIMA, ARFIMA, Prophet, etc.) are more relevant for medium-term and long-term forecasting than short-term forecasting [10]. For these reasons, statistical models are not considered in our experiments and we focus on the Off-the-shelf Learning-based models detailed in Sect. 6.

Authors in [11] introduced a residential short-term load forecasting framework for a 69 general individual families households over three months from SGSC Australia's first commercial-scale smart grid project dataset. The proposed framework was based on the LSTM model and a conservative empirical optimisation based predictor. Authors compared their framework against Empirical Approaches (Empirical Means and Empirical MAPE Minimisation), the conventional backpropagation neural network, k-nearest neighbors regression, extreme learning machine and a hybrid forecasting framework. Authors showed that LSTM can capture the subtle temporal consumption pattern persisting in a single-meter load profile and produces better forecasts than an empirical predictor. Authors concluded that forecasting approaches which are successful for grid load forecasting struggle in the single-meter load forecasting problems.

In [12], stacked autoencoders model was combined with the extreme learning machine model on one retail building. This Hybrid model outperformed propagation neural network, support vector regression, the generalized radial basis function neural network and multiple linear regression models. Similarly, [13] combined the LSTM model with the stationary wavelet transform technique to address energy consumption forecasting problem of individual households. The proposed model outperformed support vector regression, LSTM model and convolutional neural network combining long short term memory (CNN-LSTM). The experiments were conducted on five different family houses in London from the UK-DALE dataset. In [4], authors proposed a CNN-LSTM neural network to predict residential energy consumption. In order to generate predicted electrical energy consumption in a fully connected hierarchy based on CNN-LSTM, the spatial characteristics of a multivariate time series variable are extracted from the convolution and pooling layers of the CNN layer and passed to the LSTM layer with the noise removed to model the irregular time information using the transmitted spatial features. The proposed CNN-LSTM model outperformed LSTM, Gated Recurrent Units (GRU), BiLSTM and Attention LSTM.

In [14], authors focused on the importance of feature selection for predicting peak demand and short term forecasts. LSTM and Support Vector Machine (SVM) were applied to 6 years energy consumption data of a commercial building (220,336 data points). Authors applied different features including pressure, humidity, air temperature and CO_2. Outliers and missing data were filled with the average of the energy load at the same time point of the same day type (working days and non-working days). Pearson and Spearman correlation coefficients are used to evaluate the correlation coefficient between the historical load and the output variable. Results showed that correlation coefficients between the electrical load and external factors are very low and the historical load is highly correlated with the output. Consequently, only historical load data was used as input features. Authors compared the LSTM performance against SVM performances on one hour ahead load forecasting, on one-day ahead peak and valley load forecasting. Authors concluded that LSTM is better at handling complex and unstable data based on sufficient training data. SVM is suitable for load forecasting of small-scale datasets. Authors in [15] proposed a CNN model combining long short-term memory autoencoder (LSTM-AE) model for future energy prediction in residential and commercial buildings. Applied to a household electric power consumption dataset and a commercial building data, the proposed model outperformed CNN, LSTM, CNN-LSTM and LSTM-AE. Authors in [16] presented deep recurrent neural networks (DRNN) to forecast industrial electric usage for the next 24 h based on one year data with 1-hour sampling frequency. GRU achieved the best performance outperforming Autoregressive prediction, Multi-Layer Perceptron, RNN, LSTM and GRU-LSTM models. Authors highlighted that increasing the number of hidden layers and neurons in each layer can negatively impact the performance of the DL algorithms.

The aforementioned papers did not mention a direct comparison of the study cases because of the numerous variables influencing their performances. Indeed, the cited models are implemented for different locations, in different time periods, with different dataset variables and with different applied evaluation metrics making comparison difficult.

3 Datasets Description

In order to evaluate forecasting energy consumption models, we built the WATT 2.0 dataset (detailed in Sect. 3.2). In addition, two publicly available datasets or datasets chunks are also considered in this study: AMPds [18] and ASHRAE [19]. Datasets description are shown in Table 1 including the primary use, the period and the sampling frequency. In addition, each dataset used features and characteristics (the number of rows, the number of outliers and the number of rows with missing values) are also specified in Table 2.

3.1 AMPds

AMPds dataset [18] includes measurements of electricity, water, and natural gas meters taken at one-minute intervals, resulting in a total of 1,051,200 read-

Table 1. Datasets description.

Dataset	Primary use	Period	Sampling frequency
AMPds	House	60 Days	10 min
WATT 2.0	Supermarket	1 year	1 h
ASHRAE B-164	Warehouse	1 year	1 h
ASHRAE B-276	Office	1 year	1 h
ASHRAE B-346	Entertainment/public assembly	1 year	1 h
ASHRAE B-1081	Manufacturing/industrial	1 year	1 h

Table 2. Datasets characteristics ans used features: timestamp (t), active power (P), humidity (H), temperature (T), apparent temperature (Ta), sea level pressure (Pr), rain (R), cloud cover (Cl), wind speed (Ws), wind direction (Wd), dew point (Dp), anomaly (A), season (S), weekend (We), business hours (Bh).

Dataset	Features	#Rows	#Outliers	#Missing rows
AMPds	t, P, A, S, We	8352	1043	0
WATT 2.0	t, P, H, T, Ta, S, Bh, Ws, We	8760	342	74
ASHRAE B-164	t, P, T, Dp, R, Pr, Wd, Ws, We	8784	72	0
ASHRAE B-276	t, P, T, Dp, R, Pr, Wd, Ws, We	8784	14	0
ASHRAE B-346	t, P, T, Dp, R, Pr, Wd, Ws, We	8784	1	0
ASHRAE B-1081	t, P, T, Dp, R, Pr, Wd, Ws, We	8784	0	0

ings per meter over a period of two years. Authors in [33] downsampled one house electricity data to 10 min and manually annotated outliers. We used a data Chunk of 8352 consecutive labeled rows including 1043 outliers in order to evaluate outliers processing (Sect. 6.1) and resampling processing (Sect. 6.2).

3.2 WATT 2.0

Energy data is collected by smart meters installed in a Food sales and service building. The collected dataset encompasses meteorological data and a year-long time frame taken at one-hour frequency resulting in a total of 8760 rows including 74 missing data rows.

3.3 ASHRAE

The ASHRAE dataset, a cornerstone for advancing energy prediction research, encapsulates a diverse range of hourly energy measurements from 16 global sites having different primary use: Education, Office, Parking, Lodging/residential, Entertainment/public assembly, Public services, Manufacturing/industrial, Services, Healthcare, Food sales and service, Religious worship and Other [19]. In our experiments, we only used ASHRAE data collected in 2016, a leap year, including 8784 rows. We conducted extensive experiments on all ASHRAE buildings categories but due to space constraints, we show, in this paper, energy forecasting

performances on four buildings categories: Warehouse (referred to as ASHRAE B-164 from now on), Office building (referred to as ASHRAE B-276 from now on), Entertainment/public assembly site (referred to as ASHRAE B-346 from now on), and Manufacturing/industrial building (referred to as ASHRAE B-1081 from now on).

4 Proposed Framework

In this section, we describe the proposed framework as shown in Fig. 1 that includes four processing components, namely, outliers and missing data processing, resampling processing, data normalization and features reduction.

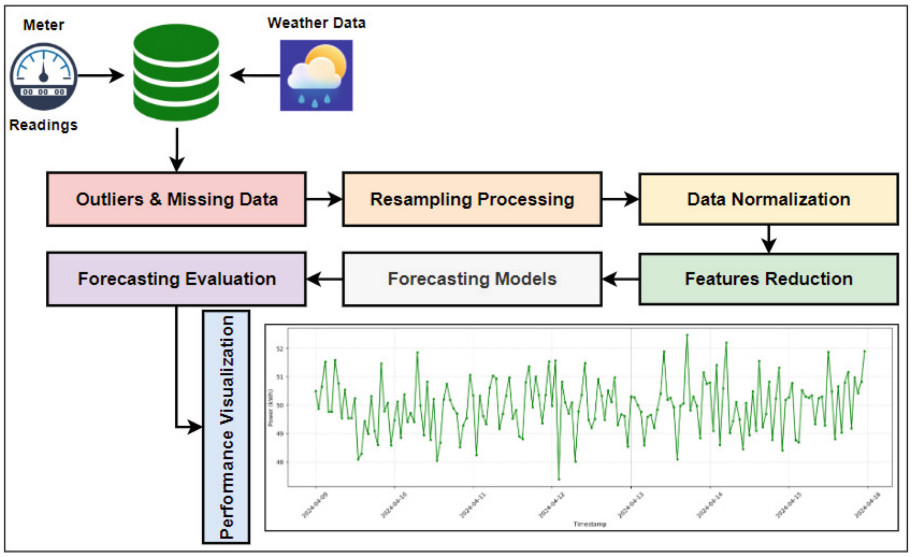


Fig. 1. Learning-based framework for short term energy consumption forecasting.

4.1 Outliers and Missing Data Processing

Monitoring energy consumption may result in the occurrence of missing values or outliers, usually caused by power outages or malfunctions in the measurement device's hardware or software. Instead of dropping rows containing outliers or missing values, a standard procedure described in [20] has been used to identify outliers in the datasets. Then, a linear interpolation method was applied to replace outliers and to fill missing data values [21].

4.2 Resampling Processing

Typically, training data is collected from the input time series by sliding a window along the time axis e.g., 8760 time steps of hourly data to describe a year.

Nevertheless, collected data might be taken at less than one hour frequency (e.g. 1-min frequency or 10-min frequency) resulting in very large time steps. In Sect. 6.2, we evaluate the impact of temporal data frequency on the forecasting models performances. For this reason, we compare the forecasting performances of a single future value on the AMPds dataset using 10-min frequency and 1-h frequency. We used the approach of averaging the six original 10-min frequency energy values because it achieved better performance than the sum approach.

4.3 Data Normalization

Learning-based models are sensitive to the scale of input data. Since the value range of the raw data varies, the value range of all features needs to be normalized so that each feature contributes approximately proportionally to the forecasting model. In addition, the gradient descent converges much faster with feature scaling than without. For this reason, we applied the following linear transformation based on min-max normalization, constraining its range between $[0, 1]$:

$$x_n = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{1}$$

where x denotes the original data and x_n denotes the normalized data. $x_{\max}$ and $x_{\min}$ represent the maximum and minimum values in the dataset, respectively. We also applied the following denormalization equation:

$$\hat{x} = (x_{\max} - x_{\min}) \cdot x_n + x_{\min} \tag{2}$$

where $\hat{x}$ represents the denormalized data, and x stands for the normalized data.

4.4 Features Reduction

Features that are either irrelevant (uninformative features) to the forecasting problem or redundant (increasing the dimensions) might be included as inputs without providing any additional forecasting benefit and increasing the models training time. To quantify the redundancy and the correlation of inputs, we conducted a correlation analysis [22] based on Pearson Correlation Coefficient (PCC) commonly used for linear variable selection. While most researchers would probably agree that a coefficient of <0.3 indicates "Weak correlation" and "Negligible correlation" [22], the cutoff points are arbitrary and data dependent [23]. Therefore, we conducted, for each dataset, an empirical study to determine the PCC cutoff point that leads to the best forecasting performances.

5 Evaluation Metrics

The used evaluation metrics [17] and the reason behind their selection are detailed in this section.

5.1 Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) indicates the average difference between the actual and predicted values. The formula for computing MAE is as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3}$$

where, $\hat{y}_i$ stands for the predicted value, y_i represents the actual value, and N refers to the number of prediction points or samples. Unlike other metrics, MAE is less sensitive to outliers, providing a better representation of the actual predictive errors. However, MAE does not indicate the errors magnitudes.

5.2 Mean Absolute Percentage Error (MAPE)

The Mean Absolute Percentage Error (MAPE) measures the error percentage relative to the actual value. The formula for computing MAPE is as follows:

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \tag{4}$$

where, $\hat{y}_i$ denotes the predicted value, y_i signifies the actual value, and N represents the number of prediction points or samples.

5.3 R-Squared (R^2)

R-squared (R^2), known as the coefficient of determination, measures the degree of correlation between predicted and actual values. The formula for computing R^2 is as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}, \qquad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{5}$$

where, y represents the average of the actual values, $\hat{y}_i$ refers to the predicted value, y_i stands for the actual value, and N denotes the number of prediction points or samples. An R^2 value approaching one indicates a high level of accuracy in forecasting but it can also be negative, highlighting a model that performs worse than a purely arbitrary prediction.

5.4 Pbias

Pbias measures the average tendency of predicted values to be larger or smaller than actual values [24]. It assess the outcome of a single value (the optimal value is zero) to determine underestimation bias (Positive values) or overestimation bias (Negative values). The formula for computing Pbias is as follows:

$$PBIAS = \frac{1}{n}\sum_{i=1}^{n}\frac{obs_i - sim_i}{obs_i} \times 100\% \tag{6}$$

where, obs_i represents the actual values and sim_i stands for predicted values.

6 Experimental Results and Discussion

Extensive experiments were conducted on 10 Off-the-shelf ML models, DL models and Hybrid models:

- ML models: Extremely Randomized Trees (ERTR) [25], eXtreme Gradient Boosting (XGBOOST) [26] and Light Gradient-Boosting Machine (LightGBM) [27].
- DL models: Artificial neural network (ANN) [28], CNN [29], GRU [30], LSTM [31] and BiLSTM [32].
- Hybrid models: we experimented with different hybrid models but due to space constraints, we present the performances of the hybrid CNN-LSTM model (referred to as Hybrid-M1 from now on) and the hybrid LSTM-Attention model (referred to as Hybrid-M2 from now on).

The motivation behind the models selection is that some models achieved the best state of the art results and the others achieved high performances on similar forecasting applications.

6.1 Outliers and Missing Data Evaluation

The AMPds dataset is inherently a supervised dataset for anomaly detection [33], featuring a column labeled "anomaly" with values of 0 indicating normal instances and 1 indicating outliers instances. We use this manual annotation in our experiments to study the impact of preprocessing outliers on the energy forecasting performances.

Table 3. AMPds Outliers and resampling processing evaluation.

Model	Case 1		Case 2		Case 3	
	MAE	Pbias (%)	MAE	Pbias (%)	MAE	Pbias (%)
ERTR	191.09	−36.79	113.20	−21.79	461.87	−85.50
XGBOOST	124.25	−23.92	42.26	−8.13	395.25	−73.17
LightGBM	86.06	−6.57	46.89	−9.02	491.62	-91.01
ANN	139.94	−26.94	9.79	−1.88	246.71	−45.67
CNN	107.12	−20.62	11.49	−2.21	216.99	−40.16
LSTM	24.73	4.76	**6.24**	**-1.20**	372.43	−68.94
BiLSTM	**11.17**	**−2.15**	8.38	−1.61	259.17	−47.97
GRU	28.47	−5.48	9.77	1.88	157.12	−29.08
Hybrid-M1	22.38	4.31	10.72	−2.06	**135.20**	**−25.02**
Hybrid-M2	118.48	−22.81	66.88	−12.87	427.79	−79.19

Table 3 shows results achieved by forecasting a single value, disregarding outliers by treating them as normal instances in Case 1. The best performances are

achieved by the BiLSTM model. In Case 2, outliers were considered as missing values then filled based on linear interpolation. MAE and Pbias metrics show better forecasting performances for all the models in Case 2 compared to Case 1. Indeed, all models achieved improved performances when outliers values were replaced. The LSTM model achieved the best forecasting performances (MAE value of 6.24 and Pbias value of -1.2%) followed by BiLSTM (MAE value of 8.38 and a Pbias of -1.61%). These results suggest that replacing outliers and filling missing values increase the energy consumption forecasting performances.

6.2 Resampling Processing Evaluation

Table 3 shows Case 3 results performances where the AMPds dataset was resampled to 1-h frequency. Compared to the 10-min frequency, performances dropped for all the models. The best performance after resampling was achieved by the Hybrid-M1 model with a Pbias of -25.02% compared to -2.06% Pbias performance before resampling. Table 3 results suggest that performances drop when resampling the original data values.

6.3 Features Reduction Evaluation

For both WATT 2.0 dataset and ASHRAE buildings datasets, a correlation cutoff of 0.3 achieved the best performances.

Table 4. Comparing performances before and after features reduction on WATT 2.0 dataset.

Model	Before features reduction			After features reduction		
	MAE	MAPE (%)	R^2	MAE	MAPE (%)	R^2
ERTR	**3.05**	**4.52**	0.96	2.74	4.03	0.97
XGBOOST	3.18	4.68	0.96	**2.71**	**3.99**	**0.97**
LightGBM	3.54	5.38	0.95	3.38	5.12	0.96
ANN	5.69	7.47	0.89	3.35	5.07	0.96
CNN	4.46	6.28	0.94	3.55	5.34	0.95
LSTM	4.07	6.07	0.94	3.60	5.51	0.95
BILSTM	3.64	5.50	0.95	3.39	5.02	0.96
GRU	5.06	6.85	0.93	3.27	5.04	0.96
Hybrid-M1	4.64	6.61	0.93	3.97	5.82	0.95
Hybrid-M2	3.55	5.24	0.95	3.36	4.86	0.96

Due to space constraints, we show, in Table 4, the WATT 2.0 results for a Week-ahead forecast (168 predicted values) before and after features reduction. WATT 2.0 features were first reduced from 9 features to 4 features (T, Ta, S, Bh)

after applying a PCC >= 0.3 between the feature P (dependent variable) and the other features (independent variables). Then after applying the correlation between the independent features, apparent temperature was considered as a redundant feature and dropped due to its high correlation with the temperature.

For all the datasets, better performances were achieved when applying features reductions. In Table 4, the XGBOOST model stood out with MAE of 2.71 and a MAPE of 3.99%. Additionally, there was a slight enhancement of R^2, improving from 0.96 to 0.97 when using only correlated features.

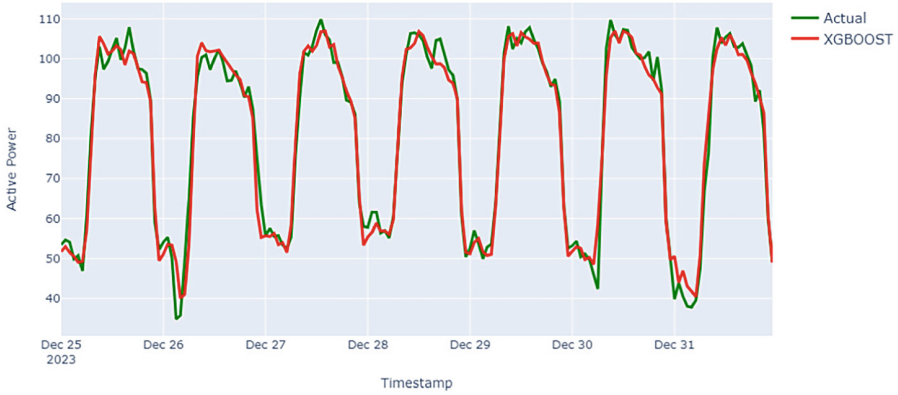


Fig. 2. Week-ahead WATT 2.0 actual energy consumption and XGBOOST prediction results.

The predicting results of the XGBOOST applied to WATT 2.0 dataset was compared with actual energy consumption values, as shown in Fig. 2. The prediction curve fit well; whether in the peaks or troughs, the green line (Actual) and the red line (Forecast) almost overlaps.

6.4 Forecasting Performances

Table 5 shows model performances applied on a Week-ahead forecast after applying the data preprocessing steps. For instance, ASHRAE B-1081 best performances were achieved by the ERTR model with MAE, MAPE and R^2 values of 0.48, 2.98% and 0.97 respectively. Results indicate small-scale magnitude of estimated absolute error, low error percentage relative to the actual value and high degree of correlation between predicted and actual values.

Results indicate that the applied models have comparative forecasting performances. ML models achieved competitive or better performances compared to Deep Learning models and Hybrid models performances. Indeed, ERTR and XGBOOST models achieved the best performances when applied to the ASHRAE B-164 dataset, ASHRAE B-345 dataset, ASHRAE B-276 dataset and WATT 2.0 dataset. The model performances rely heavily on the datasets features

Table 5. Forecasting performances on the ASHRAE datasets;

Model	ASHRAE B-164			ASHRAE B-346			ASHRAE B-1081			ASHRAE B-276		
	MAE	MAPE	R^2	MAE	MAPE	R^2	MAE	MAPE	R^2	MAE	MAPE	R^2
ERTR	1.74	8.76	0.70	**0.49**	**2.81**	**0.98**	**0.48**	**2.98**	**0.97**	3.51	2.47	0.96
XGBOOST	1.93	9.44	0.68	0.80	4.70	0.96	0.56	3.49	0.96	**3.37**	**2.34**	**0.97**
LightGBM	1.81	8.93	0.68	0.55	3.15	0.98	0.58	3.65	0.96	4.01	2.81	0.95
ANN	1.65	8.33	0.77	0.62	3.53	0.97	0.57	3.60	0.96	3.90	2.70	0.96
CNN	1.67	8.01	0.76	0.53	2.97	0.98	0.56	3.46	0.95	5.06	3.65	0.93
LSTM	**1.49**	**7.50**	**0.80**	0.60	3.44	0.98	0.60	3.86	0.95	4.21	3.01	0.95
BILSTM	1.67	8.11	0.76	0.68	3.63	0.97	0.56	3.59	0.96	4.22	3.02	0.95
GRU	1.60	8.19	0.76	0.56	3.11	0.98	0.57	3.36	0.96	3.58	2.53	0.96
Hybrid-M1	1.60	7.65	0.77	0.72	3.92	0.97	0.50	3.06	0.97	3.82	2.74	0.96
Hybrid-M2	1.63	8.11	0.78	1.02	5.37	0.89	0.81	5.31	0.92	4.32	3.07	0.94

and characteristics more than the datasets sizes. Results suggest that high performances short-term (one week) forecasting energy consumption based on unsupervised learning-based approaches can be achieved avoiding the time-consuming data labeling step, such as a status feature that assumes 'on' or 'off' values based on the appliance's status, in supervised learning-based approaches in addition to avoiding the statistical models drawbacks.

7 Conclusion

In this study, we explored several recent models for short-term (one week) energy consumption forecasting. We compared their performance across datasets having different primary use. Our findings revealed the importance of applying preprocessing steps before applying any learning-based algorithms. More specifically, instead of dropping rows containing outliers or missing values, these rows' values have to be filled based on linear interpolation. In addition, finding results showed that performances drop when resampling the original data values and performances increase when reducing the features by applying Pearson Correlation Coefficient in order to eliminate irrelevant (uninformative features) to the forecasting problem or redundant (increasing the dimensionality) features. ML models, especially ensemble models such as ERTR and XGBOOST, can match or achieve better performances than Deep learning models and hybrid models.

The study is of course subject to limitations and provides room for further research. First, we will add more features that might impact buildings energy consumption such as buildings orientation, building shape, envelope system, passive heating, and cooling mechanisms. Second, as ML models evolve and new models are introduced, we aim to evaluate the performances of deep generative models such as Variational Autoencoder and Generative Adversarial Networks models in addition to their combination.

References

1. Tian, C., Ma, J., Zhang, C., Zhan, P.: A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network. Energies **11**(12), 3493 (2018)
2. Li, C., Ding, Z., Zhao, D., Yi, J., Zhang, G.: Building energy consumption prediction: An extreme deep learning approach. Energy **10**(10), 1525 (2017)
3. Liu, X., Niu, Z., Yang, Y., Wu, J., Cheng, D., Wang, X.: VAP: a visual analysis tool for energy consumption spatio-temporal pattern discovery. In: 23rd International Conference on Extending Database Technology, pp. 579–582. OpenProceedings.org, Copenhagen, Denmark (2020)
4. Kim, T.Y., Cho, S.B.: Predicting residential energy consumption using CNN-LSTM neural networks. Energy **182**, 72–81 (2019)
5. Siami-Namini, S., Tavakoli, N., Namin, A.-S.: A comparison of ARIMA and LSTM in forecasting time series. In: 17th IEEE International Conference on Machine Learning and Applications, pp. 1394–1401. IEEE, Florida, USA (2018)
6. Siami-Namini, S., Tavakoli, N., Namin, A.-S.: The performance of LSTM and BiLSTM in forecasting time series. In: International Conference on Big Data (Big Data), pp. 3285–3292. IEEE, California, USA (2019)
7. Dubey, A.K., Kumar, A., García-Díaz, V., Sharma, A.K., Kanhaiya, K.: Study and analysis of SARIMA and LSTM in forecasting time series data. Sustainable Energy Technol. Assess. **47**, 101474 (2021)
8. Kontopoulou, V.I., Panagopoulos, A.D., Kakkos, I., Matsopoulos, G.K.: A review of ARIMA vs. machine learning approaches for time series forecasting in data driven networks. Future Internet **15**(18), 100121 (2023)
9. Akhtar, S., Shahzad, S., Zaheer, A., Ullah, H.S., Kilic, H., Gono, R., Jasiński, M., Leonowicz, Z.: Short-term load forecasting models: a review of challenges, progress, and the road ahead. Energy **16**(10), 4060 (2023)
10. Chen, Y., Fu, Z.: Multi-step ahead forecasting of the energy consumed by the residential and commercial sectors in the United States based on a hybrid CNN-BiLSTM model. Sustainability **15**(3), 1895 (2023)
11. Kong, W., Dong, Z.Y., Jia, Y., Hill, D.J., Xu, Y., Zhang, Y.: Short-term residential load forecasting based on LSTM recurrent neural network. IEEE Trans. Smart Grid **10**(1), 841–851 (2017)
12. Li, Z., Liu, X., Lin, Y., Xu, X., Wang, X.: Energy efficiency prediction of energy storage virtual synchronous machine based on long short-term memory network. J. Phys: Conf. Ser. **2665**(1), 012014 (2023)
13. Yan, K., Li, W., Ji, Z., Qi, M., Du, Y.: A hybrid LSTM neural network for energy consumption forecasting of individual households. IEEE Access **7**, 157633–157642 (2019)
14. Pallonetto, F., Jin, C., Mangina, E.: Forecast electricity demand in commercial building with machine learning models to enable demand response programs. Energy AI **7**, 100121 (2022)
15. Khan, Z.A., Hussain, T., Ullah, A., Rho, S., Lee, M., Baik, S.W.: Towards efficient electricity forecasting in residential and commercial buildings: a novel hybrid CNN with a LSTM-AE based framework. Sensors **20**(5), 1399 (2020)
16. Ungureanu, S., Topa, V., Cziker, A.C.: Deep learning for short-term load forecasting-Industrial consumer case study. Appl. Sci. **11**(21), 10126 (2021)
17. Botchkarev, A: Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. arXiv preprint arXiv:1809.03006 (2018) https://arxiv.org/abs/1809.03006. Accessed 12 Dec 2023

18. Makonin, S., Popowich, F., Bartram, L., Gill, B., Bajić, I.V.: AMPds: a public dataset for load disaggregation and eco-feedback research. In: IEEE Electrical Power & Energy Conference, pp. 1–6. IEEE, Nova Scotia, Canada (2013)
19. Miller, C., et al.: The ASHRAE great energy predictor III competition: overview and results. Sci. Technol. Built Environ. **26**(10), 1427–1447 (2020)
20. Snedecor, G.W.C., William, G.: Statistical Methods, 8th edn. Iowa State University Press, USA (1989)
21. Noor, M.N., Yahaya, A.S., Ramli, N.A., Al Bakri, A.M.: Filling missing data using interpolation methods: study on the effect of fitting distribution. Key Eng. Mater. **594**, 889–895 (2014)
22. Zou, K.H., Tuncali, K., Silverman, S.G.: Correlation and simple linear regression. Radiology **227**(3), 617–628 (2003)
23. Schober, P., Boer, C., Schwarte, L.A.: Correlation coefficients: appropriate use and interpretation. Anesthesia Analgesia **126**(5), 1763–1768 (2018)
24. Gupta, H.V., Sorooshian, S., Yapo, P.O: Status of automatic calibration for hydrologic models: Comparison with multilevel expert calibration. J. Hydrologic Eng. **4**(2), 135–143 (1999)
25. John, V., Liu, Z., Guo, C., Mita, S., Kidono, K.: XGBoost: real-time lane estimation using deep features and extra trees regression. In: 7th Pacific-Rim Symposium (PSIVT), pp. 721–733. Springer International Publishing, Auckland, New Zealand (2016)
26. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. IEEE, CA, USA (2016)
27. Ke, G., et al.: Lightgbm: a highly efficient gradient boosting decision tree. Advances in neural information processing systems 30 (2017)
28. Petrus, J.B., Thuijsman, F., Weijters, A.J.: Artificial neural networks: an introduction to ANN theory and practice. Springer Science & Business Media 931 (1995)
29. Yao, G., Lei, T., Zhong, J.: A review of convolutional-neural-network-based action recognition. Pattern Recogn. Lett. **118**, 14–22 (2019)
30. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS 2014 Workshop on Deep Learning (2014)
31. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(9), 1735–1780 (1997)
32. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)
33. Rashid, H., Batra, N. and Singh, P.: Rimor: towards identifying anomalous appliances in buildings. In: Proceedings of the 5th ACM Conference on Systems for Built Environments, pp. 33–42. ACM, Shenzen, China (2018)

Machine Learning Models for Electricity Generation Forecasting from a PV Farm

Adam Krechowicz[1], Maria Krechowicz[2(✉)], and Artur Pawelec[2]

[1] Faculty of Electrical Engineering, Automatic Control and Computer Science, Kielce University of Technology, Kielce, Poland
a.krechowicz@tu.kielce.pl

[2] Faculty of Management and Computer Modelling, Kielce University of Technology, Kielce, Poland
{mkrechowicz,apawelec}@tu.kielce.pl

Abstract. Accurate forecasting of the electricity generation from photovoltaic farms plays a significant role in their proper technical and financial management. Reliable forecasts enable management of inertia and frequency response during contingency events and proper planning of the spinning reserve of PV farms. In this work, six machine learning models applying Convolutional Neural Network, Extreme Learning Machine, Random Forest Regression, Gradient Boosted Regression, AdaBoosted Regression, and K-Nearests Neighbors Regression were proposed to forecast electricity generation from a 700 kW photovoltaic farm located in Poland. The models were developed based on four widely available meteorological parameters: ambient air temperature, cloudopacity, and relative humidity, and hour, day, month and year. The comparative performance of the model revealed that the gradient-boosted regression was the most reliable with the determination coefficient $R^2 = 95.503\%$, the mean absolute error MAE = 15.003 kWh, and the root mean square error RMSE = 29.975 kWh.

Keywords: machine learning · energy generation forecasting · PV farm · renewable energy

1 Introduction

Today, we are faced with an increase in energy demand due to population growth, technological and urban development [18,19]. Covering this demand to a large extent by fossil fuels negatively impacts the environment by polluting the atmosphere and increasing the carbon footprint [16]. Implementation of the assumptions of The European Green Deal (reduction of green house emissions by at least 55% by 2030, Europe as the first climate-neutral continent by 2050) [7] is possible thanks to energy generation from Renewable Energy Sources (RES).

Forecasting electricity generation from photovoltaic (PV) farms is a complex task, due to the high volatility of PV power generation (from 0 to 100%) according to meteorological and climatic conditions and the geographic characteristics

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 252–264, 2024.
https://doi.org/10.1007/978-3-031-63215-0_19

of the region [17]. PV farms operators must provide electricity generation forecasts in advance [5,25]. Reliable and accurate electricity generation forecasts from PV farms are required to manage inertia and frequency response during contingency events, as well as scheduling and capacity planning [24]. The lack of them impairs proper operation, balancing, and scheduling of PV farms, threatening the security of the grid.

In the case of locating PV farms in climates characterized by large fluctuations in solar radiation and temperature, it is more difficult to predict the energy production from photovoltaic farms. These are the typical conditions of Poland, located in a warm, temperate transitional climate.

The purpose of this work is to develop and compare the performance of various machine learning (ML) models to forecast electricity generation from a PV farm located in Poland. The contents of this paper are arranged as follows. Section 2 describes the review of the literature on ML applications for the generation of electricity from photovoltaic systems. Section 3 presents the proposed approach, including data collection and pre-processing, correlation analysis of meteorological parameters with energy production from a PV farm, and ML models development and evaluation. Section 4 shows results and a discussion. The paper ends with a conclusion.

2 Literature Review

There are several methods of forecasting the electricity generation from PV systems: numerical weather prediction (NWP), forecast based on sky images, and forecasting using machine learning models [32]. In [39] NWP was successfully applied for hour-ahead solar forecasting in the United States. In [36] short-term solar irradiance forecasting model that combines ML with processed sky images and cloud motion was proposed.

In [15] a review of ML applications for forecasting of electricity generation from Renewable Energy Sources was presented. Based on the analysis of 262 relevant research articles from the Scopus database published in 2020–2022, it was found that ML models are a promising solution for forecasting electricity generation from RES in various regions of the world. The study revealed that Extreme Learning Machine and the ensemble methods were the most popular for the forecasting of electricity generation from RES in the years analyzed, and hybrid models were very popular. Moreover, most of the works concerned short-term forecasting. In [38] support vector machine (SVM) and Gaussian process regression (GPR) were applied to forecast solar PV power in a tropical climate in Nigeria. It was found that GPR outperformed the SVM model. In [2] an artificial neural networks model for prediction of daily average solar radiation in Kuwait was proposed. In [24] PV power generation forecasting in Australia was carried out using various models: Linear Regression, Polynominal Regression, Decision Tree Regression, Support Vector Regression, Random Forest Regression, and Multilayer Perceptron Regression. It was found that Random Forest Regression model performed the best. In [13] pattern identification of PV energy

generation model was developed applying several deep neural architectures, such as Long short-term memory (LSTM), Gate recurrent unit, Convolution Neural Network, and Autoencoder. In [28] a deep learning approach to PV energy generation forecasting using Long short-term memory neural network, adaptive neuro-fuzzy inference system (ANFIS) accompanied by fuzzy c-means and ANFIS with grid partition. LSTM model got the best results with RMSE = 60.66 kWh, MAE=30.47 kWh, and R^2 = 0.9777.

The geographical location of a PV farm significantly impacts the behavior of the ML model that aims to forecast the generation of electricity from a PV farm [31]. That is why a model developed, e.g. for one USA region may not work well in another USA region, not to mention a model developed for Australia may not work on Polish dataset. It is important to stress that the location of Poland in a warm temperate transitional climate with large fluctuations in solar radiation and temperature makes the PV power forecasting task more difficult. Depending on the climate and geographical location, the generation of photovoltaic power may depend on various meteorological parameters. Generally, it depends on solar irradiance, relative humidity, temperature, and wind speed [10], while solar irradiance generally depends among others on temperature, pressure, dew point, relative humidity, wind speed, and direction [33]. Jurasz [12] simulated Polish PV power generation on a national applying artificial neural network. The analyzed dataset covered data from 23 meteorological stations in one month. Chwieduk [4] compared a real PV solar power with results of a SODA forecasting model for a Polish city, Falenica. It resulted in low absolute error for annual energy generation forecasting and a significant error for monthly forecasting.

3 Proposed Approach

The proposed approach consists of three steps and is shown in Fig. 1. Each step was described in detail in Subsects. 3.1–3.3.

3.1 Data Collection and Preprocessing

The hourly data concerning electricity generation were collected from a 700 kW PV farm located in Poland in the świętokrzyskie voivodeship in Poland from 1 January 2019 to 21 December 2021. Several technical breaks or failures in the operation of the PV power plant lead to the occurrence of inconsistent data, which were filtered. Table 1 presents the parameters that characterize the analyzed PV farm. Figure 2 presents a sample data form January and Fig. 3 from June.

The meteorological data was collected from the nearby meteorological station in Kielce and the Solcast database [35]. Data concerning the following parameters was gathered: year, month, day, hour, cloudopacity (%), dew point, diffuse horizontal irradiance (DHI) (W/m2), direct normal irradiance DNI (W/m2), Direct (beam) horizontal irradiance (EBH) (W/m2), horizontal global irradiation (GHI) (W/m2), pressure (hPa), wind speed (m/s), ambient air temperature (°C), and relative humidity (%).

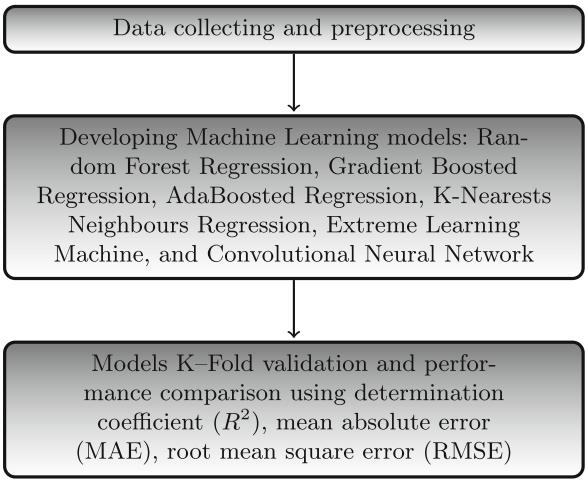


Fig. 1. Proposed approach.

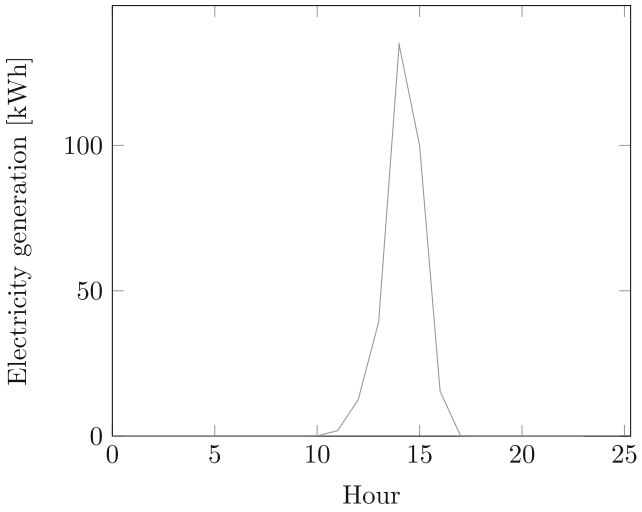


Fig. 2. A sample data from a day from January.

Table 1. The parameters characterizing the analyzed PV farm.

Parameter	Value
The number of PV modules	2540
The type of PV modules	Solar polycrystalline PV modules
Nominal power of a module	275 Wp
Module efficiency	16.80%
Fill factor	76.65%
Inclination of the modules	30° to the South
Installed power of the plant	700 kW

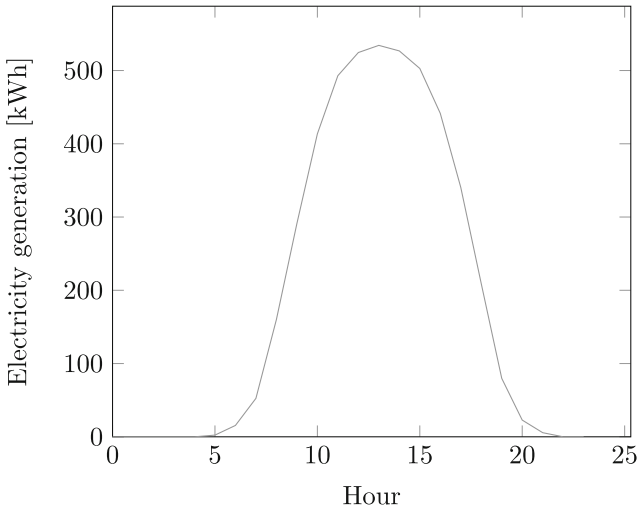


Fig. 3. A sample data from a day from June.

In this work, for Extreme Learning Machine and Convolutional Neural Network models, the z-score standardization technique was applied to make those models results independent of large absolute values. Z-score enables to show how far the analysed value is from the mean in terms of standard deviation.

3.2 Machine Learning Models Development

Convolutional Neural Network (CNN) belongs to deep learning models to process data. The inspiration for its grid pattern was the visual cortex of the animal [9]. It can automatically and adaptively learn the spatial hierarchies of the characteristics. It usually consists of 3 kinds of layers: called convolution, pooling, and fully connected layers. Feature extraction is carried out in convolution and pooling layers, while a fully connected layer aims at mapping the extracted features into the final result. The most important mathematical operations that significantly impact CNN performance are carried out in a convolution layer [37].

The extreme learning machine (ELM) was first introduced to improve the performance and speed of a single-hidden-layer feedforward network. A characteristic feature of the ELM algorithm is the fact that it does not require the tuning of hidden nodes (neurons), as it randomly assigns them, develops biases and input weights of hidden layers, and generates output weights applying least squares methods [6]. Thanks to this, compared to Artificial Neural Network, it can offer a shorter experiment time. Its main advantages are efficient learning speed, fast convergence, good generalizability, and easy implementation [27].

Random Forest Regression (RFR) is an ensemble model, so its predictive power is greater than that of single models [20]. In this model, a number of decision tree regressors are fitted on various subsamples of the dataset and averaging

is applied to upgrade the and control overfitting [29]. Its accuracy is usually high due to the derivation of the final result due to extracting the most common values on average applying the results obtained through several decision trees [22,34].

In Gradient Boosted Regression (GBR), a prediction model is developed in the form of an ensemble of weak prediction models employing a boosting strategy to bring their output together in a strong "committee" [11]. In the case of GBR additive models are learned in a forward stage-wise manner. Its main advantages are high effectiveness, simplicity, and interpretability [30].

The AdaBoosted algorithm, first presented in [8], is known as an effective ensemble technique, in which altering the distribution of sample weights results in the effectiveness increase of weak learners [14,23]. AdaBoosted Regression (ABR) fits a regressor on the analyzed data set, subsequently adjusting the copies of supplementary regressors in this data set, fitting the weights of instances considering the error of the current forecasting [29].

K-Nearests Neighbors Regression (KNR) is a nonparametric ML model capable of making predictions based on the target output of the nearest neighbors of the query point [1]. In the case of a regression based on the nearest K neighbors, an overfitting problem can happen when chosen K is too low and data blur when K value is too high [3].

When performing the feature selection, the literature review was carried out on the features taken into account by other researchers in various countries. Its results were described in another work of the authors [15]. Furthermore, the information from the correlation analysis between various meteorological parameters and the electricity production from a PV farm in Poland described in another work by the authors [17] and the possibility of getting parameters from real weather forecasts were taken into account. It was decided to carry out many experiments in which various combinations of individual attributes (meteorological parameters) were considered. The experiments show that the best results were obtained considering the following input parameters: ambient air temperature (°C), cloudopacity (%), relative humidity (%), year, month, day, and hour. When forecasting, the models took these input features into account with a time shift of 6 h. An output parameter was the electricity generation from the PV power plant.

In order to select the best models that the best captures the electricity generation from a PV plant many experiments were carried out in Python applying SciKit learn libraries.

The best parameters for each ML model were chosen experimentally and are presented in Table 2.

3.3 Models Evaluation

Training the model was carried out using 80% of randomly selected data. Testing was carried out using the remaining 20% of the data. The stability of the proposed ML models in terms of the choice of training and test set was determined using 5-K fold validation. That is why the training set was additionally divided into 5 subsets to enable 5 experiments to be performed (each of the five subsets

Table 2. Parameters and hyper–parameters description of proposed models

Model	Hyper–parameters
Random Forest Regression	number of trees = 200, maximum depth = unlimited, minimum samples to split = 3, minimum samples at leaf = 1, bootstrap = False, number of features=sqrt
Gradient Boosted Regression Tree	learning rate = 0.1, number of estimators = 200, maximum depth = 7, maximum number of leafs = unlimited, fraction of subsamples = 100%, number of features=sqrt
AdaBoosted Regression Tree	learning rate = 0.1, number of estimators = 200
K-Nearests Neighbors Regression	K = 7, algorithm = BallTree, weights = distance based
Convolutional Neural Network	learning rate = 0.001, epochs = 30, network structure = Fig. 4

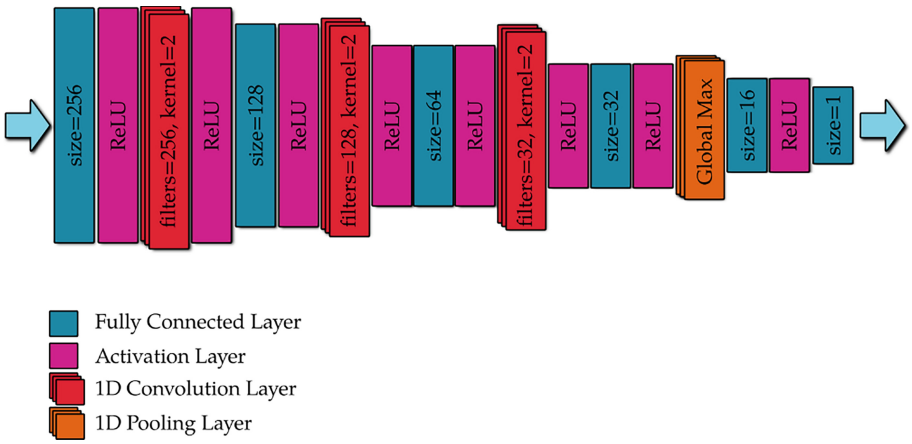


Fig. 4. The architecture of CNN model

served as the validation set, and the remaining subsets served as the training set). The proposed ML models were evaluated using average values of the metrics calculated from the experiments carried out and their standard deviations. The most commonly used metrics were applied: determination coefficient (R^2), mean absolute error (MAE), root mean square error (RMSE). Furthermore, the standard deviations received during the 5-K-fold cross-validation were analyzed. The dependency between predicted and real PV electricity generation was measured using the determination coefficient. R^2 values close to 1 indicate a good fit of the model [24]. Formula (1) defines R^2.

MAE reflects the uniform error in the prediction. It is the measure of the difference between the predicted and real PV electricity generation. Formula (3) defines MAE).

RMSE informs about the largest error in the predicted data set [21]. Formula (4) defines MAE.

$$R^2 = 1 - \frac{var(E_{real} - E_{forecast})}{var(E_{forecast})} \tag{1}$$

where: E_{real} – real electricity generation from the PV power plant (kWh), $E_{forecast}$ – forecasted electricity generation from the PV power plant (kWh).

MAE informs about a uniform error in the prediction, measuring the difference between the forecasted and real electricity generation from the PV power plant. MAE is defined by formula 2.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \mid E_{forecast,i} - E_{real,i} \mid \tag{2}$$

MSE reflects the largest error in the forecasted data set [21] and is defined by formula 3. Whereas RMSE is calculated after taking the square root from MSE.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(E_{forecast,i} - E_{real,i})^2 \tag{3}$$

Table 3. A performance comparison of the proposed models using analyzed metrics.

Model	mean R^2	std R^2	mean MSE	std MSE	mean RMSE	std RMSE	mean MAE	std MAE
CNN	94.982%	0.005	1001.801	88.698	31.620	1.416	14.992	0.433
RFR	95.073%	0.003	984.068	33.122	31.365	0.530	14.781	0.255
GBR	95.503%	0.002	898.675	26.285	29.975	0.438	15.003	0.229
ABR	81.808%	0.006	3637.107	115.851	60.301	0.958	37.183	0.701
KNR	85.549%	0.005	2891.533	157.388	53.753	1.469	29.553	0.900
ELM	80.714%	0.011	4396.388	288.864	66.270	2.165	47.731	0.983

4 Experimental Results and Discussion

Table 3 shows a performance comparison of the proposed models using analyzed metrics.

The performance of the proposed models in predicting the electricity generation from the PV farm was very good (for the models GBR, CNN, and RFR with $R^2 \geq 94.982\%$), and good for ABR, KNR and ELM with $R^2 \geq 80.714\%$). GBR model had the best performance with the highest determination coefficients R^2 = 95 503%, the lowest MSE = 889.675, the lowest RMSE = 29.975 kWh and the lowest MAE = 15.03 kWh, which was fully satisfactory for the owner of the PV farm.

Figure 5 presents the comparison of the performance of the ML models for three of the best ML models for the day, in which the best results were obtained. Figure 6 presents the comparison of ML models performance for three of the best ML models for the day with the greatest average cloud cover.

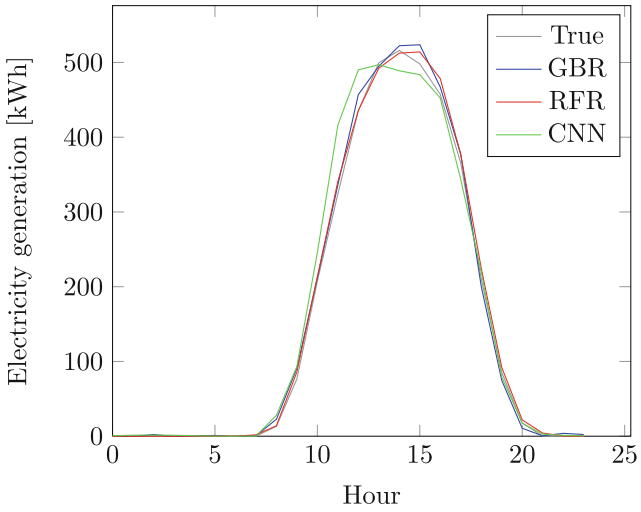


Fig. 5. Comparison of ML models performance for three the best ML models for the day, in which the best results were obtained.

Table 4 presents a comparison of the results obtained with the results obtained by other authors for various PV installations. In this table, PV installations located in various countries were analyzed, their size, the ML method applied, and the results. In some cases, it was difficult to compare the results, as the PV installation size was not given, the results were based on data from only one day or there was no information if cross-validation was carried out. It is important to stress that the installations compared came from various climates. The analysis revealed that the proposed model performs very well, despite the location of the analyzed photovoltaic farm in a climate characterized by large

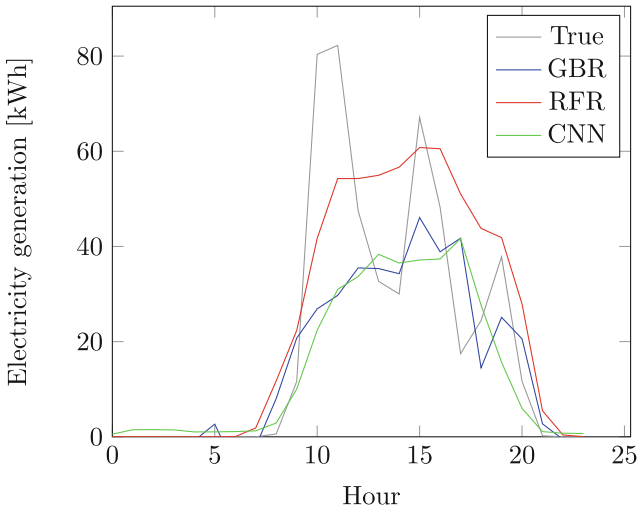


Fig. 6. comparison of ML models performance for three the best ML models for the day with the greatest average cloud cover.

fluctuations in solar radiation and temperature, making the prediction task more difficult. Conversations with owners of photovoltaic farms show that R2 values above 0.85 are acceptable for them. The MAE and RMSE values depend on the capacity of the PV installation. The larger the installation capacity, the higher the acceptable MAE and RMSE error may be. Normalization of results can be used to compare MAE and RMSE values for PV installations with different capacities. For example, based on the experience of the authors, it can be concluded that the desired MAE values after normalization are values up to 0.05, and the desired RMSE values after normalization are values up to 0.08. The values obtained in this work meet this condition. The meaning of performing the cross-validation should be stressed, as it indicates the reliability of the model. Random selection of sets (usually 5) for validation lowers the possibility of getting high model parameters at the end of the day and the risk of overfitting the model. In some cases, very good results without cross-validation may turn out to be random.

Table 4. The comparison of the obtained results with the results obtained by other authors.

Reference	Installation location	Installation size	Method	Results
This work	Poland	700 kW	GBR	$R^2 = 95.503\%$ MSE = 898.675, RMSE = 29.975 kWh, MAE = 15.003 kWh
[26]	India	544 W	SVR	MAE = <34.8839 W 69.1791 W>
[28]	Turkey	1.15 MW	LSTM	MAE = 30.47 kWh, RMSE = 60.66 kWh
[24]	Australia	Not given	LR, PR, SVR, DT, RFR, MLP and LSTM	R^2 = <-140.27%, 98.80%>
[38]	Nigeria	Not given	SVM, GPR	R^2 = <88%,98%> (Only for the data from one day)

5 Conclusions

Developing accurate machine learning models to predict PV power output is vital not only for the owner of a PV farm, but also for the operators of the grid. It allows scheduling, effectively managing inertia and frequency response during contingency events, and planning the capacity, supporting the security of the grid. The forecasting task is more difficult for climates characterized by large fluctuations in solar radiation and temperature, such as the warm temperate transitional climate in which Poland is located.

In this study, six Machine Learning models applying Convolutional Neural Network, Extreme Learning Machine, Random Forest Regression, Gradient Boosted Regression, AdaBoosted Regression, and K-Nearests Neighbors Regression were developed to forecast electricity generation from a PV farm of 700 kW located in Poland. The models were developed based on widely available meteorological parameters: ambient air temperature, cloudopacity, and relative humidity, and hour, day, month and year. The comparative performance of the model showed that Gradient-Boosted Regression was the most reliable model with determination coefficient $R^2 = 95.503\%$, mean absolute error MAE = 15.003 kWh, and root mean square error RMSE = 29.975 kWh.

The limitation of the proposed approach is the fact that the proposed ML models were trained and tested on a historical data set since the weather forecast data had not been collected. Therefore, future work is planned to develop novel machine learning models that work on weather parameters taken from weather forecasts. It may happen that the ML performance may slightly change due to the forecast error inherent in the weather forecasts. Therefore, more experiments with various ML models and hyperparameters may be needed to integrate ML models with available weather forecast data. Furthermore, because of the increasing problem of air pollution, dust and smog in the world, future work is planned to develop such ML models that could capture the effect of smog on the energy generation from photovoltaic panels. This will be of great importance for industrial regions, large cities, and areas with high air pollution. In addition to this, limitations of this work include the fact that, due to climate changes, the accuracy of the models can lower over time. Therefore, there may be a need to develop a retraining module, which allows one to retrain the model with new weather patterns.

References

1. Ahmed, N.K., Atiya, A.F., Gayar, N.E., El-Shishiny, H.: An empirical comparison of machine learning models for time series forecasting. Economet. Rev. **29**(5–6), 594–621 (2010)
2. Bou-Rabee, M., Sulaiman, S.A., Saleh, M.S., Marafi, S.: Using artificial neural networks to estimate solar radiation in Kuwait. Renew. Sustain. Energy Rev. **72**, 434–438 (2017)
3. Bruce, P., Bruce, A., Gedeck, P.: Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python. O'Reilly Media (2020)

4. Chwieduk, M.: Use of solar radiation data from HelioClim database for short-term PY system power output prediction for polish localization. Pol. Energetyka Słoneczna (2017)
5. Csereklyei, Z., Qu, S., Ancev, T.: The effect of wind and solar power generation on wholesale electricity prices in Australia. Energy Policy **131**, 358–369 (2019)
6. Deo, R., Samui, P., Roy, S.S.: Predictive Modelling for Energy Management and Power Systems Engineering. Elsevier (2020)
7. European green deal. https://ec.europa.eu/clima/eu-action/european-green-deal_en. Accessed 28 Feb 2024
8. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
9. Fukushima, K.: Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol. Cybern. **36**(4), 193–202 (1980). https://doi.org/10.1007/BF00344251
10. Gutiérrez, L., Patiño, J., Duque-Grisales, E.: A comparison of the performance of supervised learning algorithms for solar power prediction. Energies **14**(15), 4424 (2021)
11. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. SSS, Springer, New York (2009). https://doi.org/10.1007/978-0-387-84858-7
12. Jurasz, J., Wdowikowski, M., Figurski, M.: Simulating power generation from photovoltaics in the polish power system based on ground meteorological measurements-first tests based on transmission system operator data. Energies **13**(16), 4255 (2020)
13. Khodayar, M., Khodayar, M.E., Jalali, S.M.J.: Deep learning for pattern recognition of photovoltaic energy generation. Electr. J. **34**(1), 106882 (2021)
14. Krechowicz, A., Deniziak, S., Kaczmarski, D.: Machine learning approach to gait deviation prediction based on isokinetic data acquired from biometric sensors. Gait Posture **101**, 55–59 (2023)
15. Krechowicz, A., Krechowicz, M., Poczeta, K.: Machine learning approaches to predict electricity production from renewable energy sources. Energies **15**(23), 9146 (2022)
16. Krechowicz, M., Krechowicz, A.: Risk assessment in energy infrastructure installations by horizontal directional drilling using machine learning. Energies **14**(2), 289 (2021)
17. Krechowicz, M., Krechowicz, A., Lichołai, L., Pawelec, A., Piotrowski, J.Z., Stępień, A.: Reduction of the risk of inaccurate prediction of electricity generation from PV farms using machine learning. Energies **15**(11), 4006 (2022)
18. Kuźniak, R., Pawelec, A., Bartosik, A., Pawełczyk, M.: Determination of the electricity storage power and capacity for cooperation with the microgrid implementing the peak shaving strategy in selected industrial enterprises. Energies **15**(13), 4793 (2022)
19. Kuźniak, R., Pawelec, A., Bartosik, A.S., Pawelczyk, M.: Determining the power and capacity of electricity storage in cooperation with the microgrid for the implementation of the price arbitration strategy of industrial enterprises installation. Energies **15**(15), 5614 (2022)
20. Kwak, S., et al.: Machine learning prediction of the mechanical properties of γ-TiAl alloys produced using random forest regression model. J. Market. Res. **18**, 520–530 (2022)
21. Lauret, P., Voyant, C., Soubdhan, T., David, M., Poggi, P.: A benchmarking of machine learning techniques for solar radiation forecasting in an insular context. Sol. Energy **112**, 446–457 (2015)

22. Lin, W., Wu, Z., Lin, L., Wen, A., Li, J.: An ensemble random forest algorithm for insurance big data analysis. IEEE Access **5**, 16568–16575 (2017)
23. Liu, Q., Wang, X., Huang, X., Yin, X.: Prediction model of rock mass class using classification and regression tree integrated AdaBoost algorithm based on TBM driving data. Tunn. Undergr. Space Technol. **106**, 103595 (2020)
24. Mahmud, K., Azam, S., Karim, A., Zobaed, S., Shanmugam, B., Mathur, D.: Machine learning based PV power generation forecasting in Alice springs. IEEE Access **9**, 46117–46128 (2021)
25. Muttaqi, K.M., Sutanto, D., et al.: Transactive energy-based planning framework for VPPs in a co-optimised day-ahead and real-time energy market with ancillary services. IET Gener. Transm. Distrib. **13**(11), 2024–2035 (2019)
26. Nageem, R., Jayabarathi, R.: Predicting the power output of a grid-connected solar panel using multi-input support vector regression. Procedia Comput. Sci. **115**, 723–730 (2017)
27. Nanjappan, V., et al.: Big data analytics for sensor-network collected intelligence, Academic Press (2017)
28. Ozbek, A., Yildirim, A., Bilgili, M.: Deep learning approach for one-hour ahead forecasting of energy production in a solar-PV plant. Energy Sources Part A: Recovery Utilization Environ. Eff. **44**(4), 1–16 (2021)
29. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
30. Persson, C., Bacher, P., Shiga, T., Madsen, H.: Multi-site solar power forecasting using gradient boosted regression trees. Sol. Energy **150**, 423–436 (2017)
31. Premalatha, N., Valan Arasu, A.: Prediction of solar radiation for solar systems by using ANN models with different back propagation algorithms. J. Appl. Res. Technol. **14**(3), 206–214 (2016)
32. Shawon, M.M.H., Akter, S., Islam, M.K., Ahmed, S., Rahman, M.M.: Forecasting PV panel output using prophet time series machine learning model. In: 2020 IEEE Region 10 Conference (TENCON), pp. 1141–1144. IEEE (2020)
33. Singla, P., Duhan, M., Saroha, S.: A comprehensive review and analysis of solar forecasting techniques. Front. Energy 1–37 (2021). https://doi.org/10.1007/s11708-021-0722-7
34. Smith, P.F., Ganesh, S., Liu, P.: A comparison of random forest regression and multiple linear regression for prediction in neuroscience. J. Neurosci. Methods **220**(1), 85–91 (2013)
35. Solcast: solar API and weather forecasting tool. https://solcast.com Accessed 27 Feb 2024
36. Tiwari, S., Sabzehgar, R., Rasouli, M.: Short term solar irradiance forecast based on image processing and cloud motion detection. In: 2019 IEEE Texas Power and Energy Conference (TPEC), pp. 1–6. IEEE (2019)
37. Yamashita, R., Nishio, M., Do, R.K.G., Togashi, K.: Convolutional neural networks: an overview and application in radiology. Insights Imaging **9**, 611–629 (2018)
38. Zazoum, B.: Solar photovoltaic power prediction using different machine learning methods. Energy Rep. **8**, 19–25 (2022)
39. Zhang, G., Yang, D., Galanis, G., Androulakis, E.: Solar forecasting with hourly updated numerical weather prediction. Renew. Sustain. Energy Rev. **154**, 111768 (2022)

Pollutant Concentration Prediction by Random Forest to Estimate a Contaminant Source Position

Sidi Mohammed Alaoui[1(✉)], Khalifa Djemal[1], Amir Ali Feiz[2], and Pierre Ngae[2]

[1] Université Paris-Saclay, Univ Evry, IBISC, 91020 Evry-Courcouronnes, France
{sidimohammed.alaoui,Khalifa.djemal}@univ-evry.fr
[2] Université Paris-Saclay, Univ Evry, LMEE, 91020 Evry, France
{Amir.feiz,Pierre.ngae}@univ-evry.fr

Abstract. The goal of Source Term Estimation (STE) is to accurately identify the parameters that describe the source of a release, namely the position and strength . This requires a reliable dispersion model. Recent advancements have integrated machine learning with the Gaussian dispersion model, enhancing the prediction of pollutant concentrations while mitigating the impact of complex terrains. However, pollutant dispersion varies significantly across different atmospheric stability classes. Addressing this, we introduce a novel strategy termed the Multiple Learning Model (MLM), which segments predictions based on stability classes: Neutral (MLM_N), Unstable (MLM_U), and Stable (MLM_S). This approach, by building models to specific atmospheric conditions, promises more precise source estimations. In comparative study, MLM not only refined prediction accuracy from 0.04 to 0.06 but also improved source location estimates, narrowing the discrepancy to 7 m–25 m from the actual source, a marked improvement over traditional random forest models. This methodological advancement underscores the potential of stability-class-specific models in enhancing the accuracy and reliability of pollutant source estimations.

Keywords: Source term estimation · Gaussian dispersion model · Random forest · Multiple Learning Model · Bayesian inference · Markov Chain Monte Carlo

1 Context and Introduction

According the the World Health Organization, air pollution affects both indoor and outdoor environments through various contaminants, posing significant public health risks. Notable pollutants such as particulate matter (PM), carbon monoxide (CO), ozone (O3), nitrogen dioxide (NO2), and sulfur dioxide (SO2) are linked to respiratory diseases and increased mortality rates [17]. Addressing pollution in local scale typically involves efforts to determine the source of pollutants by their location and intensity, crucial for effective management and mitigation.

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 265–278, 2024.
https://doi.org/10.1007/978-3-031-63215-0_20

To accurately identify pollutant sources, methods of source term estimation are explored, and while optimization methods have been traditionally employed [2–5], their reliability is often difficult to achieve, and this lead to a preference for stochastic approaches [6,7]. The methodology involves two key steps: developing a forward model to simulate pollutant dispersion, and using an inverse model for source identification. Computational models like the Lagrangian and Eulerian are resource-intensive but necessary for complex settings [1,19,20], whereas the Gaussian dispersion model is suitable for simpler and smaller areas.

In this article, We explore source estimation through inverse problem resolution using Delayed Rejection Adaptive Metropolis (DRAM) algorithm within a Bayesian framework, highlighting the forward model's accuracy as a significant factor [8,9,13].

We introduce a novel integration of machine learning with the Gaussian dispersion model to enhance prediction and source estimation accuracy, employing the Multiple Learning Model (MLM) approach. MLM leverages atmospheric stability, indicated by the Monin-Obukhov Length, to categorize datasets based on atmospheric stability for model construction. The higher accuracy of MLM and the higher reliability of the DRAM algorithm [10,21] led to a better estimation of the source.

Section 2 outlines the process of identifying a pollutant source from developing and optimizing a Random Forest model using Genetic algorithm ([20]) for pollutant concentration prediction, the Bayesian framework, DRAM algorithm for source identification, to the construction of MLM model. In Sect. 4, Empirical validation is conducted using the Indianapolis campaign data, with MCMC sampling and the DRAM method for Bayesian inference of source parameters [10,21,24]. The MLM's performance is compared against conventional models, demonstrating its effectiveness in pollutant source estimation.

2 Proposed Multiple Learning Model

In this section, we detail the methodology for source term estimation, which uses an air dispersion model, also referred to as the forward model, integrated with an inverse problem approach for source estimation. We introduce a novel strategy, the Multiple Learning Model (MLM), designed to enhance the forward model's accuracy. Previous studies [1,10,11] have combined machine learning models with forward models to predict concentrations. Our methodology innovates by constructing machine learning models customized to atmospheric stability classes, and then training the MLM on these classes.

2.1 Gas Dispersion Models and Indianapolis Experiment

Atmospheric dispersion modeling uses the Advection-diffusion equation to track chemical species transport within a flow, for more details [18] did a recent review on air quality modeling. This article focuses on the Gaussian dispersion model.

Gas Dispersion Models. Gaussian dispersion models are based on the advection-diffusion equation's analytical solution. In the case of a continuous point-emission noted Q, at a height h of the ground, the concentration field is presented in the form of a Gaussian, and is written:

$$C(x,y,z) = \frac{Q}{2u\pi\sigma_z\sigma_y}\{exp(\frac{-(z+h)^2}{2\sigma_z^2} + exp(\frac{-(z-h)^2}{2\sigma_z^2})\}\{exp(\frac{-y^2}{2\sigma_y^2})\} \quad (1)$$

Where, x, y, z are the location parameters (downwind, crosswind and vertical distance respectively). u is the mean wind speed in ms^{-1} and h the effective height of the source in m. σ_y and σ_z are the standard deviations of a statistically normal plume in the lateral and vertical dimensions, respectively, finally C is the integrated parameter calculated by Gaussian model.

Indianapolis Experiment. The Indianapolis field study conducted tracer experiments using SF_6 released from an 83.8 m stack at Perry K power plant, Indianapolis, Indiana, USA, with coordinates at UTM-N 4401.59 km and UTM-E 571.40 km. Over September and October 1985, 170 h of data were collected across all stability classes and various wind speeds in 19 blocks of 8 or 9 h each. Meteorological data were sourced from a 94 m high urban building and three 10 m towers in different locales, supplemented by National Weather Service observations and vertical profiles via minisondes and acoustic sounders. Ground-level concentrations were monitored by around 160 stations within a 0.25 to 12.0 km radius from the stack (Fig. 1).

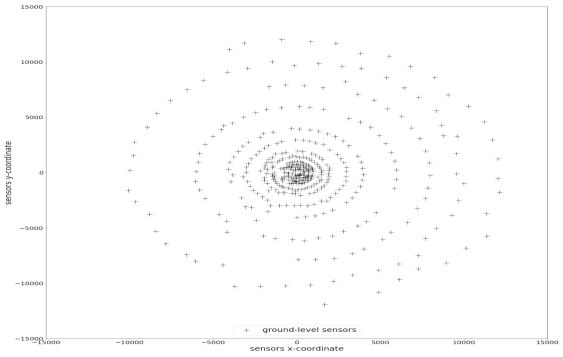


Fig. 1. The 160 gound-level sensors used in the Indianapolis field study

2.2 Concentration Prediction

The Gaussian model, with its straightforward formula, is generally used for predicting concentrations from point sources on flat terrain. Yet, its effectiveness

diminishes under complex conditions. Traditional deterministic methods, while accurate, are too time-consuming for rapid contaminant source identification, crucial in emergency scenarios. To address this, a swift predictive model for the inverse problem is essential [1,11]. Integrating machine learning with the Gaussian model has been proposed, utilizing concentration predictions from the Gaussian model (Eq. 1) or incorporating two features derived from the Gaussian equation as inputs for enhanced accuracy

$$\begin{cases} G_y = exp(\frac{-y^2}{2\sigma_y^2}) \\ G_z = exp(\frac{-(z+h)^2}{2\sigma_z^2} + exp(\frac{-(z-h)^2}{2\sigma_z^2}) \end{cases} \tag{2}$$

Both studies [1,11] used a Support Vector Machine (SVM) [28] as a machine learning model, which showed better accuracy compared to neural network models. However, in this paper, we used the Random Forest model. This choice is Justified later in (**Random Forest Regressor vs Support Vector Regressor**).

To enhance concentration prediction, we aim to identify a machine learning model that aligns with the accuracy and speed benchmarks set by studies in [1,11]. Random Forest, a leading supervised learning method for both regression and classification, emerges as a strong candidate. This model leverages: Decision trees for its foundational structure and bootstrap aggregating (bagging) to train each tree on a randomly selected subset of the data, enhancing prediction reliability by reducing variance.

Our goal is to model the measured concentration Y using atmospheric parameters $X_1, X_2, \ldots, X_p$. We employ the CART algorithm for constructing regression trees, a strategy incorporated in scikit-learn [27]. Despite decision trees' tendency for high variance, Random Forest mitigates this by combining multiple 'weak' learners' predictions [12].

Input Data and Tuning the Hyperparameters of the Model. To construct our statistical model, we used eight input parameters, including wind speed, direction, temperature, and Monin-Obukhov length. The choice of the number of trees (M) in the forest model is crucial; a larger M improves accuracy without causing overfitting [12]. Yet, computational costs rise with M, necessitating a balance between complexity and prediction stability [14]

Genetic Algorithm. We employed a genetic algorithm for hyperparameter tuning [29,30], leveraging the scikit-learn [26] and sklearn-genetic packages [31]. Genetic algorithms optimize search problems using mutation, crossover, and selection.

Random Forest Regressor Vs Support Vector Regressor. We justified choosing the random forest over the SVM model based on the Indianapolis campaign. After hyperparameter tuning via genetic algorithm, random forest showed superior R^2 performance (0.64 vs. 0.45 for SVM). Additionally, SVM's hyperparameter optimization required significantly more time (42 h vs. 13 h for random forest). The faster prediction time of random forest (0.16 s) compared to SVM (1.62 s) further supported our preference for random forest in experiments Table 1 (Fig. 2).

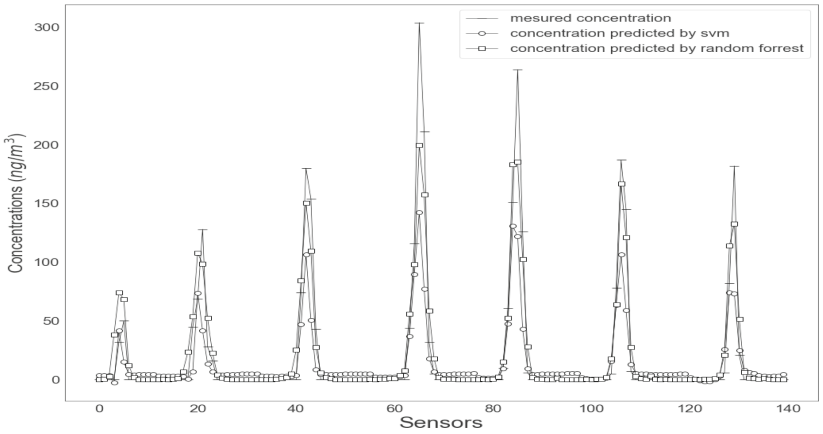


Fig. 2. Concentration prediction of the first 140 randomly selected data points from the test dataset

2.3 Pollutant Source Identification

In this section, we present the process of identifying a pollutant source, as summarized in Fig. 3. The problem of identifying a pollutant source is usually an ill-posed inverse problem, which involve using the measurements from sensors to identify the source parameters. For more details [15] is highly recommended.

In our case, an the problem is formulated as:

$$d = K(x, \theta) + e \tag{3}$$

where K is a function representing the forward model, x is the data, d is a vector of measurements (concentration on sensors), θ is a vector of parameters (source position in our case), and e the error. Assuming e follows a normal distribution with variance σ^2 [32].

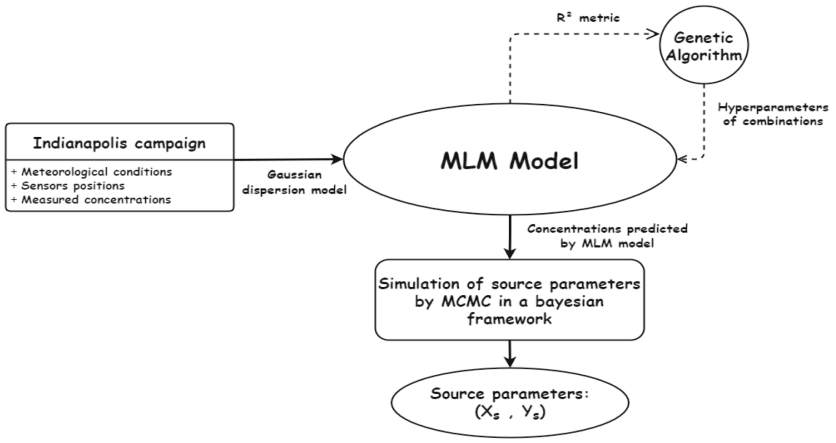


Fig. 3. Overview of the STE, from the measurement campaign to solving the inverse problem with MCMC

A Probabilistic Approach. Bayesian inference, employing Bayes' rule [13], combines prior information, likelihood of measurements given parameters, and seeks the posterior distribution, which is computationally intensive but can be approximated via Markov Chain Monte Carlo (MCMC) sampling. The likelihood function for sensor error e_i is defined as:

$$\mathcal{P}(\theta|d) \propto \mathcal{P}(d|\theta)\mathcal{P}(\theta) \quad with\ estimator\ \ \widehat{\theta} = argmax\, \mathcal{P}(\theta|d) \tag{4}$$

The choice of the estimator is justified by the Law of Large Numbers.

Markov Chain Monte Carlo sing the Bayesian framework, the DRAM algorithm, a fusion of delayed-rejection (DR) and Adaptive-Metropolis (AM), optimizes pollutant source identification. DR improves Metropolis-Hastings efficiency by proposing multiple candidates, with probabilities

$$\alpha_1 = min(\frac{\pi(y_1)q_1(y_1, x)}{\pi(x)q_1(x, y_1)}, 1) = min(\frac{N_1}{D_1}, 1) \tag{5}$$

In the case of rejection in the Metropolis-Hastings algorithm, we set $x_{t+1} = a$ as well. However, the DR method provides an additional opportunity by proposing another candidate y_2 and a distribution based on the first proposal, denoted as $q_2(\cdot, y_1, a)$:

$$\alpha_2 = min(\frac{\pi(y_2)q_1(y_2, y_1)q_2(y_2, y_1, x)(1 - \alpha_1(y_1, y_2))}{\pi(x)q_1(x, y_1)q_2(x, y_1, y_2)(1 - \alpha_1(x, y_1))}, 1) = min(\frac{N_2}{D_2}, 1) \tag{6}$$

The delayed rejection method allows further attempts after rejection, but this study only applies it to the second step.

Adaptative Metropolis (AM). In the context of Adaptive Metropolis (AM), the proposed distribution q_i is a normal distribution with a mean equal to the position of x_{i-1} and a covariance defined as:

$$C_{i+1} = s_d C_t(X_0, X_1, .., X_i) + s_d \epsilon I_d \tag{7}$$

s_d is set based on dimensionality ($s_d = \frac{(2.4)^2}{d}$ [33]), with a small $\epsilon > 0$. [21] demonstrated the ergodicity even if Markovian properties are not maintained.

2.4 Multiple Learning Model (MLM)

The idea behind improving the prediction model (the forward model) is that the behavior of a pollutant gas dispersion depends on the stability class. We can identify these classes based on the Monin-Obukhov Length parameter. In this study, we use the table from [16].

Here, we consider only three classes: stable, neutral, and unstable. To do so, we categorize "very stable," "stable," and "near stable" as stable, "near unstable," "unstable," and "very unstable" as unstable, and the rest as neutral.

From the Monin-Obukhov Length L_{MO} of each experiment in the dataset from the campaign, we can create three datasets. Then, three models will be built, each model corresponding to a dataset. During production, the model used will be based on the atmospheric stability of the situation. The difference between each MLM model in Fig. 4 lies in the hyperparameters. After using the Genetic Algorithm to tune each random forest model on data based on its stability class, we obtain three different models, as described in Table 1 where: $n\_estimators$ Total trees, $max\_features$ is Random features for the best split, where 'sqrt'

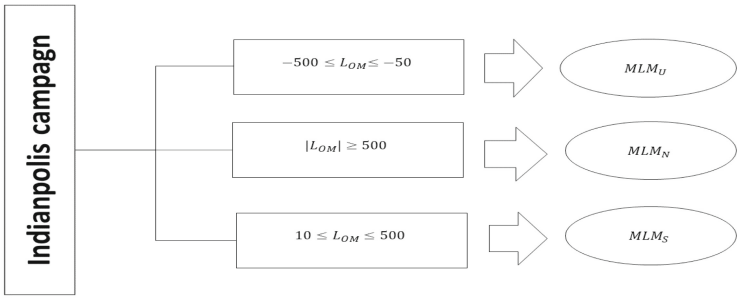


Fig. 4. Three databases based on L_{MO} are created for the Multiple Learning Model (MLM), with each serving a specific stability condition using a Random Forest model.

Table 1. The hyperparameters of the MLM consist of three models: MLM_S, MLM_U, and MLM_N, along with a random forest model RF trained on all the training data. MLM_S, MLM_U, and MLM_N are three random forest models trained exclusively on stable, unstable, and neutral atmospheric stability class training data, respectively.

Hyperparameters	Range	Optimal value			
		RF	MLM_S	MLM_U	MLM_N
n_estimators	(100,1200)	268	818	414	158
max_features	(“auto”,“sqrt”)	“sqrt”	“sqrt”	“sqrt”	“sqrt”
max_depth	(5,50)	16	36	23	49
min_samples_split	(2,200)	12	10	20	13
min_samples_leaf	(1,10)	2	3	3	2

uses the square root of feature count, and ‘auto’ uses all. The max_depth is the limits node splitting depth in each tree, min_sample_split the minimum samples to split a node and min_sample_leaf is the minimum samples at a leaf node.

3 Experiment and Results

To demonstrate the efficiency and accuracy of the MLM model, we utilized data from the Indianapolis campaign, as discussed in Sect. 2.2.

To improve our machine learning model’s understanding of gas dispersion, we applied a quality index for data cleaning, proposed by [24–26]. This index, based on SF6 observation patterns, ranges from 0 to 3. We only used data rated 2 or 3 and excluded sensor readings beyond 10 km, aligning with the Gaussian model’s local scope. This reduced the dataset from 26,548 to 22,137 entries.

3.1 General Model Vs. Multiple Learning Model

[1] explored the efficiency of statistical learning models, including SVM and neural networks (RBF and BP), against traditional gas dispersion models. They discovered that integrating machine learning with the Gaussian dispersion model was superior. Following this, [10] found that combining SVM with the Gaussian model and MCMC methods enhanced source term estimation. Differently, [11] used the Gaussian model’s G_y and G_z parameters as distinct input features, contrasting [1]’s approach of using the Gaussian model’s output directly. Our research builds upon these findings by deploying three machine learning models tailored to various atmospheric stability conditions, recognizing that gas dispersion behaves differently across stability classes. This method aims to refine the prediction of pollutant gas concentrations, leveraging distinct datasets from comprehensive emission studies (Table 2).

Based on the Monin-Obukhov length, we can classify the experiments into stable, neutral, and unstable classes. Therefore, from our dataset of 170 experiments,

Table 2. Three experiments were randomly selected from the neutral, unstable, and stable datasets, which were used to evaluate the MLM and the random forest model.

	Time of the experiment	Stability class	L_{MO}	Sensors number
Experiment 1	5th of October 1985 at 10 AM	Neutral	−932.0 m	140
Experiment 2	22nd of September 1985 at 4 PM	Unstable	−112.8 m	151
Experiment 3	11th of October 1985 at midnight	Stable	114 m	164

three separate datasets are created: one for the stable class, one for the neutral class, and one for the unstable class. A machine learning model is built for each of these datasets, resulting in a triplet known as the Multiple Learning Model. The threshold values of L_{MO} to identify the three classes are taken from [16].

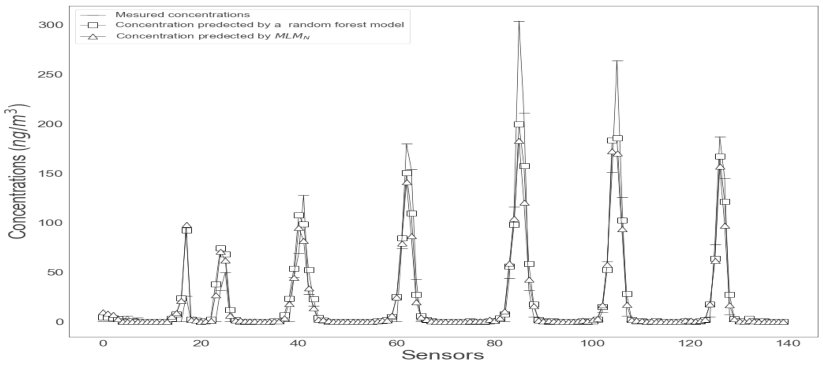


Fig. 5. Concentrations predictions for Experiment 1 were generated using the which is specifically designed for the corresponding stability class of the experiment.

Experiments 1 , 2 and 3 were randomly selected from the unstable, stable and neutral datasets, respectively. These experiments were not included in the model training process. As shown in Figs. 6, 7 and 5, the concentrations predicted by MLM_U and MLM_S and MLM_N exhibit closer agreement with the measured concentrations compared to the random forest model.

We can clearly see that the Multiple Learning Model achieves higher accuracy compared to the random forest model. Table 3 presents various metrics for comparing the two models.

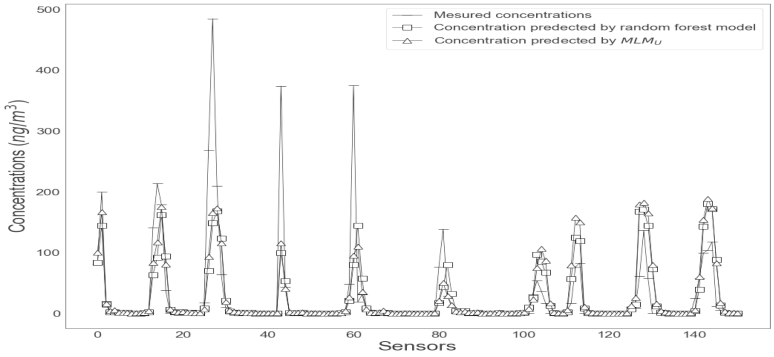


Fig. 6. Predictions of concentrations for Experiment 2 were obtained using the MLM_U model, which was selected based on the stability class of the experiment.

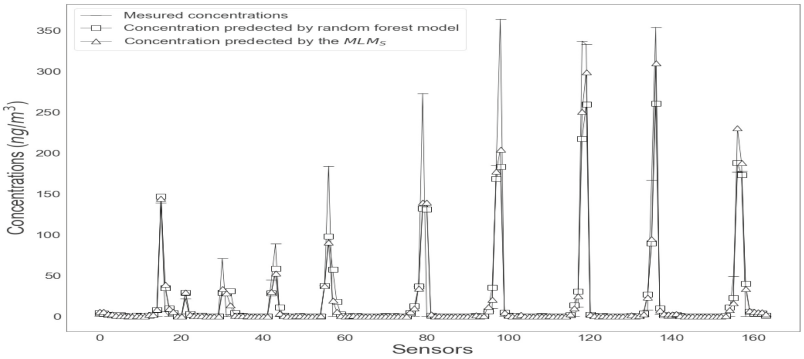


Fig. 7. Predictions of concentrations for Experiment 3 were obtained using the MLM_S model, which was selected based on the stability class of the experiment.

Table 3. Metrics of the two models are presented in this Table. Some of these metrics are obtained from [23]. A perfect model would have MG and R^2 equal to 1.0, FB and NMSE equal to 0.0, and MSE (mean squared error) closer to 0.0.

Metrics	Neutral experiment		Stable experiment		Unstable experiment	
	RF	MLM	RF	MLM	RF	MLM
R^2	0.84	0.90	0.81	0.88	0.44	0.50
FB	0.027	−0.067	−0.089	0.056	0.062	0.037
NMSE	1.18	0.72	1.87	0.073	3.33	3.78
MG	1.15	0.96	1.46	1.17	0.805	0.808
MSE	19.59	16.8	24.85	5.28	51.393	54.017

3.2 Source Identification

The Markov chain, initialized with zeros for all parameters over 10,000 iterations, dedicates the first half to the burn-in phase, assuming non-convergence. Thus, only 5,000 post-burn-in samples from the target distribution are utilized for estimations. Additionally, the bounds for downwind (x) and crosswind (y) distances span 0 to 10,000 and −10,000 to 10,000, respectively (Figs. 8, 9 and 10).

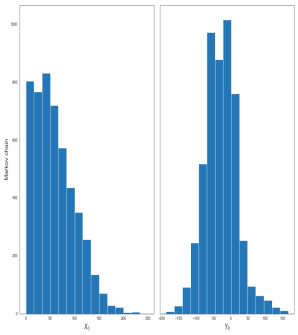


Fig. 8. Results of the MCMC method for Experiment 1, using MLM_N as the forward model.

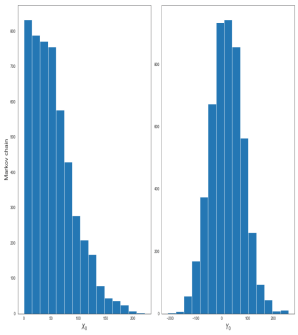


Fig. 9. Results of the MCMC method for Experiment 2, using MLM_U as the forward model.

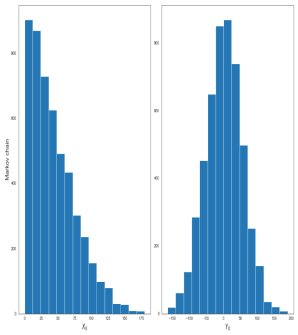


Fig. 10. Results of the MCMC method for Experiment 3, using MLM_S as the forward model.

In those three cases, over 83.60% (more than 8,360) of 10,000 particles were accepted, with the latter half of the chain contributing more than 4,000 samples, accounting for over 81.74% acceptance, to pinpoint the source. By contrast, the Metropolis-Hastings method sees at best a 34.86% acceptance, yielding less reliable results. The DRAM algorithm's superior acceptance rate, facilitated by a second-chance mechanism with an adaptive proposal density, enhances source position estimation accuracy compared to the Metropolis-Hastings algorithm.

Table 4. Estimation results comparing the estimated positions (x , y) obtained from the MLM and random forest models. The Euclidean distance is used to quantify the estimation error between the two models.

	Source position	RF model	MLM	Distance	
				RF	MLM
Neutral case	(9.63 , 22.87)	(27.94 , 51.2)	(15.65 , 28.9)	33.73	8.52
Stable case	(0.14 , 7.83)	(14.35 , 39.15)	(10.65 , 21.34)	34.39	17.11
Unstable case	(29.04 , 7.20)	(44.11 , 7.68)	(26.01 , 13.73)	15.07	7.19

Table 4 showcases comparative estimations from Experiment 2 utilizing the DRAM algorithm (Table 4).

4 Conclusion

To address the complexity of pollutant source identification, a new approach leverages machine learning models based on atmospheric stability, resulting in the Multiple Learning Model (MLM). This method, informed by the Indianapolis dataset, differentiates between stability classes to tailor three models to specific dispersion behaviors, illustrating the impact of atmospheric conditions on gas dispersion.

To assess this method, we chose three experiments from the Indianapolis campaign, each representing a different stability class. The MLM surpassed traditional machine learning in dispersion prediction accuracy and offered source position estimates nearer to the actual locations across all experiments.

Future studies will extend the Gas Dispersion Multiple Learning Models (MLM) to various locations and pollutants, incorporate environmental factors like wind speed and atmospheric conditions for improved accuracy in source identification, and explore real-time MLM applications for immediate response planning.

Acknowledgment. This work was supported by the Laboratory of Mechanics and Energy (LMEE) and the Laboratory of Informatics, Bioinformatics, and Complex Systems (IBISC) at the University of Evry Parsi Saclay, as part of their joint internship program for Master's students.

References

1. Ma, D.L., Zhang, Z.: Contaminant dispersion prediction and source estimation with integrated gaussian-machine learning network model for point source emission in atmosphere. J. Hazard. Mater. **311**, 237–245 (2016). https://doi.org/10.1016/j.jhazmat.2016.03.022
2. Haupt, S.E.: A demonstration of coupled receptor/dispersion modeling with a genetic algorithm. Atmos. Environ. **39**, 7181–7189 (2005). https://doi.org/10.1016/j.atmosenv.2005.08.027
3. Ma, D.L., Deng, J.Q., Zhang, Z.X.: Comparison and improvements of optimization methods for gas emission source identification. Atmos. Environ. **81**, 188–198 (2013). https://doi.org/10.1016/j.atmosenv.2013.09.012
4. Ma, D.L., Gao, J.M., Zhang, Z.X., Wang, Q.S.: An improved firefly algorithm for gas emission source parameter estimation in atmosphere. IEEE Access **7**, 111923–111930 (2019). https://doi.org/10.1109/ACCESS.2019.2935308
5. Ma, D.L., Gao, J.M., Zhang, Z., Zhao, H., Wang, Q.S.: Locating the gas leakage source in the atmosphere using the dispersion wave method. J. Loss Prev. Process Ind. **63**, 104031 (2020). https://doi.org/10.1016/j.jlp.2019.104031
6. Johannesson, G., Hanley, B., Nitao, J.: Dynamic Bayesian models via Monte Carlo-an introduction with examples, Cape Town, South Africa. August 24-29, Lawrence Livermore National Laboratory, UCRL-TR-207173 (2014)

7. Ma, D.L., Tan, W., Zhang, Z.X., Jun, H.: Parameter identification for continuous point emission source based on Tikhonov regularization method coupled with particle swarm optimization algorithm. J. Hazard. Mater. **325**, 239–250 (2017). https://doi.org/10.1016/j.jhazmat.2016.11.071
8. Yee, E., Hoffman, I., Ungar, K.: Bayesian inference for source reconstruction: a real-world application. Int. Sch. Res. Not. **2014**, 507634 (2014). https://doi.org/10.1155/2014/507634
9. Gunatilaka, A., Ristic, B., Skvortsov, A., Morelande, M.: Parameter estimation of a continuous chemical plume source. In: 2008 11th International Conference on Information Fusion, 1–8 (2008)
10. Ma, D.L., Gao, J., Zhang, Z., Zhao, H.: Identifying atmospheric pollutant sources using a machine learning dispersion model and Markov chain Monte Carlo methods. Stoch. Env. Res. Risk Assess. **35**, 271–286 (2021). https://doi.org/10.1007/s00477-021-01973-7
11. Wang, R., et al.: Comparison of machine learning models for hazardous gas dispersion prediction in field cases. Int. J. Environ. Res. Public Health **15**, 1450 (2018). https://doi.org/10.3390/ijerph15071450
12. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001). https://doi.org/10.1023/A:1010933404324
13. Keats, A., Yee, E., Lien, F.S.: Bayesian inference for source determination with applications to a complex urban environment. Atmos. Environ. **41**(3), 465–479 (2007). https://doi.org/10.1016/j.atmosenv.2006.08.044
14. Biau, G., Scornet, E.: A random forest guided tour. TEST **25**(2), 197–227 (2016). https://doi.org/10.1007/s11749-016-0481-7
15. Tarantola, A.: Inverse Problem Theory, and Methods for Model Parameter Estimation. Published by Society for Industrial and Applied Mathematics, Philadelphia (2005)
16. Gryning, S.E., Batchvarova, E., Brümmer, B., Jørgensen, H., Larsen, S.: On the extension of the wind profile over homogeneous terrain beyond the surface boundary layer. Bound.-Layer Meteorol. **124**, 251–268 (2007). https://doi.org/10.1007/s10546-007-9166-9
17. WHO Home Page. https://www.who.int/health-topics/air-pollution#tab=tab1
18. Khan, S., Hassan, Q.: Review of developments in air quality modelling and air quality dispersion models. J. Environ. Eng. Sci. **16**, 1–10 (2021). https://doi.org/10.1680/jenes.20.00004
19. Brioude, J., et al.: The Lagrangian particle dispersion model FLEXPART-WRF version 3.1. Geosci. Model Dev. **6**, 1889–1904 (2013)
20. Juraj, L., L'udovít, J.: CFD based atmospheric dispersion modeling in real urban environments. Chem. Pap. **67**, 1495–1503 (2013). https://doi.org/10.2478/s11696-013-0388-7
21. Haario, H., Laine, M., Mira, A., Saksman, E.: DRAM: efficient adaptive MCMC. Stat. Comput. **16**, 339–354 (2006). https://doi.org/10.1007/s11222-006-9438-0
22. Chang, J., Hanna, S.: Air quality model performance evaluation. Meteorol. Atmos. Phys. **87**, 167–196 (2004). https://doi.org/10.1007/s00703-003-0070-7
23. Hanna, S.R., Hansen, O.R., Dharmavaram, S.: FLACS CFD air quality model performance evaluation with Kit Fox, MUST, Prairie Grass, and EMU observations. Atmos. Environ. **38**, 4675–4687 (2004). https://doi.org/10.1016/j.atmosenv.2004.05.041
24. Hanna, S.R., Chang, J.C.: Hybrid Plume Dispersion Model (HPDM) improvements and testing at three field sites. Atmos. Environ. **27A**, 1491–1508 (1993). https://doi.org/10.1016/0960-1686(93)90135-L

25. Hanna, S.R., Chang, J.C.: Modification of HPDM for Urban Conditions and its Evaluation using the Indianapolis Data Set, Final Report Prepared for EPRI by EARTH TECH, 196 Baker Ave., Concord, MA 10742 (1991)
26. TRC: Urban Power Plant Plume Studies, EPRI Report EA-5468, EPRI, 3412 Hillview Ave, p. 94304. Palo Alto, Ca (1986)
27. Buitinck, L., et al.: API design for machine learning software: experiences from the scikit-learn project. Languages for Data Mining and Machine Learning, ECML PKDD Workshop (2013)
28. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**, 273–297 (1995)
29. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, USA (1975)
30. De Jong, K.: Analysis of the behavior of a class of genetic adaptive systems, Ph.D. thesis, University of Michigan, Ann Arbor, MI, USA (975)
31. https://sklearn-genetic-opt.readthedocs.io/en/stable/
32. Goyal, A., Small, M.J., von Stackelberg, K., Burmistrov, D., Jones, N.: Estimation of fugitive lead emission rates from secondary lead facilities using hierarchical Bayesian models. Environ. Sci. Technol. **39**, 4929–4937 (2005)
33. Gelman, A.G., Roberts, O.G., Gilks, W.R.: Efficient Metropolis jumping rules, Bayesian statistics, pp. 599–609, (Eds J.M Bernardo, J.O Berger, A.F. David and A.F.M Smith), Oxford University Press (1996)

Predictive Maintenance Under Absence of Sensor Data

Ioannis Pierros[1(✉)], Vasileios Kochliaridis[1], Eirini Apostolidou[2], Eleni Delimpasi[3], Vasileios Zygouris[3], and Ioannis Vlahavas[1]

[1] School of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
ipierros@csd.auth.gr
[2] Department of Clinical Pharmacology, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
[3] Aluminium of Greece, MYTILINEOS S.A., 32003 Viotia, Greece

Abstract. In industrial settings, component breakdowns can cause production delays, until repaired or replaced, and incur high costs. To address this issue, many industries have adopted *predictive maintenance*, which is an approach that combines machine learning (ML) and condition monitoring sensors to estimate when the equipment is likely to fail. This allows for early repairs and efficient maintenance scheduling, reducing maintenance costs and downtime. However, installing the necessary sensors can be a significant undertaking for a large company with many industrial machines. To reduce the installation costs and labor required, we investigated an intermediate solution for estimating the remaining useful life (RUL) based only on construction data and their running lifetime. This paper examines how treating RUL estimation as a classification task (i.e. calculating the likelihood of breaking within a period of time instead of its lifespan), increases the volume of available data and allows the employment of ML techniques, which have demonstrated satisfying performance in classification and regression tasks. This method also allows us to integrate additional construction information for each individual component, leading to an increase in the prediction accuracy. Our approach is applied on real-world data from a large production company, forecasting how many smelting pots will malfunction in the near future, resulting in a two-fold increase in accuracy over the company's previous statistical life usage model.

Keywords: Predictive Maintenance · Remaining Useful Life · Machine Learning · Construction data · Classification

1 Introduction

Industrial processes depend on the nominal operation of machinery. An unexpected, failed component can have adverse effects from taking the production

The work was supported by MYTILINEOS S.A.

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 279–292, 2024.
https://doi.org/10.1007/978-3-031-63215-0_21

line out of order, to causing severe physical damage to the machinery and/or environment due to an explosion, fire, or leakage. Consequently, the machine will have to be repaired, leading to time and monetary losses with the potential of introducing significant delays and impacting downstream tasks in the logistical pipeline.

Manufacturers commonly provide a maintenance manual that includes a schedule for performing maintenance in predetermined time frames. Referred to as preventive maintenance, it is a well-rounded approach to avoid failures, though it often leads to unnecessary corrective actions and elevated operating costs [4] and will miss early failures. Life usage models are most commonly used [19], relying on large samples of time-to-failure which is gathered from statistical information of mass-produced components. Afterwards, remaining useful life (RUL) predictions can be made for an individual component, using only the time that the component has been in use. For example, a car manufacturer might suggest replacing the tires of the car every 3 or 5 years.

On the other hand, predictive maintenance (PdM) employs predictive techniques to detect impending failures. By providing early warning of failures, it minimizes machine faults and maintenance costs, reduces spare parts inventory and repair stops while also increases the life of the spare parts and the overall production and profit. Contrary to life usage models, PdM takes into account measured characteristics of the component in question, allowing for more complex analysis and higher accuracy.

Predictive maintenance has emerged in recent years as a promising solution for forecasting failures [24]. This change is driven, in part, from the prevalence of condition monitoring sensors that collect operational data such as temperature, vibration, noise and event logs. Machine learning and deep learning models are especially popular, arising as exceptional candidates that can process and take advantage of the growing amount of industrial data [18].

However, there are cases where condition monitor data are unavailable. This is commonplace with older machinery that wasn't built with such specifications or weren't retroactively fitted with sensors due to time/cost limitations. Additionally, a company might be hesitant to take on such an undertaking, which requires investing in expensive sensors and disrupting the production pipeline to fit the sensors, not to mention training personnel on their appropriate usage and maintenance. There are also cases where a long lead time is required to prepare for the maintenance, such as when spare parts delivery takes a long time or when specialized personnel must be transported to a remote location. In such scenarios where operational data are nonexistent or it is ill-advised to use them due to the huge time-shift, the established approach is to fallback to life usage models that rely on statistical information.

In this paper, we present a hybrid solution between a life usage model and a ML model that would use operational data. We propose modelling the problem as a classification task, classifying each individual component whether it will fail in the following year and then aggregating the results to reach the objective; the prediction of the number of future failures. With our proposed methodology, it

is possible to use ML models efficiently (larger volume of data) and integrate additional information for the individual component, leading to more accurate predictions. Essentially, it is an intermediary solution to allow the use of ML in data sparse scenarios such as the absence of sensor data and to the best of our knowledge it is the first time that it is proposed in these circumstances.

To validate this approach, we present the real case of a production company that must order the necessary materials for the repairs of its smelting pots 6–12 months in advance. Except for the running lifetime of the individual component, we additionally incorporate the specific materials and other construction data related to the repair procedures which are selected with a thorough feature analysis. Operational data are not available. We demonstrate that modelling the problem as a classification task can result in a fourfold increase in prediction accuracy for life usage ML models. We provide comparisons between common ML algorithms to find the best classification model and compare the results with the previous statistical approach that was used, showcasing a 50% reduced error.

Our contributions can be summarized as follows:

- we propose a methodology for modeling RUL problems as classification tasks, allowing the use of ML and the incorporation of supplementary information for the individual component
- we provide a step-by-step analysis for resolving the imbalance problems that arise, due to the small number of faults compared to the total number of running machines
- we present a real-world case study where the application of this methodology achieved a 50% error reduction over the established life usage model

The remainder of the paper is organized as follows. Section 2 discusses recent works of ML applications on RUL prediction, highlighting the absence of appropriate ML solutions without operational data. Section 3 presents our proposed approach for effectively employing ML models in PdM tasks under absence of sensor data, as well as a typical ML pipeline for RUL prediction and some feature selection and imbalanced learning techniques. The case study and the data are described in detail in Sect. 4, while the regression and classification (our) approaches and their results are presented in Sect. 5. Finally, we conclude with a summary of our work and possible future research avenues, which are available in Sect. 6.

2 Related Work

Recently, data-driven predictive (or prognostic) maintenance solutions have largely been dominated by AI techniques, marking a clear shift from statistical methods [16]. Readily available condition monitoring data is the main pillar of approaches that depend on AI/ML algorithms. We found research on the use of AI/ML in other scenarios (without condition monitoring data) to be lacking severely. The majority of research in PdM with ML techniques is focused on data from sensors [13].

Traditional AI algorithms, which do not have enormous data requirements, such as Random Forest (RF) and Support Vector Machine (SVM), have been applied effectively on RUL prediction. Random Forests are an ensemble of weak learners whose predictions are combined to arrive to the final RUL estimate. Li et al. [10] used RFs to predict imminent battery failures in a data center 3 days in advance, while variations of RF have also been developed to aid in the feature selection process [22]. Recently, Alfarizi et al. [2] experimented with employing empirical mode decomposition as a feature extraction step to extract health indices and afterwards predict the RUL of bearings in rotating machinery with an RF model.

Support Vector Machines work by constructing a hyperplane in a high-dimensional space where the original space is mapped to with a kernel transformation, thus making the separation of two classes easier. A variation of SVM for regression has also been applied to PdM, such as in the aerospace industry. Researchers used a hybrid ARIMA-SVM model to predict the RUL of aircraft engines [8] or selected the appropriate kernel parameters for an SVM model with the particle swarm optimization algorithm [17]. Another important aspect is the data preprocessing and feature extraction processes, with Makiaho et al. discussing the use of Pearson correlation as an essential feature selection tool and employing it in their pipeline for predicting the wear of milling blades with an SVM model [12].

Deep learning, a branch of ML that uses multiple non-linear transformation layers to extract high-level features, has recently attracted the interest of researchers, showing great promise in PdM applications. In the study by Garcia et al. [7], an Artificial Neural Network (ANN) was deployed to continuously evaluate the condition of a wind turbine gearbox, which is a crucial component and has high maintenance costs. The researchers trained the ANN to predict the generated power of the wind turbine based on the gearbox and cooling oil temperatures, scheduling maintenance when the model indicated a decrease in generated power below the established level in order to prevent future malfunctions and avoid additional replacement losses. Chen at al. employed a deep neural network to extract features wind, energy conversion-related and temperature sensor data, and used a k-Nearest Neighbors model to detect icing accretion on the blades of wind turbines [6]. Yue et al. opted for a convolutional neural network (CNN) for the feature extraction part of the same problem and classified the health state of the blades [23]. There are many other similar deep learning approaches, such as autoencoders [11], restricted Boltzmann machines and deep belief networks [3].

These works however, require expensive sensors to be installed (also a costly and time-consuming process), that will be able to generate sufficient volume of condition monitoring data for the machine learning algorithms to be effective [16]. As we mentioned before, research on ML use cases without condition monitoring data is lacking. An interesting alternative is to use the inspection and maintenance records [20], though it has its own downsides: records might be incomplete, unavailable or hand-written.

3 Methodology

We present a methodology for employing ML methods for predictive maintenance under absence of sensor data. Specifically, we investigate the case where accurately predicting the RUL of an individual machine is only required to order in advance the necessary materials for its repair or replacement. The established approach are life usage models that create a statistical profile of life expectancy and can subsequently be used to make an RUL prediction for an individual component. Alternatively, a regression model can be trained to predict the total number of faults in a pre-determined, time span, using as inputs the number of operational components and statistical information such as the mean running lifetime.

We propose an intermediary solution, specifically, treating the RUL prediction as a classification problem where we predict if the machine will fail within a predetermined time frame. With this approach, we expand the volume of available data and allow the inclusion of additional high granularity construction information regarding the component. These two benefits are the two driving factors for transforming the RUL prediction to a classification task and we argue that they provide the opportunity to employ ML models with high accuracy.

This approach might seem similar to the classification approach used when condition monitor data is present, where a health index is built that describes the current status of the component based on its running conditions. Health-index and the analogous fault or change-point detection problems are often handled with ML models. However, under absence of sensor data, it actually has a greater resemblance to life usage models where the only changing variable is also the running lifetime of the component. The advantage of our proposed approach over life usage models is that the ML models have a higher capacity of modelling a high-dimensional problem by including multiple explanatory features and achieving better performance.

3.1 Imbalanced Learning

Industrial machines are typically robust and present very few faults, making the collection of data on malfunctions a challenge. Treating RUL as a regression task is challenging as it requires data gathering data over multiple decades, which is the main reason why statistical life usage models are commonly preferred [19]. In a classification approach, this results in imbalanced data sets where the distribution of the two classes (nominal operation/fault) is severely skewed and can lead to biased models that perform poorly in RUL estimation. Over-sampling the minority class (faults) and under-sampling the majority class are common techniques to dealing with imbalanced data sets. We briefly present SMOTE and ADASYN for over-sampling, and NearMiss and TomekLinks for under-sampling.

SMOTE (Synthetic Minority Over-sampling Technique) [5] is a popular over-sampling approach due to its simplicity in design. It creates synthetic data points by interpolating neighboring minority class data points. For each minority class data point, synthetic data points are generated between the member and each

of its k-nearest neighbors. The basic implementation of SMOTE makes no distinction between easy and hard data points to classify, thus it can often lead to over-generalization. Many variants that address this use have been proposed. SVM-SMOTE [15] considers the support vectors of an SVM classifier when generating data points, focusing on the data points of the minority class that reside along the decision boundary. Similarly, ADASYN [9] focuses on data points that have been miss-classified by a KNN classifier.

NearMiss3 [14] is a 2-steps algorithm for under-sampling data points from the majority class using simple heuristics. First, a set is created for each minority class data point with its k-nearest neighbors. Then, N majority class data points with the largest average distance to the data points in the set are removed. TomekLinks [21] is considered a cleaning technique which does not provide control over the number of data points that will remain in each class. It identifies data points of different classes that are the nearest neighbors of each other and removes one of them according to its sampling strategy (majority, in our case).

4 Case Study

In this section we describe a real-world case from the electrolysis field. We provide in-depth details of the available data as well as an analysis of the importance of each feature that helped guide the selection process.

4.1 Electrolysis

Electrolysis is an electrochemical process in which direct electric current is used to force chemical reactions and decompose materials. In metallurgy, it is used for the production of metals such as aluminium, potassium and calcium. This is generally carried out inside big containers, called pots. Construction plans of these pots can vary greatly, even when they are built for the same process.

One of the most crucial catastrophic failures a pot can sustain is to leak. Sometimes, it can be repaired, extending its lifespan, otherwise it must be reconstructed. In both scenarios, the company must source the necessary materials months in advance due to protracted delivery time frames. However, keeping a large inventory of possibly unused materials is inefficient; it comes with storage costs, while materials also deteriorate. It is in the company's best interests to minimize its unused inventory and order only the materials it will use soon.

4.2 Data Set

The data set used in this case study consists of 2888 entries. Each corresponds to a period of operation for a specific pot and includes the start date, end date and various construction features. Each pot can appear in more than one entry, but in a different operation time period. During each period, the pot operates normally, until it experiences a failure or becomes inoperative for business related reasons. After the pot is stopped, it goes under maintenance; any damage is repaired or

a part of the pot might be reconstructed and the pot restarts. There are 5±1 periods per pot, though some can have only 2 or as much as 8 periods.

The data span from 1979 until 2022, including 520 pots in total, which are grouped in two Series. On average, there are 479 pots ($SD = 50$) in operation at any given time and 41 are undergoing repairs or reconstruction ($SD = 50$). Each year there are 63 ($SD = 30$) faults. The average lifespan of a pot is 88 months ($SD = 37$). A histogram of the pots' lifespan is available in Fig. 1a.

The feature that encodes the resolution of pots is named *BPR* and can take 4 categorical values: *repair it (R), reconstruct it (M), repair/reconstruct it for business related reasons (A/P)*. In our application, we are only interested in predicting the malfunctioning pots or the RUL of each pot and not the ones that became inoperative due to business decisions. There were 4 years with many such occurrences, which can be seen in Fig. 1b that depicts the number of pots stopped per year. We excluded such occurrences from the dataset, resulting in 2630 valid entries in the subsequent analysis.

There are 45 features in total, which can be divided into six groups. The 1st group includes four features that uniquely identify the pots and their operational period. The 2nd group includes seven features regarding the contractor who worked on the maintenance/reconstruction of the pots. These features were discarded by the business experts of the company as unrelated to the lifetime of a pot. The 3rd group has seven features of certain chemical components and their concentration. The 4th group includes six features that denote whether certain procedures were applied when repairing/reconstructing the pot. In the 5th group, sixteen features that describe the materials of the pot's casing can be found, including the code of the construction plan. The final group includes five features regarding the cause and details of malfunction of the pots, as well as details about its restart procedure.

Before deciding which features are the most appropriate independent variables for estimating the RUL, it is important to handle missing values. From the group of chemical concentrations, 5 features were discarded because more than 50% of the examples in the data set have missing values. The other 2 features had relatively few (less than 50 samples) and were imputed using the respective mean value of the pot. Similarly, 1 feature was discarded from the *procedures* group with the categorical features. The other 5 features had no missing values.

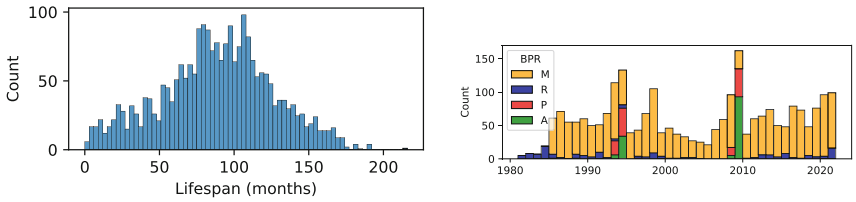


Fig. 1. Frequency histograms of pot lifespan and stops per year

Table 1. Mann-Whitney U test for the categorical features.

Feature	Group	Median	IQR	Mean Rank	Z	p-value
flcaiss	Y	92.5	42	1299.78	−2.172	0.03
	N	89	49	1220.39		
flrecca	Y	96	47	1367.90	−3.553	<0.001
	N	88	47	1215.86		
flrecsi	Y	95	42	1354.15	−19.637	<0.001
	N	47	47	566.34		
flrspot	Y	95	43	1342.50	−11.52	<0.001
	N	75	60	979.96		
flsands	Y	97	43	1379.35	−16.533	<0.001
	N	69	53	844.25		
Previous BPR	P	88	45	1214.71	−2.167	0.03
	F	92	54	1282.01		
Previous BPC	M	94	44	1332.72	−19.34	<0.001
	R	39	39	443.35		

4.3 Feature Selection

To guide the feature selection process, we investigated whether the lifespan for each unique value of the categorical features came from different distributions, in which case the selection of the respective feature is advised.

Descriptive statistics were calculated for all variables. Continuous variables are reported as mean ± standard deviation (SD), or median and interquartile range (IQR) for non-normal data. Assumptions for parametric tests, including normality and homogeneity of variance, were assessed. Normality was checked using the Shapiro-Wilk test and homogeneity of variance was assessed using Levene's test. Expected lifetime was compared between groups using Student's t-test for normally distributed continuous variables or Welch's t-test in case of unequal variances. Mann-Whitney U test was used for non-normal data. Statistical significance was accepted at the two-sided 0.05 alpha level. The results of this analysis are listed in Table 1.

The categorical features *flrecca*, *flsands* and *flrecsi* are deemed to have statistical different lifespans for the Y/N values. For example, flrecsi has a 48-months difference in the median lifespan.

The fault & restart features (BPR and BPC) indicate whether the pot was stopped to repair (R) or reconstruct it (M) and respectively if it leaked (P) or had high concentration of iron (F). Their value correspond to the end of each period, which is clearly unavailable at the time of prediction. Instead, these are shifted down so that we can equate the lifetime of the running period with the BPR and BPC of the previous period.

A few observations can be made from the distributions of pot lifetimes according to the values of the *previous* BPR and BPC codes:

– A repaired pot has significantly lower expected lifetime. In practice, a pot with minor damage can be repaired to extend its lifetime
– Leaks (P) had significantly lower lifespan than high iron concentration (F) in the previous period.

Careful analysis and management of the materials is necessary. There are 15 features that describe the materials that were used for the pot and can take multiple values. In practice, there are 58 materials, 24 of which aren't used anymore, though there might still be merit in using them if the material is expressive enough. To consider a material for inclusion in the model, it must have been used often, otherwise the analysis might not be accurate. The selected materials are those with $p\text{-}value < 0.05$ and a larger than 6 month difference in mean lifetime, namely CE, CF, EE, HC and PI.

Finally, *bagscrout* and *bagssoude* record the amount of solid crushed bath and sodium carbonate. We use the Spearman coefficient to measure the correlation of these features with the expected lifetime. The results suggest a very weak, but statistically significant, positive correlation (0.16 and 0.23 respectively).

5 Experiments and Results

In this section, we evaluate the proposed classification approach for addressing PdM problems that lack sensor data. First, we present the results of the classic regression approach with statistical/ML models. Afterwards, we assess different ML algorithms using our approach and investigate the use of imbalance learning algorithms to alleviate the imbalance data set problem. Finally, we provide a comparison with the established life usage model that is currently being used.

In the regression approach, the goal is to minimize the RUL prediction error, whereas in the classification approach the model predicts if the pot will fail in the following year. The predictions are aggregated to calculate the number of faults in the next year. Both approaches consider the running lifetime, if the pots were repaired and a combination of construction/material info.

The models are evaluated with 5-fold cross-validation during optimization and a variation of backtesting for the final evaluation. Backtesting works as follows: for the years 2015–2021, the models are trained using all available data up to that date and assessed on the year in question. Thus, we can verify the model's performance retrospectively using historical data and ensure a more stable performance. All metrics presented in this section have been calculated with backtesting. Mean squared error (MSE) is reported, which assigns greater weight on larger differentials as they have a larger impact on the production line.

5.1 Evaluation of Regression Approach

The regression approach is evaluated on 3 linear and 4 ML models: Ridge, Huber, ElasticNet, and MLP, Random Forest, KNN and SVM. Ridge minimizes the residual sum of squares, penalized by L2-norm. Huber is similar to Ridge, with

Table 2. Hyper parameters and their search spaces for the regression approach

Algorithm	Hyper parameters	Search space
	alpha	[1e−10, 1e+10]
Huber	epsilon	(1, 100]
Elastic Net	L1 ratio	[0, 1]
Random Forest	estimators	[3, 98]
	min samples to split node	[2, 4]
SVM	kernel	linear, poly, rbf, sigmoid
	C	[0.01, 100]
	loss (linear kernel)	L1, L2
	gamma (non-linear kernel)	auto, scale
	degree (poly kernel)	[3, 11]
MLP	layers	[1, 4]
	units (different for each layer)	[5, 500]
	activation	logistic, tanh, relu

the exception that it applies a linear loss for the samples that it identifies as outliers, while ElasticNet applies both l1 and l2-norm regularization.

The Optuna framework [1] is used to automatically optimize for the hyper parameters that minimize the MSE of predicted number of pots that will fail in the next year. All hyper parameters and their search spaces are listed in Table 2.

Table 3 presents an overview of the best linear and ML models for regression. Elastic Net, which uses both L1 and L2-norm regularization, achieved the best accuracy, though the performance of the three linear models is comparable. Of all the ML models, Random Forest often achieved the lowest errors. On the other hand, only one MLP model managed to score less than 300 MSE, while the best SVR[1] with a linear kernel had a 370 MSE. These outcomes are in-line with the difficulty of the problem and the regression approach, which consists of few training data for the MLP to converge and has a high standard deviation which the linear SVM is unable to model properly. When comparing the linear and ML models, the best models from both categories achieved a similar performance.

5.2 Evaluation of Classification Approach

In the proposed classification approach, we predict separately for each pot whether it will soon fail, as opposed to the regression approach, which attempted to predict the total number of pots that would fail. This allows for input features of high granularity instead of sum or mean values. Additionally, it generates a larger volume of data that can be used to train an ML model adequately.

Specifically, a sample is generated for each pot and for each year, taking into consideration whether the specific pot was built using *flrecsi, flsands* or one of

[1] SVM for regression is called SVR.

Table 3. Best performing linear models (left) and ML models (right) and their hyper-parameters, for the regression approach.

Algorithm	Parameters	MSE
Elastic Net	alpha: 0.221 L1: 0.836	203.71
Huber	alpha: 0.117 ϵ: 6.951	220.14
Ridge	alpha: 0.411	224.00

Algorithm	Parameters	MSE
Random Forest	n_estimators: 3 min_samples_split: 4	201.00
SVR	kernel: rbf C: 100 ϵ: 20.0 gamma: auto	261.43
MLP	activation: tanh neurons: (10,15,15)	270.14
Linear SVR	C: 10 ϵ: 0.0	369.73

Table 4. Hyper parameters and their search spaces for the classification approach

Algorithm	Hyper parameters	Search space
Random Forest	estimators	[5, 2000]
	criterion	gini, entropy
	max depth	5, 10, 15, 20, 25, inf
SVM	kernel	poly, rbf, sigmoid
	C	[0.01, 1000]
	gamma (poly/sigmoid)	auto, scale
	degree (poly)	[3, 11]
KNN	neighbors	[5, 2000]
	weights	uniform, distance
	distance power	[2, 5]
MLP (with Nesterov)	layers and units	[1, 10] and [5, 500]
	activation	logistic, tanh, relu
	L2 and learning rate	[1e-6, 1e-3]
	solver and momentum (SGD)	SGD, Adam and [0.8, 0.9]
	beta1 and beta2	[0.8, 0.99] and [0.9, 0.999]

the available materials and of course its running lifetime. Because the data set is larger, the ML models can theoretically grow in size, allowing Random Forests to have more estimators and MLPs to have more layers and units per layer. The search spaces for each model hyperparamater are available in Table 4.

A significant drawback is that because the pots have a high mean lifetime, the resulting data set is imbalanced, with the non-failure samples making up the majority of the data set. To remedy this issue, we employ a mixture of over-sampling and under-sampling algorithms (see Sect. 3.1).

Table 5. Best performing ML models for the classification approach.

Algorithm	Model Parameters	MSE
Random Forest	estimators: 3 and max depth: 24	52.13
+ *SVM-SMOTE: 50*	criterion: gini	
KNN	neighbors: 5	92.71
+ *SVM-SMOTE: 5*	weights: distance	
+ *TomekLinks*	distance power: 2	
SVM	kernel: rbf	132.57
+ *ADASYN: 20*	C: 0.87	
+ *NearMiss3: 10*	gamma: 1	
MLP	units: (405, 50, 255, 105, 305, 205, 255, 155)	273.86
+ *SVM-SMOTE: 5*	activation: relu and L2: 10e−4	
	learning rate: 0.011187 and solver: adam	
	beta1: 0.95 and beta2: 0.905	

All models incorporated additional features, such as the running lifetime, previously repaired status, addition of bagscrout, the use of flrecsi and flsands, as well as the selected materials. Despite the imbalance issues that arose, switching from regression to classification yielded a twofold increase in prediction accuracy for the SVM and fourfold increase for the RF. The MSE and parameters of the best models of each category are available in Table 5.

The SVM-SMOTE technique was the favoured over-sampling method for all ML models, except for the SVM which worked better with ADASYN. On the other hand, there was no clear preference for the under-sampling method. Notably, while some models with competitive performance did not make use of an under-sampling technique, all models without over-sampling performed much worse, highlighting the necessity of over-sampling in rectifying the evident problems of the imbalanced data set.

Both RF and KNN had <100 MSE, while SVM's error was twice as low as the regression approach. The MLP was the only exception, exhibiting marginally worse accuracy than its equivalent. Overall, the accuracy of the ML models improved drastically, showcasing the benefits of treating the RUL prediction as a classification problem when sensor data is not available. When compared with the company's previously used life usage model, the final ML model achieved a 50% reduction in error.

6 Conclusion

Predictive maintenance can be tremendously valuable in safeguarding the continuous operation of a production line and minimizing supply chain disruptions. In this paper, we sought to address the question of how to improve ML models' RUL

predictions based on construction data and the running lifetime, thereby providing a substitute for circumstances where sensor data are unavailable. We proposed modeling the estimation of RUL as a classification problem, thus increasing the size of the data set and allowing the application of ML. Furthermore, we investigated the import of imbalanced learning techniques for malfunction data sets, where the non-failure samples constitute the majority of the data set. Our methodology was applied on a real-world case study in the field of electrolysis, specifically for the prediction of faults in pots that are used in metallurgy.

The goal of this work was to provide an alternative solution for predictive maintenance for companies that are unwilling to invest in costly condition monitoring sensors that collect operational data or unable due to operational reasons or unavailability of appropriate sensors. When compared to traditional regression models, our results show a twofold to fourfold reduction in prediction error, marking ML algorithms as a promising solution for such scenarios. Additionally, using this methodology we were able to train a model that performed 2 times better than the company's life usage model. In this study, we demonstrated that switching to a classification approach in conjunction with imbalanced learning may offer the requisite amount of data to efficiently train ML models and accurately predict the number of impending failures.

A prominent research direction could be the optimization of the prediction models to maximize the F1 score. Alternatively, one could explore the possibility of improving model accuracy with ensemble techniques.

Acknowledgements. The research was conducted in cooperation with Aluminium of Greece, MYTILINEOS S.A. who provided the data used in this case study as well as valuable feedback while developing the proposed solution.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
2. Alfarizi, M.G., Tajiani, B., Vatn, J., Yin, S.: Optimized random forest model for remaining useful life prediction of experimental bearings. IEEE Trans. Industr. Inf. **19**(6), 7771–7779 (2023)
3. Cao, M., Zhang, T., Wang, J., Liu, Y.: A deep belief network approach to remaining capacity estimation for lithium-ion batteries based on charging process features. J. Energy Storage **48**, 103825 (2022)
4. Carvalho, T.P., Soares, F.A.A.M.N., Vita, R., Francisco, R.d.P., Basto, J.P., Alcalá, S.G.S.: A systematic literature review of machine learning methods applied to predictive maintenance. Comput. Industr. Eng. **137**, 106024 (2019)
5. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
6. Chen, L., Xu, G., Liang, L., Zhang, Q., Zhang, S.: Learning deep representation for blades icing fault detection of wind turbines. In: 2018 IEEE International Conference on Prognostics and Health Management. pp. 1–8. IEEE (2018)

7. Garcia, M.C., Sanz-Bobi, M.A., Del Pico, J.: Simap: intelligent system for predictive maintenance: Application to the health condition monitoring of a windturbine gearbox. Comput. Ind. **57**(6), 552–568 (2006)
8. García Nieto, P., García-Gonzalo, E., Sánchez Lasheras, F., de Cos Juez, F.: Hybrid PSO-SVM-based method for forecasting of the remaining useful life for aircraft engines and evaluation of its reliability. Reliab. Eng. Syst. Saf. **138**, 219–231 (2015)
9. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322–1328. IEEE (2008)
10. Li, X., Pang, A., Yang, W., Zhao, Q.: VRLA battery fault prediction for data center based on random forest model and feature enhancement method. J. Energy Storage **72**, 108666 (2023)
11. Ma, M., Sun, C., Chen, X.: Deep coupling autoencoder for fault diagnosis with multimodal sensory data. IEEE Trans. Industr. Inf. **14**(3), 1137–1145 (2018)
12. Mäkiaho, T., Vainio, H., Koskinen, K.T.: Wear parameter diagnostics of industrial milling machine with support vector regression. Machines **11**(3), 395 (2023)
13. Mallioris, P., Aivazidou, E., Bechtsis, D.: Predictive maintenance in Industry 4.0: a systematic multi-sector mapping. CIRP J. Manuf. Sci. Technol. **50**, 80–103 (2024)
14. Mani, I., Zhang, J.: kNN approach to unbalanced data distributions: a case study involving information extraction. In: Workshop on Learning from Imbalanced Datasets II, ICML (2003)
15. Nguyen, H.M., Cooper, E.W., Kamei, K.: Borderline over-sampling for imbalanced data classification. Int. J. Knowl. Eng. Soft Data Paradigms **3**(1), 4–21 (2011)
16. Ochella, S., Shafiee, M., Dinmohammadi, F.: Artificial intelligence in prognostics and health management of engineering systems. Eng. Appl. Artif. Intell. **108**, 104552 (2022)
17. Ordóñez, C., Sánchez Lasheras, F., Roca-Pardiñas, J., Juez, F.J.d.C.: A hybrid ARIMA-SVM model for the study of the remaining useful life of aircraft engines. J. Comput. Appl. Math. **346**, 184–191 (2019)
18. Pachouly, J., Ahirrao, S., Kotecha, K., Selvachandran, G., Abraham, A.: A systematic literature review on software defect prediction using artificial intelligence: datasets, data validation methods, approaches, and tools. Eng. Appl. Artif. Intell. **111**, 104773 (2022)
19. Schwabacher, M.: A survey of data-driven prognostics. In: Infotech@Aerospace. pp. 1–5. American Institute of Aeronautics and Astronautics (2005)
20. Shiomi, R., Shimasaki, H., Takano, H., Taoka, H.: A study on operating lifetime estimation for electrical components in power grids on the basis of analysis of maintenance records. J. Int. Council Electr. Eng. **9**(1), 45–52 (2019)
21. Tomek, I.: Two modifications of CNN. In: Systems, Man, and Cybernetics. Transactions, vol. 6, pp. 769–772. IEEE (1976)
22. Voronov, S., Jung, D., Frisk, E.: A forest-based algorithm for selecting informative variables using variable depth distribution. Eng. Appl. Artif. Intell. **97**, 104073 (2021)
23. Yue, G., Ping, G., Lanxin, L.: An end-to-End model based on CNN-LSTM for industrial fault diagnosis and prognosis. In: 2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC), pp. 274–278. IEEE (2018)
24. Zonta, T., da Costa, C.A., da Rosa Righi, R., de Lima, M.J., da Trindade, E.S., Li, G.P.: Predictive maintenance in the Industry 4.0: a systematic literature review. Comput. Industr. Eng. **150**, 106889 (2020)

Simulation Study for Evaluating Efficiency of McPhail Traps in Olive Groves

Nikolaos Avgoustis(✉), Evangelos Alvanitopoulos, Nikolaos Marios Polymenakos, Ioannis Karydis, and Markos Avlonitis

Ionian University, Plateia Tsirigoti 7, Corfu, Greece
{avgoustis,alvanitopoulos,polymenakos,karydis,avlon}@ionio.gr

Abstract. Olive tree production has been of paramount importance to nutrition and culture since the early fifth millennium B.C. The most serious pest of olive groves is the olive fruit fly, known as *Bactrocera Oleae or Dacus Oleae*, which can lead to loss of production up to 80–90%. Nowadays, measuring the olive fruit fly's population in the olive groves is key to most pest control strategies. Accordingly, herein, a simulation of olive fruit fly's population dynamics is presented. Initially, the simulation focuses on the correlation between the ratio of captured olive fruit flies in bait traps in relation to the entirety of population (Trap Efficiency). Subsequently, field and its factors were added in the simulation, such as the ratio of olive fruit flies captured in the traps in relation to flies within the trap's attraction area (Capture Rate), crops' variation, and temperature. The simulation's results initially indicated a correlation between *Trap Efficiency* and *Capture Rate* based on previous field experiments, as well as a significant correlation between *Trap Efficiency* and field *Temperature*, using various *Capture Rates*. These results lead towards a contemporary tool for the estimation of olive fruit fly population as well as, by use of regression, the identification of a model that provides trap efficiency estimation for future pests' traps.

Keywords: Bactrocera Oleae · Simulation · Olive fruit · Trap Efficiency · Temperature · Regression · Pest trap modeling

1 Introduction

The olive fruit fly (also known as *Bactrocera Oleae or Dacus Oleae*) presents a major risk to the cultivation of olives and inflicts damage to olives through its reproductive cycle. The female olive fruit fly lays eggs inside olives by using their ovipositors to puncture the fruit's skin and deposit eggs. After hatching, the larvae consume the pulp of the olives as they mature, which ultimately affects the quality and market value of the fruits due to decay and damage [1].

In order to control the population of olive fruit fly, a plethora of methods have been proposed in the literature such as baited, sticky, or pheromone traps, visual inspections, and degree-day models, among others [2]. These methodologies allow for measurement of the olive fruit fly population rather than their

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 293–306, 2024.
https://doi.org/10.1007/978-3-031-63215-0_22

elimination. From the aforementioned various measurements, the cultivators can infer the overall population of olive fruit flies in the field and act accordingly in their application of pest management strategies. Traps are essential in pest management as they offer important information on the presence and spread of olive fruit fly populations in olive groves.

Control methods for managing olive fruit flies' infestations in olive groves involve a range of strategies. These can consist of traditional techniques like removing infested fruits through sanitation, employing insecticides for chemical control, introducing natural predators or parasitoids for biological control, and utilizing traps to decrease olive fruit fly populations [3].

1.1 Motivation and Contribution

Several primary reasons drive the motivation to address the olive fruit fly infestation in olive orchards. Initially, addressing this matter is essential for the agricultural industry, affecting both the volume and quality of olive yields [4–6], which in turn has financial consequences for olive farmers and the broader agricultural sector. Secondly, the non-extensive previous research in this field emphasizes the need for further focused investigation and intervention methods. Accurately measuring the population dynamics of the olive fruit fly is crucial for the development of effective pest control techniques, leading to substantial enhancements in the life cycle and overall health of olive trees. The results of this study have broader implications beyond just olive cultivation, offering insights and solutions that can be transferred to other crops dealing with similar pest problems. Many agricultural pests exhibit similar traits and behaviours, including their reproductive patterns, habitat preferences, and susceptibility to control measures, thus helping extend the impact of the research to various agricultural environments.

The main objective of this research is to create an algorithm that simulates the trap's efficiency [5] in an explicit radius in order to calculate the actual population of olive fruit flies. Thus, the aforementioned algorithm will consist of an useful tool for the agronomists and cultivators, that will pave the way for additional population estimation applications, as this modern tool can be adapted and extended according to the needs of each insect. The major contributions of this paper are presented by the following propositions:

- Design and development of a simulation model for evaluating trap's efficiency considering temperature and various types of cultivation,
- Identification of an optimal model, using regression (as part of AI [7]), that provides trap efficiency estimation for future traps' implementation in Olive Groves.

The remainder of this work is organised as follows: Sect. 2 examines previous research on the olive fruit fly, including its impact on olive trees, methods for managing its population, and the importance of trap precision. Section 3 outlines the suggested methodology, followed by a the methodology's results in Sect. 4, continuing with Sect. 5 discussing the results obtained. Finally, the paper is concluded in Sect. 6.

2 Background

Dacus Oleae (Diptera: Tephritidae), also known as the olive fruit fly, is considered to be one of the main parasites of olive trees mainly around Mediterranean countries. Nevertheless, there are a significant number of records in other regions like America, Asia, and South Africa, that are either lately (year 2018) invading, or have not developed, or have not yet been discovered, as seen on Fig. 1 [8]. According to the work by Kalamatianos et al. [1], the olive fruit fly can cause great damage to the production of olive oil and table olives. Additionally, the olive fruit fly is most active during the summer and the population reaches it's highest levels during autumn. It hibernates during winter and early spring, remaining dormant until the environment becomes suitable for re-emergence [1]. The control measures against the olive fruit fly mainly involve the use of insecticides, which require several applications and highly specific favorable conditions in order to protect the trees [5].

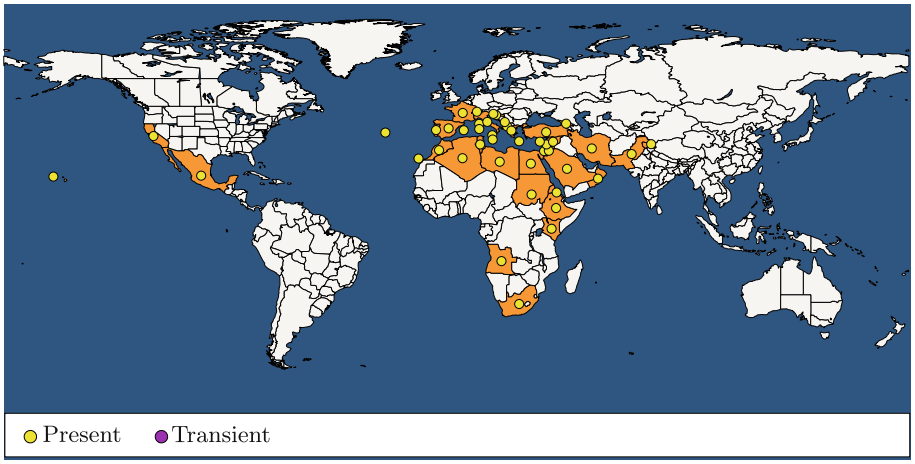


Fig. 1. Olive fruit fly's incidents around the world

As mentioned above, the olive fruit fly causes infestation during its life cycle. Fletcher et al. [9,10] mention that the population growth of olive fruit flies and the extent of infestation in olive groves are affected by wide variety of environmental factors, such as: temperature, humidity, fruit production, olive grove orientation, olive's varieties, spatial spread of olive trees, and interaction between neighbouring microclimates. In any case, the primary factors that affect the activity of the olive fruit fly are temperature [11] and relative humidity [10].

According to the work by Haniotakis et al. [4], the impact of the olive fruit flies can be substantial: without timely control measures, at least 10% of the yield may be lost, and without treatment in olive groves, the loss of production could potentially reach 90% [5]. Other issues include premature fruit shedding prior

to harvest, considered highly severe as it leads to complete loss for cultivators. Another challenge, although minor unless infestation is too extensive, is larval consumption of some pulp, resulting in reduced quantity of olive oil. Additionally, there is a risk of deterioration in olive oil quality in infested fruits due to increased acidity levels [6].

Amr et al., in their work [2], mention that the effect of damage caused by olive fruit fly is influenced by the degree of infestation which can be observed by the presence of exit holes created by the full-grown larvae. These holes damage the fruit skin, exposing it to the atmospheric conditions and other destructive factors like fungi. This results in the acceleration of hydrolytic and oxidative types of rancidity which can be estimated by measuring oil acidity (FFA) and peroxide value (PV).

The mobility of the olive fruit fly is significantly influenced by ambient temperature within the grove, as highlighted in previous studies [9,12]. For instance, research conducted in Greece revealed notable temperature thresholds affecting the flight behaviour of the olive fruit fly. When the average temperature fell below 9 °C, observations indicated a cessation of flight activity among the olive fruit flies. Conversely, as temperatures exceeded 29 °C, adult flies exhibited heightened agitation, with a subsequent loss of motion occurring above 35 °C. These findings underscore the profound impact of temperature fluctuations on the behavioural patterns of the olive fruit fly, with implications for pest management strategies in agricultural settings.

Various methods are available for trapping olive fruit flies and assessing their presence in olive groves. According to the work by Zalom et al. [13], one of the most efficient methodologies to monitor the presence of olive fruit fly is by using McPhail traps [14], with effective bait for the current season.

Traps designed to capture olive fruit flies are crucial tools for monitoring population levels and implementing focused control strategies. According to the work by Doitsidis et al. [15] the accuracy of trap data is vital for efficient pest management, as it influences decisions concerning the timing and extent of control measures. Strategic positioning and effective attraction methods are essential for ensuring that traps consistently attract and capture olive fruit flies. Furthermore, the effectiveness of traps depends on variables such as design, bait appeal, and environmental conditions like temperature and humidity, highlighting the requirement for accuracy and thoroughness in the deployment and maintenance of traps. The McPhail traps have been utilized for more than five decades in various Mediterranean nations and are commonly employed in Greece to monitor the olive fruit fly [16]. The traps are typically baited with different attractants, tailored to the specific location and environmental factors, in order to optimize their efficiency in capturing the insects targeted in olive groves [5].

Additional methodology to trap olive fruit flies, as presented in [5], included placing collection nets under the olive trees and then the application of spray to affect the olive fruit flies population. After allowing some time for the treatment to take effect, counting of the flies occurred in order to estimate the local population, and thus estimate the wider population.

Yet another method to measure olive fruit flies focuses on measuring larvae pupating in the soil [17]. During the initial phase of their life-cycle, the majority of larvae go through pupation inside the fruit. However, as the environmental variables changing according to each season, a varying proportion of larvae exit the fruit and undergo pupation in the soil. To assess the quantity of larvae entering the soil, traps were deployed under olive trees to capture both falling larvae and fruits. These traps were constructed using high-sided rectangular plastic containers filled with water and equipped with drainage holes covered by fine mesh to prevent flooding during rainy periods. After that the fallen larvae were accurately counted and recorded for analysis. Then, to estimate the number of olive fruit flies emerging from the soil after reaching adulthood, emergence traps shaped like pyramids were placed beneath each olive tree. These pyramid-shaped emergence traps were designed based on preliminary studies [17] suggesting that their effectiveness at catching nearly all immature adults as they emerge from the soil.

Recent advancements in machine learning have paved the way for innovative approaches in detecting and counting olive fruit flies [18,19], or even predicting outbreaks as in work of Kalamatianos et al. [20].

3 Proposed Methodology

In [5], Varikou et al. conducted an experiment in 2008 and 2009 on the island of Crete, Greece to monitor the population of fruit olive fly. Ten McPhail traps were set up in an area, along with bait sprays applied throughout the experimentation grove. Infestation levels in olive fruits were monitored, while temperature and relative humidity data were collected from an official weather station near the experimentation site.

The population estimation procedure for olive fruit flies involved using collection nets, applying cover sprays to the trees, and installing traps. Olive fruit flies' counts were conducted at various time intervals after each treatment to minimise overestimation due to fly dispersal.

Six population sampling traps installations were conducted in 2008 and sixteen during the following year. New neighboring trees were chosen for each trap to avoid insecticide residual effects. Data analysis included calculation of the number of dropped adult olive fruit flies from olive trees by trap as well as captures of olive fruit flies.

3.1 Simulation Methodology Approach

With the simulation framework proposed herein, the study endeavors to replicate the behavioural dynamics of olive fruit flies as described in previous literature, thereby facilitating a more accurate assessment of catch rates under varying environmental conditions.

The simulation proceeds through the following steps (see also Fig. 3): Firstly (Fig. 3a), essential variables and simulation parameters are initialised. Subsequently, the simulation environment is constructed based on these parameters,

and initial positions are assigned to each fly. The main simulation loop (Fig. 3b) is then initiated, comprising of several key operations. Within this loop, new random moves and states are generated for each fly. Depending on the presence or absence of attracting trees, a drift towards the tree is incorporated for flies within the tree's radius. Fly's positions are updated accordingly, followed by boundary condition checks. Subsequently, a check for flies within the trap radius is conducted, and captured flies are removed from the simulation. Following completion of the main loop, the *Trap Efficiency* metric is computed and recorded. This systematic approach provides a comprehensive understanding of fly behaviour within the simulation environment.

The core simulation was achieved using the Python programming language due to its widespread usage/popularity and the rich ecosystem of libraries. The simulation is based on stochastic diffusion involving olive fruit flies moving within a pre-defined area, potentially being caught by a McPhail trap positioned at the center of the simulation area. The setting parameters of the simulation include the area's size, time step, *Capture Rate* and the presence or absence of other olive trees acting as attracting entities. In the context of more precise definition, the above parameters are defined in Sect. 3.2. Information kept on individual olive fruit flies focused on their positions and movement behaviour, per each step of the simulation. The simulation progresses through designated steps, encompassing olive fruit flies's movement and the possibility of trap capture based on their positions, capture probabilities and temperature. Results are then analysed to ascertain the captured olive fruit flies's absolute count and percentage in relation to overall population. The simulation provides a versatile framework for researching stochastic diffusion dynamics using Random Walk methods [21] and examining how varying capture probabilities, presence or absence of other attracting olive trees, and temperature impact capture rates.

The simulations begin with the assumption of a uniform distribution of olive fruit flies within the 20-meter radius surrounding the trap (see Fig. 2a). The first set of simulations are grounded on empirical observations, proposing that olive fruit flies allocate their time primarily to walking, with intermittent periods of flight and rest. In each iteration of the simulation, individual flies have the capability to move in one of eight directions of North, Northeast, East, Southeast, South, Southwest, West, Northwest with their respective coordinates relative to the individual fly's location being (0, 1), (1, 1), (1, 0), (1, −1), (0, −1), (−1, −1), (−1, 0), (−1, 1) and with the distance travelled determined by predefined probabilities based on field observations [6,22,23]: there is a 20% chance of flight for a distance of 1 m, a 20% chance of remaining stationary, and a 60% chance of walking a distance of 0.05 m. Subsequently, four attracting trees are strategically positioned near each corner of the experimentation space, effectively integrating a drift factor into the movement dynamics of the olive fruit fly (see Fig. 2b). This approach allows for the exploration the nuanced influence of proximity to these trees on the flies' directional movement, contributing to a comprehensive understanding of pest behaviour in agricultural ecosystems. In the second set of simulations the assumption that the movement of the olive fruit flies depends on

the temperature is introduced and the probabilities of flight, walking or remaining stationary are affected by the weekly temperature following the formalisation shown in sequel:

$$P(fly) = 1 - f(t) \tag{1}$$

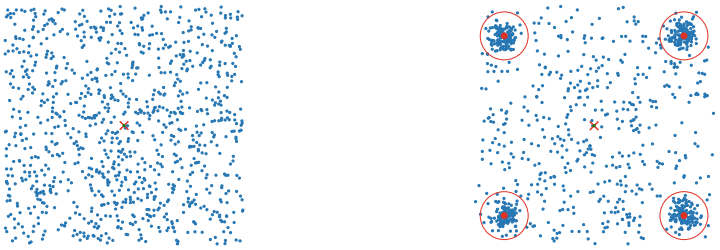
$$P(walk) = 1 - f(t) \tag{2}$$

$$P(stationary) = 1 - P(fly) - P(walk) \tag{3}$$

where $f(t)$ is a logistic equation with input the mean weekly temperature and

$$f(t) = \frac{1}{1 + e^{-k(t-25)}} \tag{4}$$

The use of the logistic function aims to represent a threshold behaviour, where there is a critical temperature range within which certain biological processes or events occur. For example, in the case of insect activity, there might be a temperature range between 20 and 32 °C where the likelihood of a specific behaviour (e.g., flight activity) is significantly higher compared to temperatures outside this range.



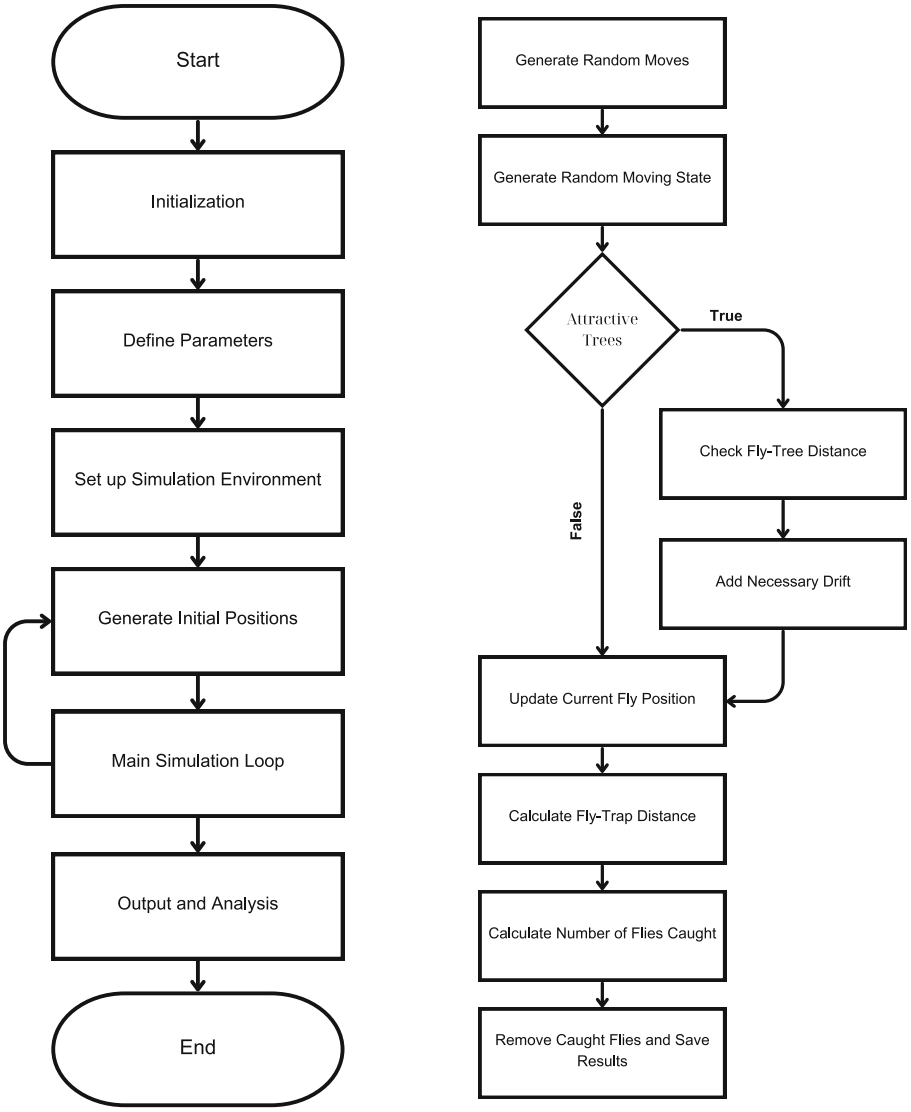
(a) (b)

Fig. 2. Olive fruit fly distribution with the presence (a) and absence (b) of attractive trees

As per the results of the studies [5,6] and to retain comparable results, each simulation corresponds to a one-week time-frame. Within each week, the simulation is executed ten times, and the average catch rate is computed based on the results of these iterations with the purpose to address the stochasticity introduced in numerous parameters of the simulation, like the random movement of the fruit fly and the different number of the trapped olive fruit flies on each simulation run.

3.2 Inputs and Outputs of the Model

The methodology employed herein draws upon the findings of Varikou et al. [5] as a foundational basis and integrates insights from earlier works by Economopoulos et al. [12] and Fletcher et al. [9] to construct two distinct models.



(a) Diagram of the simulation process (b) Diagram of the main simulation loop

Fig. 3. Architecture diagram of the simulation model

The first model is a linear model which correlates trap efficiency with capture rate, building upon a theoretical framework calibrated against empirical data. This framework posits that a McPhail trap captures approximately 0.5% of the actual olive fruit fly population within a 20-m radius.

The second model extends this approach by integrating mean weekly temperature data from the grove. It adjusts the probabilities of olive fruit fly movement

based on the observed trap's efficiencies and corresponding temperature conditions. A polynomial model is then fitted to derive a theoretical equation for calculating trap efficiency considering both capture rate and mean temperature.

The simulation framework was developed using Python, leveraging additional input variables to enhance algorithmic accuracy. These variables include Radius, Step Time, Duration Day, Fly Size, Population, Capture Rate, Fly Speed, and Temperature as outlined in Table 1 and further detailed in the sequel.

As mentioned before, Radius is defined as the efficient radius of the trap, accordingly to [5]. For simulation performance reasons *Step Time* is set to 3 s, and refers to the time that passes between two movements of olive fruit flies. For optimal performance, *Duration Day* is considered as the duration of one day in the simulation, since at night the olive fruit fly is considered not as active as the day. The *Fly Size* is the actual size of the olive fruit fly in the simulation. The *Population* is the number of flies initialised. *Capture Rate* (CR) is defined as the probability of the fly to be captured after being in close proximity or above the trap, and is considered to be from 0 to 100%. *Fly Speed* of the flies is set as three different values depending on the states of the movement of the fly. While flies are flying speed is considered to be as 1, walking is equal to 0.05 and stationary is equal to 0 [6,22,23]. Finally, the last input variable is the mean weekly *Temperature* with integer values in range of [20, 32] °C per week, following the methodology of [9,12].

The results are consist of the outputs shown in Table 2: *Captured Flies*, *Trap Efficiency*, and *Regression Models*. The *Captured Flies* are the unit of *Population* that finally trapped during the simulation. The terminology *Trap Efficiency* (TE), first discussed in [9,11], is set as the percentage of population that finally was trapped. Lastly, the *Regression Models* (Linear and Polynomial) are expressed as equations.

Through this integrated approach, the study aims to provide a comprehensive understanding of olive fruit fly behaviour in relation to *Trap Efficiency* and Temperature dynamics, facilitating more effective pest management strategies.

Table 1. Inputs that used for simulation

Inputs	Value	Variable unit
Radius	20	m
Step Time	3	s
Duration Day	12	h
Fly Size	1×1	px
Population	1000	num
Capture Rate	0.1–1	%
Fly Speed	1, 0.05, 0	m/s
Temperature	20–32	mean °C/week

Table 2. Outputs of the simulation

Outputs	Variable unit
Captured Flies	Individuals per trap
Trap Efficiency	%
Regression Models	Equations

4 Results

The first model establishes a correlation between Trap Efficiency and Capture Rate of a fly being captured upon crossing the trap in two distinct scenarios: initially the impact of tree distribution within a 20-m radius is explored in Eq. 5, contrasting instances where the trees exhibit homogeneity and where they do not, as seen in Eq. 6. Utilising as linear regression approach, the model is refined to achieve the best fit to the simulated data. The resulting equations for the best fit lines for the simulation with and without trees are displayed below, along with its corresponding coefficient of determination (R-squared value):

$$TE = 0.0479CR + 0.4435 \tag{5}$$

With

$$R^2 = 0.9842$$

$$TE = 0.0177CR + 0.1518 \tag{6}$$

With

$$R^2 = 0.9620$$

As expected, the resulting line from the simulation with homogeneity exhibits a steeper slope compared to the scenario without homogeneity, as illustrated in Fig. 4. This discrepancy suggests that in an orchard with varying types of olive trees, the *Trap Efficiency* metric may not accurately represent the actual *Capture Rate* and subsequent control of olive fruit flies. The steeper slope implies that the presence of different olive tree varieties could significantly influence the effectiveness of trapping methods. This phenomenon may occur due to the diversion of flies towards more attractive trees, diminishing their presence in the traps. Hence, accounting for tree heterogeneity is crucial for refining pest management strategies in agricultural settings.

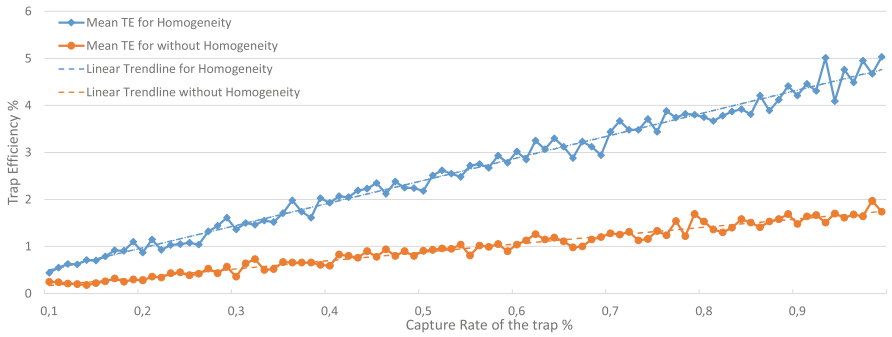


Fig. 4. Correlation of Trap Efficiency and Capture Rate.

The second model explores the relationship between *Trap Efficiency* and the average weekly temperature. Employing polynomial regression analysis, the model is optimised to provide the most accurate representation of the observed data. Fitted equations for selected *Capture Rates*, along with its corresponding coefficient of determination (R-squared value), are:

For Capture Rate of 0.2%

$$TE = 0.0023T^3 - 0.0528T^2 + 0.2558T + 0.6606 \tag{7}$$

With

$$R^2 = 0.8973$$

For Capture Rate of 0.5%

$$TE = 0.0032T^3 - 0.0717T^2 + 0.2314T + 2.1571 \tag{8}$$

With

$$R^2 = 0.9863$$

For Capture Rate of 0.8%

$$TE = 0.0077T^3 - 0.1717T^2 + 0.6777T + 3.326 \tag{9}$$

With

$$R^2 = 0.9906$$

The simulation results depict a marginal increase in *Trap Efficiency* followed by a decline as temperature rises across each of the distinct *Capture Rates* as shown in Fig. 5.

By deriving these models from the simulation results, the study provides insights into the factors influencing *Trap Efficiency* of the McPhail trap, thus contributing to the refinement of pest management strategies in agricultural ecosystems. This offers a clearer understanding of the intricate interplay between *Trap Efficiency*, environmental temperature, and non uniform terrain. Moreover, by elucidating the complex relationships between these variables, the study enhances our ability to predict and mitigate pest infestations effectively.

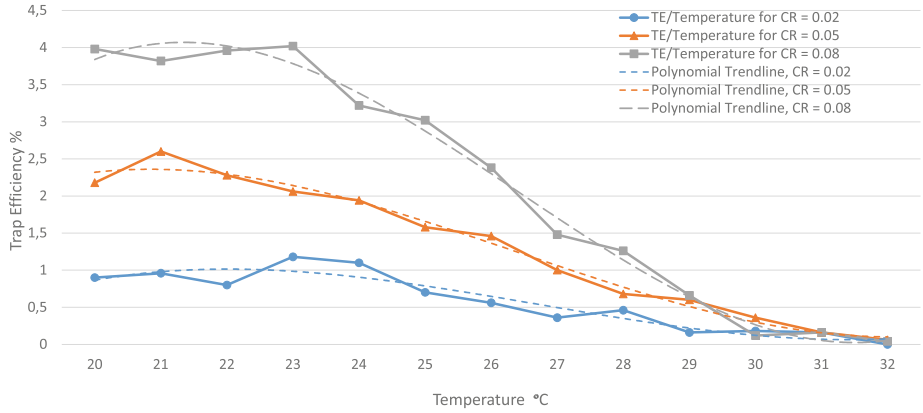


Fig. 5. Correlation of Trap Efficiency and Temperature.

5 Discussion

Based on the findings of the suggested methodology, it can be inferred that when a specific quantity of olive fruit flies is captured by the McPhail trap on olive trees, then the population in the orchard approximates to a number of individuals. This data is valuable for developing a strategy to manage the population of olive fruit flies and mitigate their infestation on olive trees.

By integrating temperature data into our simulation models, it was possible to develop predictive equations that accurately reflect the relationship between *Trap Efficiency* and *Temperature* variations, offering valuable insights for precision pest management.

Furthermore, the analysis of *Trap Efficiency* provides interesting insights regarding the efficacy of trapping methods for estimating olive fruit fly population. By calibrating simulation models against empirical data, it is able to derive regression equations that are able to predict *Trap Efficiency* based on *Capture Rates*. These models offer a practical tool for agronomists and trap manufacturers to improve trap design and positioning, thereby enhancing the effectiveness of pest monitoring and control endeavors.

Overall, this study contributes to the growing body of research aimed at improving pest management practices in olive groves. By leveraging advanced simulation techniques and mathematical modeling, it provides a deeper understanding of the complex interactions between environmental factors, trap efficiency, and olive fruit fly behaviour in non uniform field patterns.

6 Conclusion

In this study, a comprehensive analysis of olive fruit fly (*Bactrocera Oleae*) population dynamics had been presented, focusing on the correlation between *Trap*

efficiency and field factors such as *Capture Rate*, various crops, and *Temperature*. Through a combination of empirical observations, mathematical modeling, and simulation techniques, it was aimed to shed light the underlying factors influencing olive fruit fly behaviour and their implications for pest management strategies in olive groves.

The results of the first simulation concluded in correlation between *Trap Efficiency* and *Capture Rate* based on previous experiments. On the second simulation the results showed that there is a significant correlation between *Trap Efficiency* and *Temperature*, using various *Capture Rates*, leading towards a contemporary tool for the estimation of olive fruit fly population.

Future research could include the development of more sophisticated simulation models and the integration of additional parameters to further refine pest management strategies in agricultural settings. Moreover, this study has the potential to be extended in the future by leveraging techniques such as Spatial Skyline Queries [24,25], where different clusters based on trees' attractions, olive fruit flies behaviour, and microclimates can be identified. Finally, the results and methods could be transferred on other kind of crops and pests.

Acknowledgment. The authors gratefully acknowledge the financial support of the Ionian University, Greece.

References

1. Kalamatianos, R., Kermanidis, K., Avlonitis, M., Karydis, I.: Environmental impact on predicting olive fruit fly population using trap measurements. In: Iliadis, L., Maglogiannis, I. (eds.) AIAI 2016. IAICT, vol. 475, pp. 180–190. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44944-9_16
2. Amr, A., Sadder, M., Sakarneh, N., et al.: Review article olive fruit fly bacterocera oleae infestation of olives: effect on quality and detection in olive oil. Jordan J. Agric. Sci. **19**(1), 81–94 (2023)
3. Haniotakis, G., Kozyrakis, M., Fitsakis, T.H., Antonidakj, A.: An effective mass trapping method for the control of dacus oleae (diptera: Tephritidae). J. Econ. Entomol. **84**(2), 564–569 (1991)
4. Haniotakis, G.E., et al.: Olive pest control: present status and prospects. IOBC WPRS Bull. **28**(9), 1 (2005)
5. Varikou, K., Alexandrakis, V., Gika, V., Birouraki, A., Marnelakis, Ch., Sergentani, C.: Estimation of fly population density of bactrocera oleae in olive groves of Crete. Phytoparasitica **41**(1), 105–111 (2013)
6. Kapatos, E.T., Fletcher, B.S.: An assessment of components of crop loss due to infestation by dacus oleae, in corfu. Entomologia Hellenica **1**, 7–16 (1983)
7. Kesavaraj, G., Sukumaran, S.: A study on classification techniques in data mining. In: 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), pp. 1–7 (2013)
8. [World distribution]| EPPO Global Database: Bactrocera oleae (dacuol), 1989–2023 (n.d.)
9. Fletcher, B.S.: Temperature-development rate relationships of the immature stages and adults of tephritid fruit flies. In: Fruit Flies Their Biology, Natural Enemies and Control, vol. 3, pp. 273–289 (1989)

10. Broufas, G.D., Pappas, M.L., Koveos, D.S.: Effect of relative humidity on longevity, ovarian maturation, and egg production in the olive fruit fly (diptera: Tephritidae). Ann. Entomol. Soc. Am. **102**(1), 70–75 (2009)
11. Kapatos, E., Fletcher, B.S.: Seasonal changes in the efficiency of McPhail traps and a model for estimating olive fly densities from trap catches using temperature data. Entomol. Exp. Appl. **33**(1), 20–26 (1983)
12. Economopoulos, A.P., et al.: Population studies on the olive fruit fly, dacus oleae (gmel.) (dipt., tephritidae) in Western Crete. Zeitschrift für Angewandte Entomologie **93**(1–5), 463–476 (1982)
13. University of California Agriculture & Natural Resources: Olive fruit fly management guidelines, Revised 2/09
14. Steyskal, G.C.: History and use of the McPhail trap. Florida Entomol. 11–16 (1977)
15. Doitsidis, L., et al.: Remote monitoring of the bactrocera oleae (gmelin) (diptera: Tephritidae) population using an automated McPhail trap. Comput. Electron. Agric. **137**, 69–78 (2017)
16. Haniotakis, G.E., Skyrianos, G.: Attraction of the olive fruit fly to pheromone, McPhail, and color traps. J. Econ. Entomol. **74**(1), 58–60 (1981)
17. Kapatos, E.T., Fletcher, B.S.: The phenology of the olive fly, dacus oleae (gmel.) (diptera, tephritidae), in corfu. Zeitschrift für Angewandte Entomologie **97**(1–5), 360–370 (1984)
18. Mamdouh, N., Khattab, A.: Yolo-based deep learning framework for olive fruit fly detection and counting. IEEE Access **9**, 84252–84262 (2021)
19. Molina-Rotger, M., Morán, A., Miranda, M.Á., Alorda, B.: Remote fruit fly detection using computer vision and machine learning-based electronic trap. Front. Plant Sci. **14** (2023)
20. Kalamatianos, R., Kermanidis, K.L., Karydis, I., Avlonitis, M.: Treating stochasticity of olive-fruit fly's outbreaks via machine learning algorithms. Neurocomputing (Amsterdam) **280**, 135–146 (2018)
21. Pearson, K.: The problem of the random walk. Nature **72**, 294 (1905)
22. Sexton, C.: Fruit flies can travel six million times their body length. Earth.com, April 2021
23. Leitch, K.J., Ponce, F.V., Dickson, W., Van Breugel, F., Dickinson, M.H.: The long-distance flight behavior of drosophila supports an agent-based model for wind-assisted dispersal in insects. Proc. Natl. Acad. Sci. USA **118**(17) (2021)
24. Borzsony, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings 17th International Conference on Data Engineering, pp. 421–430. IEEE (2001)
25. Bavirthi, S.S., Supreethi, K.P.: Systematic review of indexing spatial skyline queries for decision support. Int. J. Decis. Support Syst. Technol. (IJDSST) **14**(1), 1–15 (2022)

SMT: Self-supervised Approach for Multiple Animal Detection and Tracking

Muhammad Moosa[1], Muhammad Mudassar Yamin[1], Ehtesham Hashmi[1], Azeddine Beghdadi[2], Ali Shariq Imran[1], Faouzi Alaya Cheikh[1], and Mohib Ullah[1(✉)]

[1] Norwegian University of Science and Technology, 2815 Gjøvik, Norway
mohib.ullah@ntnu.no

[2] L2TI, Institut Galilée, Université Sorbonne Paris Nord, Paris, France

Abstract. In the domain of animal farming and wildlife management, monitoring animal behavior and movement is crucial. This paper proposes an efficient online multi-object tracking framework named SMT (Self-supervised Multi-animal detection and Tracking) for a dynamic and complex environment. The framework is based on the tracking-by-detection approach and builds on the idea of employing self-supervised object detection and a bag of Bayesian trackers. We collected and annotated a custom dataset from an animal farm for training and validating the detection and tracking algorithms. Additionally, we utilized the public dataset Dancetrack to benchmark and compare the results against reference methods. The comparison with reference methods reveals substantial enhancements on standard tracking metrics, such as IDF1 and MOTA. The optimized combination of the self-supervised object detector and proposed tracker demonstrates robust performance by consistently preserving object identities and reducing identification errors throughout sequences. To reproduce the results, we made the code publically available at https://github.com/moosa1296/effdet_ocsort.

Keywords: Self-supervised learning · Bayesian tracker · Behaviour analysis

1 Introduction

Multi-animal tracking (MAT) plays a crucial role in selective breeding by providing valuable insights into the behavior [1], interactions, and characteristics of multiple animals [2] simultaneously. In the context of selective breeding, MAT helps to monitor and analyze the movement patterns, social dynamics, and individual traits of a group of animals that carry great interest for the breeders [3]. MAT allows the collection of detailed data on various parameters, such as growth rates, feeding habits, and reproductive behaviors [4]. By employing MTT, scientists can gain a comprehensive understanding of the genetic and environmental factors influencing the desired traits in a breeding program [5]. The ability

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 307–317, 2024.
https://doi.org/10.1007/978-3-031-63215-0_23

to track multiple targets simultaneously enhances the precision and efficiency of data collection, providing breeders with a more robust foundation for making informed decisions in their efforts to selectively breed animals with specific desirable traits. This paper proposes a self-supervised tracking framework named SMT that is based on the tracking-by-detection strategy to track multiple animals in a pig farm. The algorithm leverages the EfficientDet detector [6] to detect multiple animals in frames. The detector is trained on both the labeled and unlabelled data using self-supervised learning. The detected animals in frames are associated temporally using the Observation-Centric SORT (OCSort) algorithm [7]. The detector and tracker are optimized to track the animals robustly in indoor farm settings. The tracker uses animal observations to compute a virtual trajectory, which fixes the error accumulation of filter parameters during the occlusion period [7]. In a nutshell, the contributions of our work is three-fold:

- We trained the Object Detection model with self-supervised learning using both labeled and unlabeled data.
- The tracker leverages animal observations by calculating virtual trajectories that help to mitigate the error accumulation of tracker parameters, particularly during occlusion periods.
- To validate our algorithm, we collected a custom dataset in an animal farm. The dataset is annotated for detection and tracking.

The rest of the paper is organized in the following order. Related work is discussed in Sect. 2. An overview of the proposed algorithm including the self-supervised detection and the tracker is given in Sect. 3. Section 4 describes the datasets and the evaluation metrics. The implementation details and quantitative and qualitative results are given in Sect. 5. The final remarks are given in Sects. 6 that conclude the paper.

2 Related Works

In recent years, significant progress has been achieved in the field of computer vision at large, with particular strides made in the domain of Multi-Object Tracking (MOT) algorithms. Several algorithms have demonstrated commendable accuracy and robustness across diverse and challenging conditions. Table 1 provides a comparative analysis of six recent MOT algorithms and presents their positive aspects in terms of several key criteria like simplicity, real-time performance, resilience to occlusion, and robustness to non-linear motion. Simplicity refers to how easy the algorithm is to implement and use. Real-time performance assesses the capability of an algorithm to track objects at a speed suitable for practical applications in the real world. Some literature [13] also uses the term "real-time tracking" to denote online tracking, wherein the algorithm can continuously track objects in a dynamic setting where new data is consistently received enabling context-based decision for tracking [14]. Robustness to occlusion refers to how well the algorithm can track objects that are occluded by other objects or by the background scene. Robustness to non-linear motion refers to how well

Table 1. Comparison of Recent State-of-the-Art MOT Algorithms: Positive Aspects in Simplicity, Real-time Performance, Occlusion Resilience, and Non-linear Motion Robustness.

Algorithm	Simple	Real-time	Robust to occlusion	Robust to non-linear motion
OC-SORT [7]	Yes	Yes	Yes	Yes
SORT [8]	Yes	Yes	No	No
DeepSORT [9]	No	Yes	Yes	Yes
FairMOT [10]	No	Yes	Yes	Yes
ByteTrack [11]	Yes	Yes	Yes	Yes
CenterTrack [12]	No	Yes	Yes	Yes

the algorithm can track objects that are moving in a non-linear fashion, such as objects that are accelerating or decelerating. We can see OC-SORT achieves state-of-the-art performance on multiple MOT benchmarks including MOT17, MOT20, KITTI, and especially DanceTrack, where the object motion is highly non-linear. Each tracking model mentioned in Table 1 has specific challenges that impact its performance and applicability to different tracking scenarios. OC-SORT [7] relies heavily on the performance of the underlying object detector. The accuracy of OC-SORT's tracking results is directly tied to how well the object detector can identify and locate objects within the scene, which means that any limitations or inaccuracies in the object detector can significantly affect the overall tracking effectiveness of OC-SORT. Similarly, SORT [8] faces challenges related to noise in state estimations and the accumulation of errors over time. SORT [8] is particularly sensitive to inaccuracies in the predicted states of objects, which can lead to reducing tracking performance as the sequence progresses. Additionally, SORT struggles with tracking objects that exhibit non-linear motion, making it less robust in dynamic environments where objects may change direction or speed unpredictably [15]. DeepSORT [9] and FairMOT [10] are more computationally expensive and require a significant amount of training data to achieve effective performance. These requirements can limit their applicability in scenarios where computational resources are constrained, especially in real-time applications or when dealing with limited datasets. Similar to OC-SORT [7], ByteTrack's [11] effectiveness is closely linked to the object detector's ability to accurately identify and locate objects. This dependence means that ByteTrack's performance can be significantly impacted by the quality of the object detection. Furthermore, ByteTrack [11] and OC-SORT's [7] negative aspects are unique in that it is not related to the algorithm itself, but rather to the object detector that it uses. This means that the tracking results of these models can be improved by using a better object detector [16].

3 Methodology

The block diagram of our proposed algorithm is illustrated in Fig. 1. The key components are the self-supervised object detector and the tracker. In the following subsection, details of each component are elaborated.

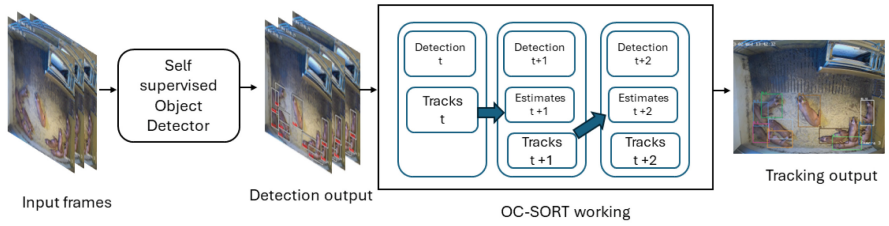


Fig. 1. Proposed algorithm workflow: a self-supervised object detector for precise animal localization, and a tracker (OC-SORT variant) integrated with detection outputs. Modified data pre-processing transforms detections (D_t) into trackable format (F_t) for seamless integration

3.1 Self-supervised Object Detector

The implementation of the self-supervised object detector forms the first basic component of our tracking system. We use EffecientDet to accurately identify and localize objects (pigs) within each frame. The detection output at any given time t is represented as a set of detections D_t, with each detection $d_i \in D_t$ detailed as $d_i = (b_i, p_i, c_i)$. In this formulation, b_i delineates the bounding box, p_i signifies the confidence score, and c_i the class label:

$$d_i = (b_i, p_i, c_i) \tag{1}$$

To extract features from the input frames, we used a variant of the EfficientNet architecture as a backbone network as shown in Fig. 2. The input frame is processed through a series of convolutional layers to create a rich feature map that captures various aspects of the frame at different scales and complexities [17]. A BiFPN, a type of feature pyramid network that allows easy and fast multi-scale feature fusion [6], is employed on the top of the backbone which enables easy and fast multiscale feature fusion. The BiFPN takes the feature maps from different levels of the backbone and combines them, enhancing the network's ability to detect objects of various sizes. It uses separate heads for predicting bounding boxes and class labels. Each head consists of several convolutional layers and outputs predictions at different scales, corresponding to different levels of the feature pyramid [6]. This multi-scale prediction is crucial for detecting objects

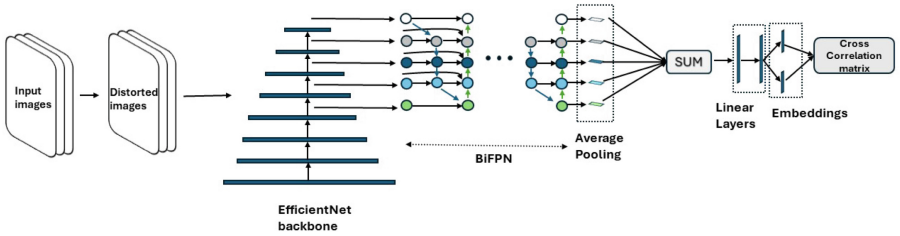


Fig. 2. Self-supervised object detector: The backbone processes the input frame, creating a feature map capturing diverse aspects at different scales. A BiFPN facilitates efficient multi-scale feature fusion, enhancing object detection across various sizes. Separate heads predict bounding boxes and class labels at different scales.

of different sizes. In our work, instead of directly training the object detection model using labeled data in a fully supervised learning fashion, we first pre-train the model on unlabeled data. For that, we used the Barlow Twins [18] strategy. However, unlike the traditional approach where the method is applied directly to the detector's backbone, here we modify it by applying it to a condensed representation of the output of the Feature Pyramid Network (FPN). The goal of this method is to train the model to learn embeddings that remain consistent even when the input data is distorted. It aims to find a representation of the data that captures its essential information while being less affected by specific distortions applied to the frames. During pre-training, the representations obtained from the BiFPN (bi-directional feature pyramid network) are each processed through an average pooling layer. These pooled representations are then combined and passed through two linear layers to generate the final embedding. These embeddings are then used to compute a cross-correlation matrix. The self detection based architecture [19] is shown in Fig. 2 The loss function employed in pre-training computes the cross-correlation matrix among the embeddings generated by the network for every distortion applied to the data. This process aims to measure the similarity between the embeddings across different distortions, thereby facilitating the learning of invariant representations that capture essential information about the data while being robust to variations caused by distortions, given by Eq. 2.

$$L_{bt} = \sum_{i}(1 - C_{ii})^2 + \lambda \sum_{i} \sum_{j \neq i} C_{ij}^2 \tag{2}$$

In Eq. 2, λ represents a constant used to balance the significance of the two loss terms. C denotes the cross-correlation matrix, with dimensions matching those of the output from the projection layer. The cross-correlation matrix between embeddings A and B, where b indexes samples and i, j indexes the embeddings produced by z, as given by Eq. 3.

$$C_{ij} = \frac{\sum_b z_{b,i}^A z_{b,j}^B}{\sqrt{\sum_b (z_{b,i}^A)^2} \sqrt{\sum_b (z_{b,j}^B)^2}} \tag{3}$$

3.2 Proposed Tracker

We adopted a variant of OC-SORT and integrated it with the output results we got from our detection model. The integration process involved ensuring that the bounding box outputs from EfficientDet were compatible with the OC-SORT tracking module. This necessitates a modified data preprocessing stage, where detection outputs D_t are transformed into F_t, a trackable format. An instant of tracker can track one object at a time. Therefore, for N animals, we used N instant of the tracker. With the detections suitably preprocessed, we employ the tracker for the sequential tracking of animals across frames, which is summarized in the tracking function:

$$T_t = \text{OC-SORT}(T_{t-1}, F_t; \Theta) \tag{4}$$

Here, T_{t-1} represents tracks from the preceding frame, F_t represents the formatted detections for the current frame, and Θ shows the set of finely tuned parameters. The new aspects of the integration are our fine-tuning to the OC-SORT parameters (θ) and our exploration of other metrics (IoU, GIoU, and DIoU) for bounding box comparison. OC-SORT's tracking mechanism is primarily based on SORT (Simple Online and Real-time Tracking) [7]. It is fine-tuned to the performance evaluation of the tracking system, using metrics such as tracking accuracy (Acc), ID switches (ID_sw), precision (Pr), and number of misses (Misses). The performance can be represented compactly with Eq. 5.

$$\text{Performance} = f(\text{Acc}, \text{ID\_sw}, \text{Pr}, \text{Misses}; \Theta) \tag{5}$$

where Θ is iteratively adjusted to enhance the model's tracking performance. Our integrated tracking model, combining self-supervised detector and OC-SORT tracking, optimized through iterative fine-tuning, is represented as:

$$T_{\text{final}} = \text{OC-SORT}_{\text{optimized}}(\text{Sef-supervised detector}(I_t; \phi, \theta); \Theta_{\text{optimized}}) \tag{6}$$

This equation of integration of detection outputs with the tracker represents a significant leap forward in achieving more accurate, reliable, and efficient results in multi-object tracking, particularly adapted to the dynamic and often challenging environment of animal farms. Our customized approach not only elevates tracking accuracy but also ensures the model's adeptness at meeting the specific challenges of object tracking.

4 Dataset and Evaluation Metrices

4.1 Custom Dataset

The dataset used was collected at the Norsvin SA's test station in Norway. Two different colored and sized pigs housed in groups were recorded in their

pens during the growth period from 30 to 120 kg. Videos were recorded 24/7 in top-down view by LOREX (4K Ultra HD IP NVR) and ELOTEC (4MP Bullet, IP67) cameras with a resolution of 1920 × 1080 under different lighting conditions. There were overall 8 videos with 600 frames each. The annotations were provided which contained bounding boxes (Darwin), class IDs, and other appropriate details of every object in every frame. The tracking ground truth is generated using the CVAT annotation tool.

4.2 Dancetrack

The DanceTrack [20] is a comprehensive benchmark dataset designed for tracking multiple objects in scenarios with uniform appearance and diverse motion. It contains a total of 100 videos, with 40 videos available for training (public annotations), 25 videos for validation (public annotations), and 35 videos for testing. Each video in the dataset features group dancing, emphasizing the importance of motion analysis in multi-object tracking rather than relying solely on appearance-based matching. The dataset provides bounding box annotations for both object positions and identities.

4.3 Evaluation Metrics

We assess the performance of our model through key metrics, including Identification Metrics (IDF1), ID switches (IDSW), fragmentation (Frag) and Multiple Object Tracking Accuracy (MOTA). The IDF1 score measures the ratio of correctly identified tracking over the average number of ground-truth and computed detections. It combines aspects of precision and recall, but unlike other metrics, it focuses on the consistency of identity assignments as given by (Eq. 7).

$$\text{IDF1} = \frac{2\text{IDTP}}{2\text{IDTP} + \text{IDFP} + \text{IDFN}} \tag{7}$$

MOTA is an aggregated metric that combines false negatives (FN), false positives (FP), and identity switches (IDSW). Matching occurs at a detection level, comparing Ground Truth Detections (gtDets) with Prediction Detections. This process involves associating each prediction detection with its corresponding ground truth detection to evaluate the accuracy of the tracking system (Eq. 8). It provides an overall sense of the tracker's performance, taking into account how often it misses objects, falsely detects non-existent objects, and makes mistakes in identity assignments.

$$\text{MOTA} = 1 - \frac{|FN| + |FP| + |IDSW|}{|gtDets|} \tag{8}$$

5 Results

5.1 Implementation Details

The implementation and experiments were carried out using the PyTorch open-source library [21] and OpenCV [22]. The computational setup involved a machine equipped with a 32GB NVIDIA GeForce RTX 2080 SUPER GPU. As part of the pre-processing stage, frames were extracted from videos and resized to dimensions of 512×512.

Figure 3 illustrates the cycle of the entire tracking pipeline. Initially, frames were extracted from the videos (as depicted in 3a). Subsequently, each frame undergoes processing through the self-supervised detection algorithm, as shown in 3b, to detect pigs, create bounding boxes around the identified animals, and record the animal spatial information in a separate file. Later, the spatial information of two consecutive frames are use to track the animals (as depicted in 3c).

(a) Actual frame (b) Frame Detection (c) Frame Tracking

Fig. 3. Tracking pipeline: frames extracted from videos (3a), processed by self-supervised detection (3b) to identify pigs and record spatial information. The spatial data of consecutive frames is then utilized for animal tracking (3c).

In addition to providing spatial information in the form of bounding boxes, the detector gives confidence scores and frame numbers. When dealing with our custom dataset where the number of animals is predefined, and they remain within the enclosure throughout tracking. However, in unconstrained surveillance scenarios, the number of objects may vary. Tracking involves associating the detected animals in the current frame with existing trajectories or identifying them as new entities. Our tracker efficiently accomplishes this by associating detections with existing trajectories and initiating new tracks when necessary. Once an object is linked to a track, the Kalman filter [23] is used to model the motion of the animals. The Kalman filter predicts the animal's position in the next frame and updating the track with the new detection. For new objects, a fresh Kalman filter is initialized. In instances where an object is occluded or missed by the detector in a frame, the Kalman filter prediction, contingent on the max_age parameter, offers an estimated position, ensuring tracking continuity. Over time, tracks that consistently lack updates, possibly due to an animal leaving the field of view or sustained occlusion, are discarded based on

the predefined max_age. The final outputs include updated tracks after each frame, delivering a continuous estimation of each detected animal's trajectory throughout the scene (3c). These processes are iteratively executed for each new frame, facilitating real-time animal tracking in the video. The self-supervised object detector demonstrates high accuracy, with a precision rate of 86.34%, a recall rate of 92.86%, and an F1 score of 91.59%, affirming its effectiveness in identifying and tracking animals reliably.

Table 2 presents the tracking performance metrics in which the IDF1 score at 85.97% indicates the tracker's effectiveness in maintaining object identities across frames. MOTA at 87.48% reflects the overall tracking accuracy, taking into account all object mismatches, missed detections, and false positives.

Table 2. Quantitative results of the proposed method against reference models: The quantitative results emphasize our tracker's effectiveness with an IDF1 score of 85.97% for consistent object identity maintenance. MOTA, at 87.48%, represents overall tracking accuracy, accounting for mismatches, missed detections, and false positives.

Models	Dancetrack dataset			Custom Pig dataset			
	DetA (%)	MOTA (%)	IDF1	MOTA (%)	IDF1	IDSW	Frag
SORT [8]	72	91.8	50.8	75.7	53.4	11684	216
FairMOT [10]	66.7	82.2	40.8	-	-	-	-
DeepSORT [9]	71	87.8	47.9	84.17	74.04	491	431
ByteTrack [11]	71.6	89.5	52.5	36.89	54.41	71	270
OC-SORT [7]	80.4	89.6	54.6	-	-	-	-
Proposed	**82.01**	**89.9**	**64.5**	**87.48**	**85.97**	**41**	**62**

6 Conclusion

We proposed an effective online multi-object tracking algorithm for dynamic and complex environments. Utilizing a tracking-by-detection approach, the algorithm incorporates self-supervised object detection and a bag of Bayesian trackers. To validate and refine the proposed algorithm, we employed two datasets. The first is the publicly available Dancetrack dataset, while the second is a custom dataset collected and annotated from an animal farm. The performance of the detection and tracking algorithms was assessed through comprehensive comparisons with reference methods. The results showcased significant improvements across standard tracking metrics, including IDF1 and MOTA. The synergy achieved between the self-supervised object detector and the proposed tracker demonstrated robust performance. This was evident in consistently preserving object identities and minimizing identification errors throughout sequences.

References

1. Höne, U., Krause, E.T., Bussemas, R., Traulsen, I., Schrader, L.: Usage of outdoor runs and defaecation behaviour of fattening pigs. Appl. Anim. Behav. Sci. **258**, 105821 (2023)
2. Kresovic, M., Nguyen, T., Ullah, M., Afridi, H., Cheikh, F.A.: Pigpose: a realtime framework for farm animal pose estimation and tracking. In: IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer, pp. 204–215 (2022) https://doi.org/10.1007/978-3-031-08333-4_17
3. Hostiou, N., et al.: Impact of precision livestock farming on work and human-animal interactions on dairy farms. a review. Biosci. Biotechnol. Biochem. **21**, 1–8 (2017)
4. Tøn, A., Imran, A.S., Ullah, M.: Wild animal species classification from camera traps using metadata analysis. In: 11th European Workshop on Visual Information Processing (EUVIP), pp. 1–6. IEEE (2023)
5. Afridi, H., et al.: Analyzing data modalities for cattle weight estimation using deep learning models. J. Imaging **10**(3), 72 (2024)
6. Tan, M., Pang, R., Le, Q.V.: Efficientdet: scalable and efficient object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10781–10790 (2020)
7. Cao, J., Pang, J., Weng, X., Khirodkar, R., Kitani, K.: Observation-centric sort: rethinking sort for robust multi-object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9686–9696 (2023)
8. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: IEEE International Conference on Image Processing (ICIP), pp. 3464–3468. IEEE (2016)
9. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: IEEE International Conference on Image Processing (ICIP), pp. 3645–3649. IEEE (2017)
10. Zhang, Y., Wang, C., Wang, X., Zeng, W., Liu, W.: FairMOT: on the fairness of detection and re-identification in multiple object tracking. Int. J. Comput. Vis. **129**(11), 3069–3087 (2021). https://doi.org/10.1007/s11263-021-01513-4
11. Zhang, Y., et al.: Bytetrack: multi-object tracking by associating every detection box. In: European Conference on Computer Vision. Springer, pp. 1–21 (2022) https://doi.org/10.1007/978-3-031-20047-2_1
12. Zhou, X., Wang, D., Krähenbühl, p.: Objects as points. arXiv:1904.07850 (2019)
13. Ullah, M., Alaya Cheikh, F.: A directed sparse graphical model for multi-target tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 1816–1823 (2018)
14. Bouttefroy, P., Bouzerdoum, A., Phung, S., Beghdadi, A.: Abnormal behavior detection using a multi-modal stochastic learning approach. In: 2008 International Conference on Intelligent Sensors, Sensor Networks and Information Processing, pp. 121–126 IEEE (2008)
15. Bouttefroy, P.L.M., Bouzerdoum, A., Phung, S.L., Beghdadi, A.: Vehicle tracking using projective particle filter. In: 2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, pp. 7–12 IEEE (2009)
16. Beghdadi, A., Mallem, M., Beji, L.: Benchmarking performance of object detection under image distortions in an uncontrolled environment. In: 2022 IEEE International Conference on Image Processing (ICIP), pp. 2071–2075 IEEE (2022)

17. Tan M., Le, Q.: Efficientnet: rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, pp. 6105–6114 PMLR (2019)
18. Wojke, N., Bewley, A.: Deep cosine metric learning for person re-identification. In: IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 748–756 IEEE (2018)
19. Dev Narayan, C.B., et al.: Tracking-by-self detection: a self-supervised framework for multiple animal tracking. In: IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer, pp. 561–572 (2023). https://doi.org/10.1007/978-3-031-34111-3_47
20. Sun, P., et al.: Dancetrack: multi-object tracking in uniform appearance and diverse motion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 20993–21002 (2022)
21. Imambi, S., Prakash, K.B., Kanagachidambaresan, G.: PyTorch, Programming with TensorFlow: Solution for Edge Computing Applications, pp. 87–104 (2021)
22. Bradski, G.: The opencv library. Dr. Dobb's J. Software Tools Prof. Programmer **25**(11), 120–123 (2000)
23. Patel, H.A., Thakore, D.G.: Moving object tracking using Kalman filter. Int. J. Comput. Sci. Mob. Comput. **2**(4), 326–332 (2013)

The Impact of Augmentation Techniques on Icon Detection Using Machine Learning Techniques

Mădălina Dicu(✉) and Camelia Chira

Computer Science Department, Babeş-Bolyai University, Cluj-Napoca, Romania
{madalina.dicu,camelia.chira}@ubbcluj.ro

Abstract. This article examines the use of image augmentation techniques to improve icon detection in mobile interfaces, a critical task due to the small size of graphical user interface (GUI) elements and the insufficiency of comprehensive datasets. It evaluates whether diversifying the dataset or using specific augmentation methods alone can enhance detection performance. The study specifically compares the performance of two models, Faster R-CNN and YOLOv8, in detecting these elements, highlighting the challenges and potential solutions in automating complex process interactions through improved icon recognition. The findings from our computational experiments reveal that the application of classical image augmentation methods to enhance dataset diversity significantly improves the performance of these models. Remarkably, such augmentation techniques are capable of yielding results that are comparable to, or even exceed, those obtained from training the models on considerably larger datasets. It is particularly noteworthy that models demonstrate superior performance when they are initially provided with a substantial volume of annotations, surpassing the outcomes associated with models trained on extensive data collections. Among the various augmentation techniques evaluated, image rotation emerged as the most effective in enhancing the performance of both models. Nonetheless, it was observed that the Faster R-CNN model consistently outperformed the YOLOv8 model across all experiments.

Keywords: Icon Detection · Faster R-CNN · YOLOv8

1 Introduction

Icons play a crucial role in the design of user interfaces (UI), offering significant benefits in navigation and interaction within applications due to their quick recognition by users [11]. Often, they are the first elements identified by newcomers to an application and are frequently used. To ensure consistency in usage and appearance, icons are usually standardized. However, computers face challenges in identifying and interacting with icons, as they can be mistaken for other UI elements such as images.

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 318–331, 2024.
https://doi.org/10.1007/978-3-031-63215-0_24

Identifying user interface (UI) elements is the first step in automating any process [16]. The accurate recognition of icons is vital, especially for automated processes that rely on these elements to perform various tasks. Icon detection in images is particularly important as it aids in understanding the purpose and functionality of UI elements, thereby improving accessibility and the user experience. This is essential in industrial applications where efficiency and speed are critical for productivity and operational safety. An example of this is the automated testing of graphical interfaces, where accurate icon identification can significantly simplify the validation of an application's functionalities [4].

However, intuitive approaches, such as directly supplying an application's icons for template matching or other direct techniques, face significant limitations [8]. These methods do not account for variations in the presentation of icons across different contexts, limiting the system's ability to generalize and its effectiveness in real-world scenarios. Therefore, detecting icons is imperative to ensure flexibility and adaptability in UI element recognition.

This work aims to address the challenge of accurately identifying icons without requiring an extensive dataset, as many modern applications seeking to automate their processes and recognize interface elements have limited screens or structures available for training a model to meet their specific requirements. To this end, we have selected two renowned models, Faster R-CNN [13] and YOLOv8 [9], for their ability to recognize small elements and their top performance in object detection.

The main contribution of this research is the investigation of the impact of three established methods of image augmentation on icon detection. We have chosen the most established methods of augmentation-rotation, horizontal flipping, and changing color features-because we want to see if these are sufficient to significantly improve the performance of the proposed models, and whether we can rely solely on these data augmentation methods when faced with a limited dataset for icon identification. A comparative analysis of these augmentation methods and the evaluation of the performance of the proposed models on a public dataset of mobile images are presented, with the experiments indicating that using augmentation methods to expand small datasets for icon detection shows promising results. These augmentation techniques significantly enhance the performance of the proposed models, achieving results comparable to those obtained by training the models on a large and diverse dataset.

The organization of the paper is as follows: Sect. 2 discusses related work on augmentation methods for UI element detection, with a focus on icon detection. Section 3 describes the chosen models, proposed augmentation techniques, and evaluation metrics. Section 4 details the dataset used and the results obtained. Conclusions are presented in Sect. 5, concluding the discussion.

2 Related Work

Numerous research studies have investigated the user interface (UI) components present in images of Graphical User Interfaces (GUIs). One study [19] has introduced a dataset for detecting elevator buttons, comprising 2005 board images

and 21767 button labels across 365 classes with varying forms and characteristics. During the training phase, the authors employed data augmentation methods such as random rotation, shift, zoom, brightness, and shear to enhance the model's training.

In a different approach, Zang et al. [17] created a UI dataset, containing annotations for the most commonly used icons across 29 classes. This dataset was derived from the Rico dataset. The authors proposed an object detection method that leveraged information from the view hierarchy, enhancing the accuracy of icon detection by associating relevant information from the view hierarchy leaf nodes with corresponding image regions.

Zhang et al. [18] attempted the challenge of detecting small icons in mobile applications, by utilizing the IconYOLO model. This model was trained on the MobileIcon dataset, which consisted of 32670 images of 20 common mobile phone icons. The authors improved small-scale icon detection by using a changed passthrough operation that combined deep semantic information with trivial detail features. Additionally, the dataset was augmented with horizontal flips and random rotations to improve the model's performance. The IconYOLO model outperformed YOLOv3 by 9.63% on the MobileIcon dataset and demonstrated generalization capabilities on the traditional Pascal VOC dataset.

The discussed research studies have shown that augmentation techniques are widely used for UI element detection models. Methods such as random rotation, shift, zoom, shear, and brightness adjustments have been used to enhance the performance of detecting GUI elements.

3 Methodology

To ascertain whether a large dataset is necessary for successful icon identification in images or if we can utilize a restricted dataset and apply augmentation techniques, our research endeavors to examine the influence of augmentation approaches on icon detection. As a result, we concentrate on studying two object identification models and assessing three well-known image augmentation techniques as part of our investigation. This part focuses on describing the three augmentation strategies, the models being examined, and the methods of evaluation in use. We also provide a comprehensive overview of the research approach and our proposed methodology.

3.1 Faster Region-Based Convolutional Neural Network (Faster R-CNN)

Faster R-CNN [13] is an advanced object recognition algorithm that employs Convolutional Neural Networks to identify and locate objects within images. Serving as an extension of the widely recognized R-CNN [7] and Fast R-CNN [6] models, it introduces the Region Proposal Network (RPN) as a key innovation. The RPN operates as a fully convolutional network, facilitating end-to-end training and efficient computation. By sliding a small network across the image's

convolutional feature map, the RPN generates object proposals using predefined anchor boxes with varying scales and aspect ratios.

After the RPN generates proposals, they are sent to a detection network that typically employs the Fast R-CNN model for bounding box refinement and classification. The convolutional feature map of the entire image is used by the detection network to extract feature vectors of a specified length for each proposal, which are then processed through a fully connected layer. Ultimately, Faster R-CNN generates a set of bounding boxes and associated class probabilities for each category of observed objects [1].

3.2 You only Look once (YOLO)

The YOLO [12] algorithm has revolutionized object detection in computer vision with its real-time, end-to-end approach. It creates a grid of cells from the input image and forecasts the bounding boxes and class probabilities for each cell's content. This method provides efficient and accurate detection by encoding spatial information and category possibilities.

Based on the successful YOLOv5 architecture, YOLOv8 [9] improves object detection performance by introducing the C2f module. This module combines high-level features and contextual information to enhance accuracy. YOLOv8 uses an anchor-free model with a decoupled head, allowing each branch to focus on its specific task of objectness, classification, and regression. The output layer uses sigmoid for the objectness score and softmax for class probabilities. Furthermore, YOLOv8 utilizes advanced loss functions for bounding box loss and binary cross-entropy for classification loss, which has proved to be particularly effective for smaller objects [15].

3.3 Data Augmentation

Data augmentation is a methodology employed to enhance the volume of a dataset by generating supplementary data. This process not only aims to expand the dataset but also to improve the model's generalization and robustness. By introducing slight alterations to the existing data or employing machine learning models to produce fresh data, we can simulate a wider variety of real-world scenarios that the model might encounter [14]. This diversity in training data helps the model learn to recognize patterns and features across a broader spectrum of conditions, thereby enhancing its ability to accurately identify icons in images under different circumstances.

The core objective of our research is to explore whether successful icon identification in images necessitates an extensive dataset with a wide variety of examples or if comparable results can be attained using a smaller dataset, provided that data augmentation techniques are applied. The reason for employing data augmentation is to make the model more versatile and improve its performance on data it has not encountered before by training it with a dataset that simulates the variability found in real-world situations.

Our study focuses on three specific augmentations: rotating images, modifying color features, and horizontal flipping, which are chosen for their potential to mimic the variations in how icons might appear in different environments.

To rotate images, we used the Python PILLOW library and applied affine transformations to update the positions of elements within the images, effectively simulating the orientation changes that icons might undergo in real-world scenarios [20].

Adjusting color features involves changing saturation, exposure, and hue (SEH), aiming to enhance the model's ability to identify icons accurately under various lighting and color conditions. We adjusted these features within a range between 1/2 and 2, using the "imgaug" Python library. This ensures that the model can accurately identify icons regardless of color variations, enhancing its applicability across different devices and settings. Annotations of icons within the image did not require adjustments.

Horizontal flipping mirrors an image along a vertical axis, which helps the model learn to recognize icons irrespective of their orientation, further contributing to the model's adaptability and efficacy. We implemented this technique in the same manner as the color feature modification, generating new images without recalculating the bounding boxes.

3.4 Evaluation Metrics

When evaluating a model's performance, we consider Intersection over Union (IoU), Average Precision (AP), and Confidence Threshold. The IoU method evaluates the degree of overlap between the actual bounding boxes and the predicted ones by measuring the ratio of the intersection area to the union area.

Precision is calculated by dividing the number of correct positive predictions by the total number of positive predictions, while recall is determined by dividing the number of correct positive predictions by the total number of ground truths in the dataset. The model's ability to recognize objects of interest is represented by Average Precision, which is calculated by determining the area under the precision-recall curve [10].

Confidence thresholds filter model predictions based on their confidence scores, allowing control over the inclusion of predicted objects and influencing precision and recall metrics.

3.5 Experiment Setup

For our experiments, we utilized established structures for both models. The Faster R-CNN model employed a pre-trained Resnet50 architecture as its backbone. Training was carried out over 100 epochs with a batch size of 2. We used stochastic gradient descent as the optimizer, setting the learning rate to 0.005.

In the case of the YOLOv8 model, we adopted the model proposed by Ultralytics [9]. Similarly, we employed the YOLOv8m model within the YOLO framework. Training was carried out over 100 epochs with a batch size of 2. Stochastic gradient descent was used as the optimizer, and the learning rate was set to 0.01.

The selection of the learning rate, optimizer, and number of epochs was based on common defaults for these models, providing a reliable starting point. Additionally, we chose a small batch size due to the limited number of images available in our dataset.

It is worth mentioning that although a large number of epochs were initially set, both models were configured to save the best-performing model during training to prevent overfitting. All experiments were conducted using the Google Colaboratory [2] platform.

3.6 Proposed Approach

The primary goal of this study is to assess the effectiveness of various image augmentation techniques in generating additional images for a dataset that initially contains a limited number of images. This process aims to determine whether these methods are adequate for significantly enhancing the algorithm's performance and to identify which of these proposed techniques are most suitable for addressing the problem of icon identification.

Evaluation methods were developed by establishing three confidence threshold levels for predictions across the two models, specifically 0.75, 0.50, and 0.25. We opted to include lower thresholds because we observed that the YOLOv8 model efficiently detects elements at lower confidence threshold values. This approach allowed us to evaluate the effects of the augmentation methods and the models' performance without having to eliminate a significant amount of detections, a constraint encountered with the use of a 0.75 threshold.

To ensure uniformity and consistency in the evaluation process, an IoU threshold of 0.50 was selected. Additionally, to calculate average precision, each model employs its approach. However, to maintain coherence and comparability in the results, we adopted the methodology proposed by Padilla et al. [10].

Regarding the augmentation methods investigated, we focused on three well-described techniques in Sect. 3.3. It is important to note that these augmentation methods will be applied exclusively to the training dataset, and not to the validation or testing datasets. Consequently, for the types of images generated, we implemented the following procedures: for each original image, we applied rotation to create three additional images with angles of 90, 180, and 270°. We also made modifications to the color characteristics (saturation, exposure, and hue) of the original images, resulting in a single modified image. Furthermore, we horizontally flipped each original image, generating an additional image. As a result, a total of five augmented images were generated for each original image, effectively expanding the dataset by a factor of six.

To discern the crucial techniques for enhancing the performance of the proposed model, we combined all the proposed augmentation methods, leading to seven distinct experiments, each integrating one or more of the proposed augmentation techniques.

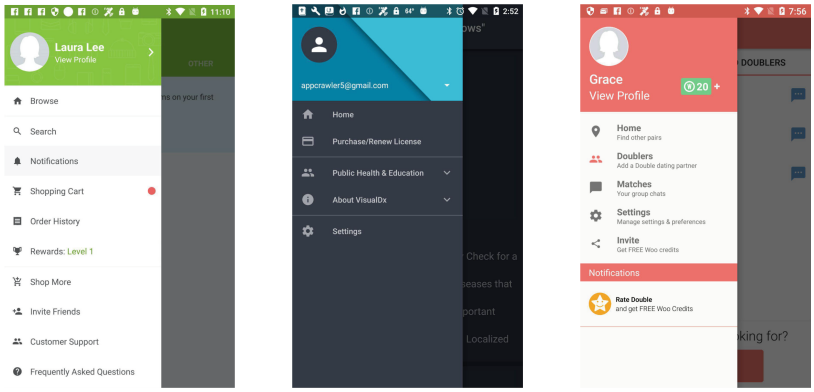
4 Experiments and Results

In this section, our focus is on presenting the dataset utilized in the experiments and the outcomes derived from conducting the experiments. Finally, we delve into detailing the observations drawn from the obtained results.

4.1 Dataset Description

In our experiments, we utilized the VINS dataset [3], which comprises 4543 images of mobile phone screens, categorized into four folders as follows: 2000 images derived from the Rico dataset [5], 1200 images of iPhone screens, 740 images of Android screens, and 603 UI design images sourced from design sharing websites. Specifically, for our investigation, we focused on the folder containing the 2000 Android screen images obtained from the Rico dataset [5].

These images are stored in JPG format and are accompanied by XML files that contain detailed annotations of the UI elements present in each image. Examples of VINS images can be seen in Fig. 1.



(a) Image 4141 (b) Image 14470 (c) Image 24659

Fig. 1. Images from VINS dataset [3]

Upon analyzing these 2000 images, we discovered that not all of them include icons in their content. Among these, there are images without even a single annotation for the icon class. Therefore, out of the 2000 images, only 1072 images include annotated icons. Furthermore, it is worth noting that, although the images feature annotated icons, there were instances where some annotations were missing. Consequently, our training and evaluation procedure focuses exclusively on using the existing annotations in the dataset. In total, the dataset we are analyzing contains 1072 images and 3486 icon class annotations.

To determine whether augmentation methods indeed offer more benefits in enhancing model performance than a dataset with a significantly larger number

of different images, we started our experiments with a dataset of 120 images. We also noted significant differences in the number of annotations within the images. Thus, there are images with a very high number of annotations in a single image, and others with a very few (some images contain only one icon). To analyze our approach more efficiently, we devised two scenarios:

- Scenario 1: includes the first 120 images that have the highest number of icon annotations. For this process, we sorted the 1072 XML files in descending order based on the number of icon elements they contain and extracted the top 120.
- Scenario 2: disregarded the number of annotations in each image and randomly selected 120 images from the dataset.

To thoroughly inspect the impact of augmentation methods on these types of images and to see if the division of data within each scenario plays a significant role in the analysis, we decided to randomly split the dataset five times for each scenario, applying augmentations each time. The division ratio is 8:1:1, meaning a training set, a validation set, and a testing set. Thus, for both scenarios, we have the same images in the dataset, but they are distributed differently across the three sets for each division variant. Detailed information regarding the images and their corresponding annotations used in the experiments is provided in Table 1.

Table 1. The number of images and annotations for each dataset and its variants. In Scenario 1, we include images with the highest number of annotations, while in Scenario 2, images are selected randomly. The Entire Dataset (last row) refers to the full dataset from which images for Scenarios 1 and 2 were selected.

Datasets		Number of Images				Number of Annotations			
		Total	Train	Validation	Test	Total	Train	Validation	Test
Scenario 1	V1	120	96	12	12	1116	897	114	105
	V2						881	118	117
	V3						889	119	108
	V4						898	114	104
	V5						882	119	115
Scenario 2	V1	120	96	12	12	571	466	57	48
	V2						475	48	48
	V3						458	64	49
	V4						454	68	49
	V5						475	49	47
Entire Dataset		1072	858	107	107	3486	2787	345	354

4.2 Results

We begin our approach by training the models associated with Scenario 1, as it contains the 120 images with the most annotations, significantly reducing the likelihood of encountering images without annotations. We train the two models, Faster R-CNN and YOLOv8, for the five existing variations. Subsequently, we continue by training the same models on datasets upon which we apply combinations of augmentations to increase the volume of data. To compare the efficiency of the models with the original dataset, we train the two models using the entire dataset, as specified in the last row of Table 1. The results obtained for Scenario 1 are in Table 2. Given the practicality of having five trained models per experimental setup, we derived average metrics across each confidence threshold for analysis. As the most relevant results for the YOLOv8 model are for a confidence threshold of 0.25, we calculate only for this threshold the standard deviation, to observe how much the results vary.

Table 2. Mean Average Precision results for Scenario 1 for the two evaluated models. Three different confidence thresholds (CT) of 0.75, 0.5, and 0.25 were applied, all using an IoU threshold of 0.5. The table results represent the average obtained for that particular combination of mean Average Precision over the five variants. The last column of each model represents the standard deviation (σ) for a confidence threshold of 0.25. The best results obtained are highlighted in bold.

Experiment	Faster R-CNN				YOLOv8			
	75	50	25	σ	75	50	25	σ
(1) Original	85.70	86.90	87.74	2.31	15.45	51.74	66.66	2.69
(2) Original + Rotation	88.95	90.62	91.39	2.29	36.66	71.10	79.76	2.87
(3) Original + Flip	89.41	89.97	90.60	1.96	36.32	65.04	72.77	2.82
(4) Original + SEH	88.38	89.65	90.52	1.88	44.94	70.80	78.13	3.51
(5) Original + Rotation + Flip	89.55	90.83	92.04	2.38	61.25	76.35	82.86	2.78
(6) Original + Rotation + SEH	90.74	92.30	92.68	1.70	54.83	77.09	82.78	2.22
(7) Original + Flip + SEH	89.98	90.81	91.46	2.06	55.46	74.72	80.47	3.09
(8) Original + Rotation + Flip + SEH	91.34	92.60	**93.27**	1.57	54.16	79.31	**84.79**	2.83
Entire Dataset	82.57	84.02	85.78		49.20	68.10	76.97	

From the obtained results, we observe that using multiple augmentation methods increases the performance of both models. Both the Faster R-CNN and YOLOv8 models achieve the best results for the situations where we use multiple augmentation methods, thus, for both models, the best result is obtained for the experiment where we add all three augmentation methods at all confidence thresholds, while the experiments where we use just one augmentation method achieve the lowest results in terms of performance. Moreover, the combination in which we have flipped images and images with changed color features achieves better results than the situation where we have only rotated images, even though the number of images in the second combination is larger. Thus, the diversity

of training data is crucial in addressing the problem of icon detection. Therefore, the combination of the three augmentation methods improves the results of the models trained only on the 120 images by 5.53% for the Faster R-CNN model and by over 18% for the YOLOv8 model (for a confidence threshold of 0.25). However, it is worth mentioning that the rotation augmentation method has the most significant impact, as all combinations that include it, obtain the best results in terms of the performance of both models. Combinations, where rotations and just one of the other two augmentation methods are used, have performances comparable with the optimal combination where we introduce all three methods. It is also worth noting that for the Faster R-CNN model, besides the rotation method, the next augmentation method that impacts the models' results is the change in color features method, compared to the horizontal flip method in the case of the YOLOv8 model. Thus, these singular methods do not provide the best results for the models in question, but in combination with each other or with rotations, they achieve results comparable with the method where we apply the most augmentation methods. Therefore, we can obtain a comparable result also using fewer data augmentation methods, but these must be specific for each model.

Regarding the comparison of the results obtained in the experiments with the results obtained in training the model on the entire dataset (the last row from Table 2), we observe that the methods in which we use at least two types of augmentations achieve much better results than those obtained by training the model with a significant number of images and annotations. However, in the case of the Faster R-CNN model, the model trained on the 120 images still manages at this stage to achieve much better results than the one trained on 1072 images. We consider this is also caused by the images in which the annotations of the icons are missing, the reason for which the model has lower performance.

Comparing the two models with each other, it is visible that the Faster R-CNN model achieves much better results compared to the YOLOv8 model, being much more suitable in the detection of smaller elements, but with a significantly longer inference time.

To validate the results obtained for Scenario 1, we continue with the experiments on the datasets corresponding to Scenario 2. Thus, the datasets contain the same number of images, but the number of annotations is much lower, having many images with few annotations of the icon class. We carry out the same procedure for Scenario 2 as described for the first, thus, the obtained results can be examined in Table 3.

With such a small number of annotations, we observe from the start a drastic decrease in the performance of the obtained models. All results are, in terms of performance, weaker than the model trained on the entire dataset (the last row from Table 2). However, we note for the Faster R-CNN model a consistency in the results, namely, the best result of the augmentation combinations is achieved by combining all three augmentation methods, followed by the combinations including the rotation method. From this, we can conclude that indeed the diversity of data plays an important role in increasing the model's performance. If we do

Table 3. Mean Average Precision results for Scenario 2 for the two evaluated models. Three different confidence thresholds (CT) of 0.75, 0.5, and 0.25 were applied, all using an IoU threshold of 0.5. The results in this table show the average Mean Average Precision for the specified combination across five variations. The last column for each model denotes the standard deviation (σ) for a confidence threshold of 0.25. The highest-performing results are highlighted in bold.

Experiment	Faster R-CNN				YOLOv8			
	75	50	25	σ	75	50	25	σ
(1) Original	75.31	77.55	78.72	2.25	3.42	16.34	40.85	2.91
(2) Original + Rotation	77.04	80.14	81.07	2.32	18.25	40.58	50.75	1.89
(3) Original + Flip	75.35	78.38	80.07	1.83	25.15	51.44	**63.20**	1.52
(4) Original + SEH	78.97	80.35	81.37	2.74	6.14	34.27	53.71	2.54
(5) Original + Rotation + Flip	76.50	79.97	81.94	2.50	30.33	51.06	60.30	2.56
(6) Original + Rotation + SEH	78.75	82.12	82.67	1.47	20.82	43.37	54.58	1.92
(7) Original + Flip + SEH	77.73	80.30	81.46	3.08	25.37	53.78	66.26	2.43
(8) Original + Rotation + Flip + SEH	79.33	81.88	**83.14**	2.85	30.03	52.39	61.88	2.67

not have this possibility, using the rotation method in combination with another augmentation method can achieve comparable results.

On the other hand, for the YOLOv8 model, we see a discrepancy in the results, because the combination of several augmentation methods was outperformed by the use of the horizontal flip method alone. This augmentation method has the most significant impact in this context, and the combinations that included this method achieved the best results. This still validates the observations made in Scenario 1, namely, that besides the rotation method, the horizontal flip method has the greatest impact on the YOLOv8 model.

From the analysis, we conclude that the diversity of data greatly helps in increasing the performance of the icon detection models. However, it is not necessarily required to introduce countless augmentation methods, as comparable results can be obtained using augmentation methods that are suitable both for our data and for the chosen model. In terms of model selection, in the detection of icons from mobile interface images, the Faster R-CNN model distinctly achieves the best results, compared to the YOLOv8 model. Therefore, for the detection of small objects, the Faster R-CNN model is more appropriate, even if it is slower.

5 Conclusions and Future Work

In our research, we focused on augmenting the image dataset for icon detection using fundamental augmentation techniques, analyzing if comparable results in icon detection could be achieved with a limited image dataset through augmentation methods, versus training a detection model on a more extensive dataset. We employed object detection models Faster R-CNN and YOLOv8 in our study.

Faster R-CNN is adept at detecting small objects, while YOLOv8 represents a modern algorithm.

We explored augmentation methods such as rotation, horizontal flipping, and color characteristic adjustments, as primary means to increase image diversity in mobile interface contexts. Selecting a small dataset, we based our experiments on two scenarios: one with a maximum number of icon annotations and another with randomly selected images. We trained both models across eight experimental conditions, applying a combination of the three augmentation methods in each, and repeated each experiment five times to ensure data variability and consistency in our analysis.

Our findings suggest that comparable results can be obtained with a limited dataset through augmentation, as opposed to relying on a larger dataset. The best configuration was achieved by applying all three augmentation methods, though selecting only two augmentation methods can yield comparable results, depending on the detection model chosen. Rotation proved most effective for both models in all scenarios, but pairing it with color characteristic adjustments for Faster R-CNN and with horizontal flipping for YOLOv8 was more efficient.

Overall, the Faster R-CNN model consistently delivered superior performance, affirming its suitability for detecting small objects. Optimal augmentation, in scenarios with the maximum possible annotations, improved model performance by 5.53% for Faster R-CNN and nearly 18% for YOLOv8. In less favorable scenarios, classical augmentation methods similarly enhanced results.

There remains much to be accomplished to achieve optimal image understanding. Therefore, our future efforts will primarily focus on expanding the range of augmentation methods within the context of icon detection to specifically address the challenges posed by small-sized elements. This will include techniques such as zoom-in, resizing, cropping, and adding noise, tailored to enhance the specificity of the problem at hand. Additionally, we will broaden our scope to include the identification of other user interface (UI) elements within images. To accomplish this, we will explore and utilize specific augmentation techniques that are designed to improve the identification of UI components, ensuring a more comprehensive approach to image understanding.

Acknowledgements. The research presented in this paper received support from a project titled "Integrated System for Business Process Automation using Artificial Intelligence," funded by the European Regional Development Fund (ERDF) through the Operational Programme for Competitiveness 2014–2020. The project, with the identification number POC/163/1/3/121075, is a collaborative effort between ENDAVA and Babeş-Bolyai University Cluj-Napoca.

References

1. Abbas, S.M., Singh, S.N.: Region-based object detection and classification using faster R-CNN. In: 2018 4th International Conference on Computational Intelligence and Communication Technology (CICT), pp. 1–6. IEEE (2018)
2. Bisong, E., Bisong, E.: Google colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners, pp. 59–64 (2019)
3. Bunian, S., Li, K., Jemmali, C., Harteveld, C., Fu, Y., Seif El-Nasr, M.S.: Vins: visual search for mobile user interface design. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, pp. 1–14 (2021)
4. Cheng, J., Tan, D., Zhang, T., Wei, A., Chen, J., et al.: YOLOv5-MGC: GUI element identification for mobile applications based on improved YoLov5. Mob. Inf. Syst. **2022** (2022)
5. Deka, B., et al.: Rico: a mobile app dataset for building data-driven design applications. In: Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, pp. 845–854 (2017)
6. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)
7. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
8. Han, Y.: Reliable template matching for image detection in vision sensor systems. Sensors **21**(24) (2021). https://doi.org/10.3390/s21248176, https://www.mdpi.com/1424-8220/21/24/8176
9. Jocher, G., Chaurasia, A., Qiu, J.: YOLO by ultralytics (2023). https://github.com/ultralytics/ultralytics. Accessed 20 Feb 2024
10. Padilla, R., Netto, S.L., Da Silva, E.A.: A survey on performance metrics for object-detection algorithms. In: 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), pp. 237–242. IEEE (2020)
11. Passini, S., Strazzari, F., Borghi, A.: Icon-function relationship in toolbar icons. Displays **29**(5), 521–525 (2008)
12. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
13. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, vol. 28 (2015)
14. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. J. Big Data **6**(1), 1–48 (2019)
15. Terven, J., Cordova-Esparza, D.: A comprehensive review of YOLO: from YOLOv1 to YOLOv8 and beyond. arXiv preprint arXiv:2304.00501 (2023)
16. Xie, M., Feng, S., Xing, Z., Chen, J., Chen, C.: Uied: a hybrid tool for GUI element detection. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 1655–1659 (2020)
17. Zang, X., Xu, Y., Chen, J.: Multimodal icon annotation for mobile applications. In: Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction, pp. 1–11 (2021)

18. Zhang, Q., Pan, X., Liu, F., Lu, S.: A benchmark dataset for real-time detection of icons in mobile apps and a small-scale feature module. Pattern Recogn. Lett. **136**, 87–93 (2020)
19. Zhu, D., Fang, Y., Min, Z., Ho, D., Meng, M.Q.H.: OCR-RCNN: an accurate and efficient framework for elevator button recognition. IEEE Trans. Industr. Electron. **69**(1), 582–591 (2021)
20. Zoph, B., Cubuk, E.D., Ghiasi, G., Lin, T.-Y., Shlens, J., Le, Q.V.: Learning data augmentation strategies for object detection. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12372, pp. 566–583. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58583-9_34

Toward Unsupervised Energy Consumption Anomaly Detection

Hatem Haddad(✉), Feres Jerbi, and Issam Smaali

Wattnow, La Goulette, Tunisia
{hatem.haddad,feres.jerbi,issam}@wattnow.io
https://wattnow.io

Abstract. Existing high-performance Machine Learning models typically rely on large training datasets with high-quality manual annotations, which are difficult to obtain in the case of energy consumption anomaly detection. This knowledge gap is addressed in this study, where off-the-shelf unsupervised anomaly detection models are evaluated on real-world buildings energy datasets from different use cases varying in size and features. To this end, empirical evaluations and methodological analysis are conducted in order to evaluate how fully automated unsupervised labeling is in accordance with the domain experts manual labeling. We evaluate and discuss the impact of the anomalies contamination ratio, the missing values, the number of used features on fully automated unsupervised labeling and on avoiding false notifications by minimizing the False Positive and False Negative rates. In addition, the performances of combining models based on a voting process are evaluated. Results demonstrate the capabilities of Machine Learning anomaly detection models in distinguishing anomalous instances from normal instances. Additionally, achieved results highlights the similarity between Machine Learning models labeling and domain experts manual labeling. These findings support the development of anomaly detection in energy consumption scenarios, significantly reducing the time and cost of manual labeling.

Keywords: Energy consumption · Anomaly detection · Machine learning

1 Introduction

Recently, anomaly detection was applied on different IoT Environments such as intelligent inhabitant environments, intelligent transport systems, healthcare systems, and industrial systems [1]. In Machine Learning (ML), energy consumption anomaly detection is a technique to analyze, detect, and visualize abnormal energy consumption behavior. Nevertheless, annotated data is a basic requirement to apply ML models. Generally, manual annotation is time consuming, costly and not scalable [2]. Unsupervised data labeling is then required, ensuring a faster and more accurate annotation, so that ML models can go from

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 332–346, 2024.
https://doi.org/10.1007/978-3-031-63215-0_25

training to be production-ready more easily. For this aim, this paper proposes an evaluation of fully automated annotation techniques for energy consumption anomaly detection based on off-the-shelf ML models.

Anomaly detection has long been studied [3]. However, energy consumption anomaly detection was only studied in some of the papers as detailed in the Sect. 2. Based on this observation, a comprehensive investigation targeting the status of fully automatic unsupervised energy consumption anomaly labeling is indeed still needed. This drives us to find answers to the following research questions: (1) To what extent can energy consumption anomaly labeling be completed in a fully automatic and unsupervised way? (2) How the anomalies contamination ratio in a dataset impacts on the unsupervised models performances? (3) In real applications, missing values are usually caused by power outage or a hardware/software failure of the measurement device. Because missing data may result in biased or misinformed analysis and can reduce the accuracy of the ML models, how should the datasets preprocessing handle missing values? (4) How the number of used features impacts on the ML models performances? (5) Can we enhance anomaly detection performance by implementing an annotation voting process based on combining the annotations of the highest-performing annotation models? (6) To what extent can automatic energy consumption anomaly detection avoid false alarms as a common issue in industry, by minimizing the False Positive and False Negative rate and accordingly prune false notifications?.

2 Literature Review

In this section, we focus only on works related to the creation of ground truth labeled data leaving out the performances of the proposed approaches for detecting energy anomalies. Because of the lack of existing labeled ground truth datasets [4,5], early works on energy anomaly consumption have manually annotated datasets by domain experts or based on domain experts' knowledge [6]. Both approaches require a team of trained annotators and they are very time consuming and costly [7], especially for large datasets.

Authors in [8] limited the validation of their proposed unsupervised model results of abnormal energy consumption in buildings to manually inspect only the anomalies detected by their proposed model. If an anomaly exists in the dataset but not detected by their model, hence this anomaly is not considered by the used metrics. To facilitate the annotation process, authors in [9] used a visualization application[1] where plot power consumption for each day is visualized and analyzed manually in addition to weather variables such as temperature and humidity. When a deviation is observed in the plot and weather variables do not explain the deviation, the deviation is then labeled manually as an anomaly by a domain expert. Because manual annotation is a time consuming process, authors annotated only 4 datasets chunks (small part of the original datasets): Dataport[2], AMPds [10] (detailed in Sect. 3), ECO [11] (detailed in Sect. 3) and

[1] https://github.com/loneharoon/PowerViz.
[2] https://dataport.pecanstreet.org/.

REFIT [12]. Chunks include consecutive three months of energy data with minimal missing values. Authors publicly released the annotation of the four chunks[3]. In our study, we used AMPds and ECO datasets exclusively because the same annotators were involved in the labeling process for all four datasets [9], hence it was deemed redundant to apply ML models across every dataset .

The QUD dataset was introduced in [13] (detailed in Sect. 3). Labels were generated using a rule-based model with the aim of extracting micro-moment features over time (the occupancy profile, power consumption of each device in reference to its active consumption rate, maximum operation time, and standby consumption level). Nevertheless, the proposed rule-based model was neither compared to a domain expert labeling nor validated by a domain expert. Authors in [13] also used the PCSiD dataset synthetically built using data generation of hourly consumption profiles for a period of 2 years. However, achieved performances on PCSiD were inferior to when using real data from QUD dataset. For this reason, we did not use synthetic data in our experiments.

Authors in [14] employed a framework of tree-based models that includes several tasks of data preprocessing, feature engineering, data downsampling, modeling, and post-processing on the LEAD dataset [15]. Authors concluded that feature engineering, such as missing data imputation and feature normalization, are important to achieve the best performances.

Previous studies on labeling energy consumption anomaly detection focused on rule-based approaches. Rules flag out power instances that are remarkably higher or lower than usual consumption footprints without considering either more information sources (e.g. weather, novelty, occupancy patterns, and appliance operation data) nor comparing current consumption footprints with the past and ideal consumption cycles.

3 Datasets Description

In order to evaluate energy consumption anomaly detection models, we built the WATT 1.0 dataset, with its ground-truth labels based on an experimental campaign performed at a Bank headquarters building. In addition, four other existing datasets or datasets chunks are also considered in this study: QUD [13], AMPds [10], ECO [11] and LEAD [15]. Datasets characteristics are shown in Table 1 including the number of rows with missing values and the used features.

– WATT 1.0: Three-phase electric data is collected over a year by smart meters installed in a Bank headquarters building at one-minute intervals, including timestamp (t), active power (P), humidity (H) and temperature (T). In addition, occupancy (O) is added based on work time and holidays calendar. Domain experts manually annotated normal and anomalies instances in a 30-day data chunk (44640 instances).

[3] https://goo.gl/SjozdF.

- AMPds: The authors in [10] introduced the AMPds dataset encompassing measurements of electricity, water, and natural gas taken at one-minute intervals, resulting in a total of 1,051,200 readings per meter over a monitoring period of two years. Authors [9] manually annotated anomalies in a period of 60 days.
- LEAD: Authors in [15] introduced the LEAD dataset for energy consumption anomaly detection that spans for over a year data from 1,413 smart electricity meters of buildings across 16 diverse global locations. The dataset was manually annotated to identify daily anomalies based on hourly data of 12 building categories including Education, Office, Parking, Healthcare, Lodging/Residential, and more. We conducted extensive experiments on all building categories. Due to space constraints, we show in this paper ML models performances on 3 buildings categories: Office (referred to as LEAD-B147 from now on), Lodging/residential (referred to as LEAD-B936 from now on) and Parking (referred to as LEAD-B1141 from now on).
- ECO: The ECO dataset [11] constitutes a comprehensive resource for studies in non-intrusive load monitoring and occupancy detection. It was collected from six households over an eight-month period. Similar to the AMPds dataset, a portion of the data related to 60 days of readings from three distinct houses were manually labeled by the authors in [9]: ECO-H1, ECO-H4 and ECO-H6.
- QUD: The dataset collected from various appliances (e.g., light lamp, air conditioner, desktop, heating system) and includes contextual information (temperature, humidity, ambient light intensity, and room occupancy) with frequency ranging from 3 s to 3 h. The dataset is collected from an academic building (Qatar University) for a period of 1 year [13]. The dataset is annotated using five micro-moments classes. The first three represent normal consumption: good usage, turn on, and turn off. On the other hand, two classes describe anomalous consumption: excessive power consumption and

Table 1. Datasets characteristics and collected features: timestamp (t), active power (P), occupancy (O), humidity (H) and temperature (T).

Dataset Acronym	Primary Use	Period	Sampling Rate	Features	#Rows	#Anomaly	#Missing Rows
WATT 1.0	Office	30 days	1 min	t, P, O, H, T	44640	1156	9756
AMPds	House	60 days	10 min	t, P	8352	1043	0
QUD	Education	1 year	3 s-3 h	t, P, O	16057	3954	0
ECO-H1	House	60 days	10 min	t, P	8496	1529	0
ECO-H4	House	60 days	10 min	t, P	8496	865	0
ECO-H6	House	60 days	10 min	t, P	8496	1451	0
LEAD-B147	Office	1 year	1 h	t, P	8784	254	2654
LEAD-B936	Residential	1 year	1 h	t, P	8762	297	146
LEAD-B1141	Parking	1 year	1 h	t, P	8784	171	1

consumption when outside. In this study, we investigated the "Excessive power consumption" label as an indicator of energy consumption anomalies, alongside the "Good usage" label representing normal energy consumption.

4 Applied ML Models

Extensive experiments were conducted using 8 Off-the-shelf ML models associated to 8 paradigms [16] for detecting anomalies:

- Angle-Based Outlier Detection (ABOD): Utilizes angles between pairs of points in vector space to distinguish outliers from clustered points [17]. Suited for boundary detection, ABOD falls under the Angle based paradigm.
- Subspace Outlier Degree (SOD): Identifies anomalies in lower-dimensional subspaces of a high-dimensional space using distance-based detection [18]. SOD is categorized within the Density based paradigm.
- Cluster-Based Local Outlier Factor (CBLOF): Combines LOF model [19] metrics with cluster sizes to assess outliers, aligning with the Clustering based paradigm.
- Class Outlier Factors (COF): Extends LOF by evaluating local density with k-nearest neighbors to detect outliers in varying distributions [20]. COF is included in the Connectivity based paradigm.
- Histogram-Based Outlier Score (HBOS): Assesses anomalies based on feature space bin probabilities [21], representing the Histogram based paradigm.
- Isolation Forest (IF): Employs an ensemble of isolation trees to detect outliers in high-dimensional data [22]. IF is a part of the Distance Based paradigm.
- Minimum Covariance Determinant (MCD): Detects multivariate outliers by minimizing the determinant of the covariance matrix [23]. MCD belongs to the Dimensionality paradigm.
- One-Class Support Vector Machine (OCSVM): Distinguishes outliers by learning from a single-class data, relevant for anomaly detection [24]. OCSVM is classified under the One Class learning paradigm.

5 Evaluation Metrics

To demonstrate that fully automated unsupervised labeling is a viable alternative to manual labeling, we compare the labeling results of each ML model to those manually assigned by domain experts. Despite their limitations, we consider the experts manual annotations as the ground truth and baseline. For this reason, our evaluation is based on two annotation evaluation metrics [25]: Pairwise Relative Agreement Metric (referred to as PRAM from now on) and Cohen's kappa (referred to as K from now on):

- PRAM measures the agreement percentage between two annotators' labels on the same data, calculated by dividing the number of similar model labels and manual labels by the total rows. A PRAM near 100% signifies high label consistency with domain expert assignments.

Table 2. Unsupervised labeling performances on WATT 1.0, AMPds and QUD.

Model	WATT 1.0		AMPds		QUD	
	PRAM(%)	K	PRAM(%)	K	PRAM(%)	K
ABOD	95.24	0.05	88.84	0.48	75.38	0.00
SOD	95.28	0.06	85.09	0.05	91.01	0.73
CBLOF	**98.38**	**0.67**	88.27	0.46	76.17	0.35
COF	95.11	0.00	77.80	-0.01	67.15	0.11
HBOS	97.01	0.40	88.91	0.35	88.95	0.70
IF	97.78	0.56	91.20	0.59	85.38	0.60
MCD	97.28	0.46	**93.94**	**0.72**	**99.95**	**0.99**
OCSVM	98.00	0.60	92.05	0.63	77.33	0.38

– Cohen's Kappa, as the proportion of agreement corrected for chance [25], it assesses the chance's role in the annotation process, yielding scores from -1 (total disagreement) to 1 (total agreement), with 0 indicating random annotation. A value of $K > .80$ is considered Perfect [26].

In Sect. 6.5, a voting process is applied based on combining annotations from the highest performing annotation models on each dataset to assign a voting label. We measure inter-rater reliability using Krippendorff's Alpha (referred to as α from now on) [25] quantifying the model's agreement. This metric accounts for the overall annotation distribution generalized to more than two annotators and rated as Good ([0.8, 1]), Tentative ([0.67, 0.8]), or Discarded (<0.67).

6 Experiments and Results

Because features can have a broad range of values, experiments were conducted with and without applying Z-score [27] as a preprocessing technique. Notably,

Table 3. Unsupervised labeling performances on 3 buildings from LEAD dataset.

Model	LEAD-B147		LEAD-B936		LEAD-B1141	
	PRAM(%)	K	PRAM(%)	K	PRAM(%)	K
ABOD	97.11	0.00	96.61	0.00	98.05	0.00
SOD	94.38	0.00	93.65	0.02	96.31	0.02
CBLOF	99.11	0.84	**99.16**	**0.87**	**99.77**	**0.94**
COF	94.67	0.05	93.83	0.05	96.40	0.05
HBOS	97.05	0.38	95.40	0.26	96.60	0.04
IF	98.79	0.78	98.97	0.84	99.65	0.90
MCD	99.11	0.84	99.14	0.86	98.25	0.53
OCSVM	**99.13**	**0.84**	99.13	0.86	99.46	0.85

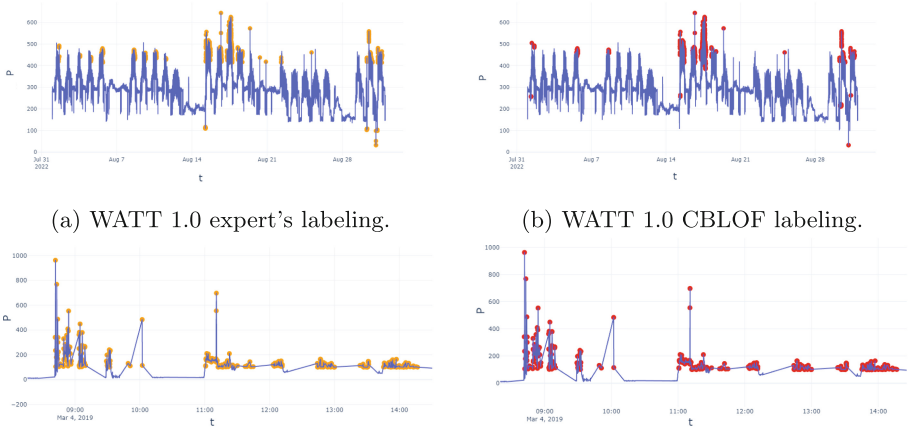
ABOD, CBLOF, IF, SOD, and OCSVM achieved higher performances when Z-score was applied, while COF, HBOS, and MCD performed better without applying Z-score.

6.1 Unsupervised Labeling Performances

Tables 2, 3 and 4 present models performances based on PRAM and K metrics where MCD model outperformed the other models on five datasets: AMPds (93.94%), QUD (99.95%), ECO-H1 (91.67%), ECO-H4 (89.48%) and ECO-H6 (95.36%). CBLOF demonstrated the highest performance on WATT 1.0 (98.38%), LEAD-B936 (99.16%) and LEAD-B1141 (99.77%).

Results show that MCD, OCSVM, and CBLOF consistently achieved the highest results among all the models indicating that the Dimensionality, One Class learning and Clustering based paradigms outperform Angle based, Connectivity based, Histogram based, Distance Based and Density based paradigms. This is also confirmed by their high K achieved performances.

Figure 1 highlights anomalies labels created by CBLOF and MCD on the WATT 1.0 and QUD datasets, respectively, and their similarity with the expert's manual labels.



(a) WATT 1.0 expert's labeling. (b) WATT 1.0 CBLOF labeling.

(c) QUD expert's labeling. (d) QUD MCD labeling.

Fig. 1. WATT 1.0 and QUD: Models vs. expert's labeling.

Table 4. Unsupervised labeling performances on 3 houses from ECO dataset.

Model	ECO-H1		ECO-H4		ECO-H6	
	PRAM(%)	K	PRAM(%)	K	PRAM(%)	K
ABOD	88.96	0.62	89.82	0.00	87.97	0.57
SOD	72.39	0.00	88.59	0.16	74.98	0.10
CBLOF	89.60	0.64	89.03	0.40	91.41	0.69
COF	68.64	−0.06	82.60	0.04	67.91	−0.13
HBOS	84.03	0.32	87.92	0.25	84.69	0.25
IF	90.03	0.66	87.90	0.33	92.75	0.74
MCD	**91.67**	**0.71**	**89.48**	**0.42**	**95.36**	**0.83**
OCSVM	86.82	0.55	88.32	0.36	89.01	0.61

6.2 Contamination Factor

In this section, we analyze the impact of anomalies contamination factor on the models performances by decreasing the ratio of anomalies for each dataset. The applied ratios are 10%, 8%, 6%, 4% and 2%.

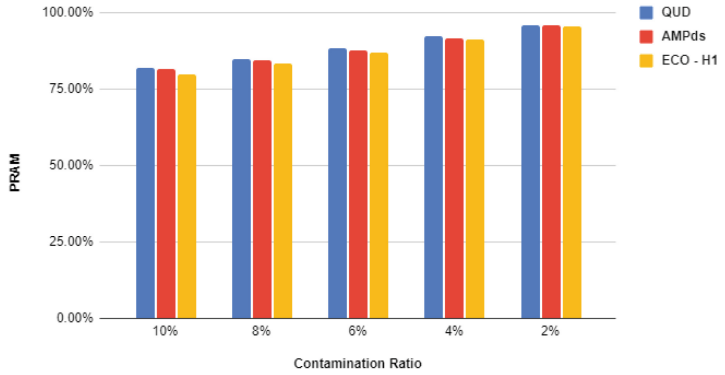


Fig. 2. COF's PRAM results based on different contamination ratios.

Figure 2 illustrates the impact of reducing the contamination ratio on PRAM performances across QUD, AMPds, and ECO-H1 datasets. Among the models, COF was actually the least performing. However, COF's performances improve when anomalies ratios decrease. As the ratio of anomalies decreases, there is a consistent improvement in PRAM performances. This continues to be the case for the other datasets. This indicates that ML models perform better when the percentage of normal instances are higher than anomalies. It is important to note that as the contamination ratio decreases, the ML models performances

increase at recognizing anomalies. However, if a high energy anomaly behavior is observed and the anomaly classes are no longer the minority classes, the task becomes more challenging. In such a situation, differentiating between normal and anomalous patterns poses a greater challenge for the ML models. Therefore, applying unsupervised anomaly detection models is less efficient.

6.3 Missing Values

Missing values can produce high probabilities for rising the False Positive notifications [28]. They are usually replaced by the values of previous year's data corresponding to the same timestamp or average of previous or current timestamp [29]. We applied two different strategies to handle missing data: linear interpolation on WATT 1.0 dataset and mean imputation on LEAD dataset.

In the case of the LEAD dataset, the missing values were initially labeled as "Normal". Table 6 illustrates a chunk of LEAD-B147, imputing it's missing values with the mean value (151.8) keeping their "Normal" label unchanged, reflecting the dominance of normal instances. However, using linear interpolation on WATT 1.0 showed mixed results: it worked well when missing values were between instances of the same label, but not when flanked by different labels as shown in the Table 5. These methods, though conventional, can bias the data, proving ineffective for handling missing values in this context.

Table 5. Chunk of WATT 1.0 before and after processing missing data.

WATT 1.0 Data				WATT 1.0 Processed Data				
Timestamp	P(A)	P(B)	P(C)	Timestamp	P(A)	P(B)	P(C)	Label
2022-08-05 14:05:00	347.2	347.7	348.1	2022-08-05 14:05:00	347.2	347.7	348.1	0
2022-08-05 14:06:00	**Null**	**Null**	**Null**	2022-08-05 14:06:00	**347.9**	**349.1**	**349.1**	**0**
2022-08-05 14:07:00	348.6	350.6	350.2	2022-08-05 14:07:00	348.6	350.6	350.2	0
...	...	...	...	...	...	...	...	...
2022-08-05 14:10:00	444.4	445.2	445.9	2022-08-05 14:10:00	444.4	445.2	445.9	1
2022-08-05 14:11:00	**Null**	**Null**	**Null**	2022-08-05 14:11:00	**448.7**	**449.3**	**450.2**	**1**
2022-08-05 14:12:00	453.1	453.5	454.6	2022-08-05 14:12:00	453.1	453.5	454.6	1
...	...	...	...	...	...	...	...	...
2022-08-05 14:15:00	461.6	461.4	462.5	2022-08-05 14:15:00	461.6	461.4	462.5	1
2022-08-05 14:16:00	**Null**	**Null**	**Null**	2022-08-05 14:16:00	**415.1**	**414.8**	**415.8**	**0**
2022-08-05 14:17:00	368.7	368.3	369.1	2022-08-05 14:17:00	368.7	368.3	369.1	0
...	...	...	...	...	...	...	...	...
2022-08-16 15:10:00	486.3	484.1	485.6	2022-08-16 15:10:00	486.3	484.1	485.6	1
2022-08-16 15:11:00	**Null**	**Null**	**Null**	2022-08-16 15:11:00	**435.0**	**432.7**	**434.0**	**1**
2022-08-16 15:12:00	383.7	381.4	382.4	2022-08-16 15:12:00	383.7	381.4	382.4	0

Table 6. Chunk of LEAD building 147 before and after processing missing data.

LEAD-B147 Data			LEAD-B147 Processed Data		
Timestamp	P	Label	Timestamp	P	Label
2016-01-05 23:00:00	129	0	2016-01-05 23:00:00	129	0
2016-01-06 0:00:00	152.821	1	2016-01-06 0:00:00	152.821	1
2016-01-06 1:00:00	**Null**	**0**	2016-01-06 1:00:00	**151.88**	**0**
2016-01-06 2:00:00	**Null**	**0**	2016-01-06 2:00:00	**151.88**	**0**

6.4 Number of Features

In this section, we study the ML models performances according to the number of applied features. Table 7 shows model performances using different numbers of features on the WATT 1.0 dataset based on 3 features combinations: (1) t, P, (2) t, P, T, and (3) t, P, O, T, H.

Table 7. WATT 1.0 performances using different features numbers.

Model	t, P, O, H, t		t, P, T		t, P	
	PRAM(%)	K	PRAM(%)	K	PRAM(%)	K
ABOD	95.24	0.05	95.17	0.04	95.29	0.06
SOD	95.28	0.06	95.22	0.05	95.28	0.06
CBLOF	**98.38**	**0.67**	97.61	0.52	97.45	0.49
COF	95.11	0.00	94.93	0.00	94.96	0.00
HBOS	97.01	0.40	96.94	0.39	97.31	0.44
IF	97.78	0.56	97.68	0.54	97.31	0.46
MCD	97.28	0.46	97.29	0.46	97.23	0.45
OCSVM	**98.00**	**0.60**	97.69	0.54	97.30	0.46

While COF, ABOD, SOD, HBOS, and MCD show marginal variations across feature combinations, CBLOF and OCSVM notably shift PRAM and K results. CBLOF exhibits improvements when increasing the number of features, achieving PRAM performance of 98.38% and K performance of 0.67 when using three features (3). The latter exceeded the 97.45% PRAM's and 0.49 K's performances using features combination (1), and the 97.61% PRAM's and 0.52 K's performances with features combination (2). Similarly, OCSVM shows enhanced performance when increasing the number of used features. With features combination (3), PRAM and K performances reached 98.00% and 0.60, respectively, outperforming the 97.69% and 0.54 with features combination (2), and the 97.30% and 0.46 with features combination (1).

6.5 Voting Labeling Performances

This section evaluates the effectiveness of model labeling combination through a voting process considering each model as an independent annotator. The study used the top 3 models for each dataset, assigning labels through unanimous or majority agreement. For evaluation, we compare the voting labels to the manually assigned labels by the domain experts based on PRAM and K metrics as shown in Table 8 (due to space constraint, ECO performances are not shown in this Table). Results indicated that the voting process lowered the performances of WATT 1.0, AMPds, and QUD datasets. This is due to the likelihood of majority-based incorrect label assignments, diminishing the reliability of voting labels. Furthermore, since single models already achieve high PRAM and K performance, combining models through voting did not enhance labeling performance, even with models from varied paradigms, indicating that the voting process may be ineffective.

Table 8. Voting process performances based on PRAM, K and α metrics.

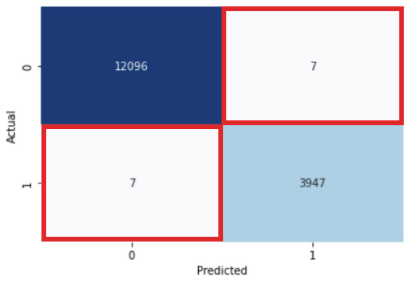
Dataset	Voting Models	PRAM (%)	K	α
WATT 1.0	IF, CBLOF & OCSVM	98.18	0.63	0.76
AMPds	IF, MCD & OCSVM	92.41	0.64	0.74
QUD	HBOS, MCD & SOD	92.63	0.78	0.75
LEAD-B147	MCD, CBLOF & OCSVM	99.13	0.84	0.94
LEAD-B936	MCD, CBLOF & OCSVM	99.17	0.87	0.98
LEAD-B1141	IF, CBLOF & OCSVM	99.66	0.91	0.90

Table 8 results showcases the high inter-rater reliability with α performances ranging from 0.74 for the Ampds dataset (tentative annotation) to 0.98 for the LEAD-B936 dataset (perfect annotation). "Tentative" annotations were achieved for WATT 1.0, AMpds, and QUD datasets with α performances between 0.74 and 0.76, while other datasets were deemed "Perfect" with α performances over 0.8. The best α performances of 0.94 and 0.98 were achieved with the LEAD-B147 and LEAD-B936 datasets, respectively, using the MCD, CBLOF, and OCSVM models, indicating high agreement despite their different paradigms.

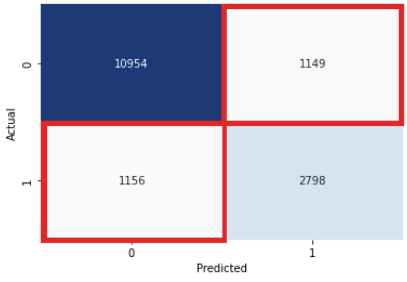
6.6 Avoiding False Alarms

It's crucial to recognize that high PRAM's performance can hide the actual performance of outlier detection. This is primarily due to the inherent rarity and scarcity of anomalies, which usually constitute a small ratio of the overall dataset. This is why we conducted an investigation to evaluate False Positives and False Negatives by extracting confusion matrices from PRAM. Due to space

limitations, we will focus on the performance of the IF model (popular outlier detection model in the SOTA) compared to the MCD model that achieved the highest performances on the QUD dataset.



(a) MCD Confusion matrix extracted from QUD.



(b) IF Confusion matrix extracted from QUD.

Fig. 3. MCD vs. IF Confusion matrices extracted from QUD results.

Figure 3 showcases that while IF achieved an Accuracy of 85.38% on the QUD dataset, it displayed a lower F1-Score of 70.83%. On the other hand, the MCD model achieved higher performances, with Accuracy and F1-Score values of 99.95% and 99.82%, respectively. Given that IF achieved a K score of 0.6 in comparison to 0.99 for MCD, results suggest that the F1-score performances are consistent with Cohen's kappa performances. The closer Cohen's kappa score is to 1, the higher the F1-score achieved. This ensures a low count of False Positives and False Negatives in energy consumption anomaly detection.

7 Discussion

In this section, we answer and discuss the questions raised on Sect. 1.

- To what extent can energy consumption anomaly annotation be completed in a fully automatic and unsupervised way?
  Results in Sect. 6.1 emphasize the high PRAM's and K's performances, demonstrating high similarity between ML models labeling and domain experts manual labeling. More precisely, MCD, OCSVM, and CBLOF models demonstrate their effectiveness for distinguishing anomalous cases from regular ones. Their performances outperformed the other popular outlier detection models in the SOTA such as IF model. The results suggest that models from the One Class Learning, Dimensionality and Clustering based paradigms benefit the energy consumption anomaly detection challenge.
- How does the anomalies contamination factor in a dataset impact on the unsupervised models performances?

Our findings indicate that as contamination ratio decreases, the ML models exhibit better performances in identifying anomalies. On the other hand, when the number of energy anomaly instances increases and anomalies are no longer in the minority class, distinguishing between normal and anomalous patterns becomes challenging for ML models.
- How do datasets preprocessing handle missing values? In the preprocessing phase, we applied linear interpolation to WATT 1.0 dataset and applied mean imputation to LEAD datasets. Results suggest that these methods can bias the data and they are ineffectual for handling missing values in this context.
- How the number of used features impact on the ML models performances? Our experiments using 3 different combinations of features showcased divergent outcomes. Specific models demonstrated performance improvements when increasing the number of used features, whereas others showed no change. The results suggest that models from the One Class learning paradigm and Clustering paradigm are more features hungry than the other paradigms. Their performances increase when increasing the number of features. Nevertheless, a deeper study is needed on features correlation.
- Can we enhance anomaly detection performance by implementing an annotation voting process?
  Results indicate that voting does not enhance the labeling task, as it is heavily reliant on individual raters' reliability.
- To what extent can automatic energy consumption anomaly detection avoid false alarms?
  Results indicate that MCD, CBLOF and OCSVM models avoid false notifications better than the other models. Compared to the IF model, which is very popular in anomaly detection state of the art, the aforementioned models are more relevant for energy consumption anomaly detection.

8 Conclusion

This study evaluates the effectiveness of 8 unsupervised anomaly detection models across 9 energy anomaly detection datasets, using PRAM, K, and α metrics for assessing their alignment with expert's manual labeling. It highlights the capabilities of OCSVM, MCD, and CBLOF models in identifying anomalies similarly to domain experts'. This research also identifies a gap in handling missing data, where traditional methods like mean imputation and linear interpolation are not relevant in the case of energy consumption anomaly detection. Results indicate that models perform better at lower anomalies contamination ratios and with an increased number of features, enhancing their ability to minimize False Positives and False Negatives. However, combining models through a voting process didn't enhance performance due to high inter-rater reliability among the models themselves.

The study has limitations and provides room for further research, including exploring the correlation between features and their suitability for different building uses (such as houses, offices, manufacturing, education, etc.). As ML models

evolve and new models are introduced, we aim to evaluate the performances of deep generative models such as Variational Autoencoder and Generative Adversarial Networks and their combination.

References

1. Fahim, M., Sillitti, A.: Anomaly detection, analysis and prediction techniques in IoT environment: a systematic literature review. IEEE Access **7**, 81664–81681 (2019)
2. Mikulová, M., Straka, M., Štěpánek, J., Štěpánková, B., Hajic, J.: Quality and efficiency of manual annotation: pre-annotation bias. In: Proceedings of the 13th Language Resources and Evaluation Conference, pp. 2909–2918. European Language Resources Association, Marseille, France (2022)
3. Zhong, L., Zhu, Y., Van Leeuwen, M.: A survey on explainable anomaly detection. ACM Trans. Knowl. Discov. Data **18**(1), 1–54 (2023)
4. Himeur, Y., Ghanem, K., Alsalemi, A., Bensaali, F., Amira, A.: Anomaly detection of energy consumption in buildings: a review, current trends and new perspectives. arXiv:2010.04560 (2020)
5. Habeeb, R.A.A., Nasaruddin, F., Gani, A., Hashem, I.A.T., Ahmed, E., Imran, M.: Real-time big data processing for anomaly detection: a survey. Int. J. Inf. Manage. **45**, 289–307 (2019)
6. Ploennigs, J., Chen, B., Schumann, A., Brady, N.: Exploiting generalized additive models for diagnosing abnormal energy use in buildings. In: Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings, pp. 1–8. Association for Computing Machinery, Roma, Italy (2013)
7. Manuel, P., Biscarri, F., Guerrero, J.I., Monedero, I., León, C.: Rule-based system to detect energy efficiency anomalies in smart buildings, a data mining approach. Expert Syst. Appl. **56**, 242–255 (2016)
8. Himeur, Y., Alsalemi, A., Bensaali, F., Abbes, A.: Building power consumption datasets: survey, taxonomy and future directions. Energy Buildings **227**, 110404 (2020)
9. Rashid, H., Batra, N., Singh, P.: Rimor: towards identifying anomalous appliances in buildings. In: Proceedings of the 5th Conference on Systems for Built Environments, pp. 33–42. Association for Computing Machinery, Shenzen, China (2018)
10. Makonin, S., Popowich, F., Bartram, L., Gill, B., Bajic, I.V.: AMPds: a public dataset for load disaggregation and eco-feedback research. Scientific Data **3**(1), 1–12 (2016)
11. Beckel, C., Kleiminger, W., Cicchetti, R., Staake, T., Santini, S.: The ECO data set and the performance of non-intrusive load monitoring algorithms. In: Proceedings of the 1st ACM International Conference on Embedded Systems for Energy-Efficient Buildings, pp. 80–89 Association for Computing Machinery, Memphis, USA (2014)
12. Murray, D., Stankovic, L., Stankovic, V.: An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study. Sci. Data **4**(1), 1–12 (2017)
13. Himeur, Y., Alsalemi, A., Bensaali, A., Amira, A.: A novel approach for detecting anomalous energy consumption based on micro-moments and deep neural networks. Cogn. Comput. **12**, 1381–1401 (2020)

14. Fu C, Arjunan P, Miller C.: Trimming outliers using trees: winning solution of the large-scale energy anomaly detection (LEAD) competition. In: Proceedings of the 9th International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, pp. 456–461 Association for Computing Machinery, Boston, USA (2022)
15. Manoj, G., Arjunan, P.: LEAD1.0: a large-scale annotated dataset for energy anomaly detection in commercial buildings. In: Proceedings of the 13th International Conference on Future Energy Systems, pp. 485–488 Association for Computing Machinery, Virtual Event (2022)
16. Himeur, Y., Ghanem, K., Alsalemi, A., Bensaali, F., Amira, A.: Artificial intelligence based anomaly detection of energy consumption in buildings: a review, current trends and new perspectives. Appl. Energy **287**, 116601 (2021)
17. Kriegel, H.P., Schubert, M., Zimek, A.: Angle-based outlier detection in high-dimensional data. In: Proceedings of the 14th SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 444–452 Association for Computing Machinery, Las Vegas, USA (2008)
18. Kriegel, H.-P., Kröger, P., Schubert, E., Zimek, A.: Outlier detection in axis-parallel subspaces of high dimensional data. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS (LNAI), vol. 5476, pp. 831–838. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01307-2_86
19. He, Z., Xu, X., Deng, S.: Discovering cluster-based local outliers. Pattern Recogn. Lett. **24**(9–10), 1641–1650 (2003)
20. Tang, J., Chen, Z., Fu, A.W., Cheung, D.W.: Enhancing effectiveness of outlier detections for low density patterns. In: Chen, M.-S., Yu, P.S., Liu, B. (eds.) PAKDD 2002. LNCS (LNAI), vol. 2336, pp. 535–548. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-47887-6_53
21. Goldstein, M., Dengel, A.: Histogram-based outlier score (HBOS): a fast unsupervised anomaly detection algorithm. KI-2012 Poster Demo Track **1**, 59–63 (2012)
22. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: Proceedings of the 8th International Conference on Data Mining, pp. 413–422. IEEE, Pisa, Italy (2008)
23. Rousseeuw, P.J., Van Driessen, K.: A fast algorithm for the minimum covariance determinant estimator. Technometrics **41**(3), 212–223 (1999)
24. Hejazi, M., Singh, Y.P.: One-class support vector machines approach to anomaly detection. Appl. Artif. Intell. **27**(5), 351–366 (2013)
25. Artstein, R., Poesio, M.: Inter-coder agreement for computational linguistics. Comput. Linguist. **34**(4), 555–596 (2008)
26. Warrens, M.J.: Five ways to look at Cohen's kappa. J. Psychol. Psychother. **5**(4), 1 (2015)
27. Gaur, M., Makonin, S., Bajić, I.V., Majumdar, A.: Performance evaluation of techniques for identifying abnormal energy consumption in buildings. IEEE Access **7**, 62721–62733 (2019)
28. Habeeb, R.A.A., Nasaruddin, F., Gani, A., Hashem, I.A.T., Ahmed, E., Imran, M.: Real-time big data processing for anomaly detection: a survey. Int. J. Inf. Manage. **45**, 289–307 (2019)
29. Saar-Tsechansky, M., Provost, F.: Handling missing values when applying classification models. J. Mach. Learn. Res. **8**, 1623–1657 (2007)

Unlocking User Privacy: A Systematic Survey of Factors and Methods in Predicting App Permission Decisions

Rena Lavranou(✉) and Aggeliki Tsohou

Ionian University, Corfu, Greece
{lavranou,atsohou}@ionio.gr

Abstract. Mobile devices have become an indispensable part of everyday life for most people and have improved our lives in many ways, offering a multitude of possibilities through the various available applications. On one hand, the risks around information privacy on mobile devices become more and more challenging. On the other hand, users are called to analyse and take plenty of complex privacy decisions. There have been many attempts to overcome this burden from users by automating the process of granting permissions to ap-plications. This study provides a literature review on works for predicting users' privacy decisions in application requests. Our research aims to shed light on the different factors that have been identified in the literature as important predictors of users' decisions on app requests, as well as the methods that have been used, focusing on machine learning approaches and building user privacy profiles.

Keywords: Application Permissions · User's Privacy Decisions Prediction · Machine Learning · Privacy Profiles

1 Introduction

With the technological developments of recent years, a large number of people are using smart devices in a regular basis, such as smartphones, and this number is increasing [10]. Users can easily access a wide range of internet services through smartphone applications (apps), such as those for communication, e-shopping, e-banking, gaming, education and health [12]. Although smart devices and mobile apps offer many benefits, they can pose serious threats to users' privacy [12]. Smart devices collect personal data, such as users' location, calendar and contacts [6], not only for supporting functionality, but also for other purposes, such as marketing and advertising, to offer personalized services to users.

Current smartphone operating systems use permission systems to control how apps access sensitive resources [28]. On Android and iOS devices, permissions to access most data, such as contacts, messages, photos, and various sensors, such as camera, microphone, and location, are granted via permission requests [12]. Mobile applications need to ask users for permission to use these resources before the application is installed

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 347–360, 2024.
https://doi.org/10.1007/978-3-031-63215-0_26

or at runtime [6]. Regulations on privacy rights, such as the General Data Protection Regulation (GDPR), require users to have control over the collection and use of their personal data, including giving informed and active consent [21].

In view of the number of apps that users typically install on their mobile phones, it is clear that they face numerous and complex privacy decisions, such as configuring the permission settings for the applications they install, the number of which can be overwhelming [24]. Studies have shown that the average user needs to make more than one hundred permission decisions per mobile device [16]. While Raber & Krueger estimate more than 470 permission decisions for an average user with 95 applications installed and an average of 5 permissions for each of them [20]. It is therefore not surprising that most users do not spend time configuring many of these settings [24]. Only a small number of users actually read the application permissions that are requested, and even fewer understand them correctly [11]. Furthermore, some users become accustomed to such requests and respond automatically, without making an informed decision [6].

Users are usually unaware of the permission settings or the context of the permissions they have granted to previously installed applications [3]. They also tend to make decisions without realising the consequences [3, 14]. In fact, many smartphone users grant data access to apps that collect a huge amount of personal and sensitive data, often much more than they really need [12]. In general, mobile users are often unaware of the data collection and sharing practices of the apps they use [3]. A typical example of a privacy violation occurring without users' knowledge is the case of Cambridge Analytica, which used a mobile app to collect private information from 50 million Facebook users to create voter profiles [22]. As a result, users of mobile devices are often surprised by the ability of various applications to collect and share personal data with third parties [4]. When they informed of their actual practices, many users become upset and even respond by uninstalling the apps [23].

Some users may simply be unaware of privacy issues, while other groups express concerns about control over their personal data. They may be highly motivated to protect their data but lack the skills and knowledge to do so [12]. There are still users that are concerned about their privacy and aware of the privacy risks, but actually do very little to protect their personal data and continue to disclose it. In other words, there is sometimes a discrepancy between users' attitudes and their behaviors. This is widely known in the literature as the "privacy paradox" [1].

Given the above, it is clear that privacy decision-making in mobile devices is particularly complex. An obvious weakness is the lack of awareness among Android users regarding the privacy of their data, which makes them vulnerable to various malicious attacks [2]. Furthermore, the current permissions systems are not well aligned with users' privacy expectations, and as a result, users often have no idea of the frequency and the conditions under which their personal data is accessed [28]. The need to reduce the user burden of complex permission decisions is highlighted in the literature [7]. The number and complexity of apps and security and privacy configurations on mobile devices require automated approaches for effective user privacy protection [18]. Thus, the automated management of privacy decisions becomes necessary to improve usability, as applications perform hundreds of permission checks per day [3].

A promising approach to mitigate the trade-off between users' privacy preferences and alleviate their burden is to use machine learning to generate setting recommendations or group some settings [24]. Many researchers propose various mechanisms to predict users' privacy decisions, using machine learning methods, based on a relatively small number of factors, such as previous privacy decisions or answers to privacy related questions [15–17]. Software agents can then use these machine learning models to make personalized privacy recommendations to users, helping them to take better control of their privacy [13], while at the same time reducing the number of decisions that users need to make themselves.

In this article, we provide an overview of the existing literature on predicting users' privacy decisions in app requests. We collect relevant studies aiming to identify the factors that have been identified for predicting users' decisions and play a crucial role in predicting users' final decision to accept or reject app requests, as well as the methods they have used, focusing on machine learning approaches and building privacy profiles.

As a guide to our research, we followed the following research questions:

- Which factors influence users' decision to accept or reject app permissions?
- How have machine learning techniques been used to predict users' privacy preferences for app permissions?

The paper is structured as follows: Sect. 2 provides an overview of the re-search methodology that we followed. Section 3 presents the various factors that have been studied in the literature in relation to predicting users' privacy decisions regarding application permissions, as well as their impact on users' final decision to accept or deny a permission. Section 4 shows how these factors have been used in the literature to create privacy profiles to predict user decisions. Section 5 presents the methods that different researchers have used to predict user preferences about app permissions, focusing on machine learning approaches. In Sect. 6, we discuss our findings and provide suggestions for future work, and Sect. 7 concludes the paper.

2 Literature Review and Analysis Methodology

For our literature analysis, we searched for articles in the academic databases Google Scholar, IEEE, Research Gate as well as the digital libraries Springer Link and ACM Digital Library. We used the keywords: application permissions, application permissions prediction, users' application permission, predicting app permissions preferences.

Subsequently, we investigated thoroughly the titles and the abstracts of the studies. Through this process, we excluded studies that, although they met the search criteria, did not focus on identifying factors that influence users' privacy decisions regarding application permissions, nor were they relevant to the prediction of such decisions. We, therefore, excluded studies that approached the issue of app permissions from a different perspective than our focus, such as a privacy risk perspective, or papers that classified apps as malicious or benign based on their permissions, as well as those that explored users' privacy awareness and concerns, the privacy paradox, or users' application selection.

Thus, we focused on 17 papers that are related to our research questions about predicting users' privacy decisions on app permissions and have been published in 2012 or later.

3 Factors Influencing Users' Application Permission Settings

Many researchers in the literature have tried to predict users' privacy decisions about app permissions. In this section we attempt to summarise all the factors that have been studied so far in predicting users' preferences for accepting or denying an app permission, as well as their impact on users' final decisions.

The first attempts to predict users' privacy decisions, such as the work by Liu et al., used only the application name (app ID), the permission type and the user ID, i.e. unique triples of the form "user, app, permission" [17]. In their subsequent work [16], Liu et al. found that significant predictors for accepting or denying permissions are application category and permission type, while information such as purpose, privacy concerns, demographics, the application name or the access frequency are not determinant for users' decisions. Similar conclusions were reached by Lin et al., who found that users tend to use different privacy controls depending on the type of private information (e.g., fine-grained location, coarse-grained location, phone states, contacts, SMS messages, and account information) to be provided [15].

Lin et al. developed a recommendation algorithm that predicts whether users would share private information with a mobile application based on the type of application, the type of private information and the purpose of the permission request i.e. application - permission - purpose triples [15]. Their analysis is also in contrast to Liu et al. [16] regarding the importance of purpose as a predictor for the users' privacy decision. Lin et al. showed that a user's willingness to grant a specific permission to a particular mobile app is strongly influenced by the purpose associated with that permission. For example, whether the source is requested for internal functionality purposes, targeted advertising from advertising and social networks, or to perform mobile analytics [15]. Likewise, Andriotis et al. found that users primarily grant or deny access to permission groups, taking into account the functionality of each app [4]. For example, they grant access to Camera permissions for the Instagram app.

In a later paper, Smullen et al. also pointed out that individuals' privacy preferences are strongly influenced by the purpose for which permissions are requested [24]. Their work was focused on three sensitive permissions, namely contacts, location, and calendar, and three general categories of purposes: internal (basic functionality), advertising (collecting and analyzing user's data) and any other unspecified or unknown purpose [24].

According to Olejnik et al. [19], contextual information is important for user preferences. In [19], they concluded that adding just one of the following four features, i.e., the name of the app requesting permission, the requested source, the method of the request (i.e., the actual API call), and whether the requesting app was in the foreground, would lead to improvements in their model results compared to using (almost) no contextual information at all. They also highlighted that the most striking of these features was the last one, i.e. whether the app was in the foreground or not.

The results of Lin et al. point in the same direction and suggest that both the purpose of using sensitive resources and the users' expectations about different applications have a major impact on users' subjective feelings and trust decisions [14]. Similarly, a study by Bonné et al. pointed out that the majority of user decisions were made by focusing on the functionality of the app and the user's expectations of whether the app really needed a particular permission [8]. Interestingly, they also found that women were twice as likely as men to deny permissions [8].

Users' opinions about the purpose of using each app were used by Biswas et al. [7]. In [7] they developed SDroid (Secured anDroid), a tool that evaluates requested permissions and allows users to selectively grant permissions by receiving a single opinion from the user about the purpose of using an app such as for social communication. This work also took into account the user's profile in terms of their level of knowledge about permissions and thus suggested the most compatible permission decision. Raber & Krueger found a significant correlation between both the user's personality, as measured by the Big Five personality scores as well as the user's privacy attitude (IUIPC questionnaire), and the application permission settings chosen by users in their study [20].

Wijesekera et al. noted that users' decisions are mainly driven by two influencing factors: privacy concerns for the specific type of data and understanding of the relevance of a permission request to the functionality of the requested application, i.e., whether the users understood why the application needed access to a resource [26]. They also showed that the visibility of the requested app (i.e., whether the user is made aware that the app is running on the device) and the frequency with which requests appear are two important factors used by users when making privacy decisions in mobile apps [26].

In a subsequent paper, Wijesekera et al. [27] confirm the importance of application visibility as a determinant of user decisions, while adding the importance of the current foreground application. It is also worth noting that [27] was the first work to use passively observed features such as locking behavior, audio settings and web browsing habits to infer future privacy decisions on a case-by-case basis at runtime. Wijesekera et al. highlighted that behavioral data that can be passively observed without the user's active involvement (i.e., without prompting) is useful for learning about the user's privacy preferences, so that a user's preferences can be predicted without asking for permission [27].

In [29], Wijesekera et al. focused more on the frequent permission requests than the rare ones, based on the "app: permission: visibility" triple, i.e., the application requesting the permission, the requested resource type, and the app visibility. They used four different contextual cues in their model to make a decision about a resource request: the name of the app requesting the permission, the app in the foreground at the time of the request (if it was different from the requesting app), the type of the permission requested (e.g., Location, Camera, Contacts), and the visibility of the requesting app (whether the user is aware that the app is running on the device) [29].

In [28], Wijesekera et al. extend their previous work [26, 27, 29] and confirm their findings, noting that users do not simply base their decisions on application name and data type. Instead, a significant proportion of participants make contextual privacy decisions. More specifically, in [28] they use contextual cues, such as what the user was doing on their phone at that time, whether the user was actively using the application requesting

their data, and whether the user knew that the app requesting the permission was running (visibility) or was in the background.

Wijesekera et al. [28] also come to the same conclusion as their previous study [27], finding that passively observable behaviors, such as mobile web usage, screen locking and call usage habits, as well as permission request runtime data (i.e., the resource requested and the time of day of the request) are predictive of whether users will grant permissions to applications [28].

The analysis by Mendes et al. shows that contextual features such as the visibility of the requesting application, the user location, and the network status have some relevance for user's privacy decisions to grant or deny permission [18]. However, the increase in predictive performance from using such features is minimal, as two features alone - the category of the requesting application and the requested permission - are able to capture a relative effect of context changes [18].

Tsai et al., with their permission manager TurtleGuard, allow users to change their decisions based on the visibility of the requesting app, which they consider to be a strong contextual cue [25]. Brandão, Mendes, & Vilela used the app category, the permission requested and its permission group to predict the permission granting outcome [9]. They also collected and based on contextual data about the phone state and the user context, including information about background and foreground running applications, network status, hour and weekday, privacy profile, whether the screen is on and the phone is unlocked (interactive mode).

Table 1 aggregates the factors that influence users' privacy decisions about permission requests. We have divided the factors into two categories, app-related and user-related factors. In the next section, we show how these factors have been used in the different studies to create users' privacy profiles as a way of predicting users' decisions on app permission requests.

4 Privacy Profiles as Predictors of Users' Application Permission Settings

Among the automation systems that either predict or recommend permission set-tings, many researchers have proposed the creation and assignment of privacy profiles, i.e., a set of predefined rules defined according to users' preferences [15, 17]. In this way, the user's personal preferences are taken into account [9]. At the same time, the amount of input required from the users is reduced, as these profiles can be assigned through a small number of questions [15, 16].

Researchers have shown that while users' privacy preferences vary, a small number of privacy profiles can predict users' decisions to allow app permissions, deny, or be prompted for app permissions with a high level of accuracy [15–17]. Lin et al. used. clustering techniques to identify privacy profiles, considering information about the purpose and users' self-reported willingness to potentially grant access, as revealed in a scenario-based online survey [15]. For their work, they used Amazon Mechanical Turk and collected privacy preferences from over 700 participants.

Specifically, Lin et al. captured the users' comfort level with granting a given permission to a particular application for a particular purpose. In this way, users' trends

Table 1. Factors influencing users' privacy decisions about app permission requests.

	Liu et al. [17]	Liu et al. [16]	Lin et al. [15]	Andriotis et al. [4]	Smullen et al. [24]	Olejnik et al. [19]	Lin et al. [14]	Bonné et al. [8]	Biswas et al. [7]	Raber & Krueger [20]	Wijesekera et al. [26]	Wijesekera et al. [27]	Wijesekera et al. [29]	Wijesekera et al. [28]	Mendes et al. [18]	Tsai et al. [25]	Brandão, Mendes & Vilela [9]
Application Factors																	
Application name (app ID)	√		√	√		√	√	√					√			√	
Application category		√	√	√	√				√						√		√
Permission type	√	√	√	√	√	√	√	√	√	√			√	√	√	√	√
Purpose			√	√	√		√	√			√						
Request's frequency											√						
Application visibility						√					√	√	√	√	√	√	√
Current foreground app												√	√	√			√
Time (hour & weekday)														√			√
Interactive mode																	√
User Factors																	
Demographics (gender)								√									
Privacy concerns											√						
User's knowledge & awareness									√	√							
User's expectation							√	√									
Privacy attitude										√							
User semantic location															√		
Network status															√		√
Passively behavior traits														√			
Privacy Profile												√					√

regarding data privacy have been grouped into 4 different privacy profiles: conservatives, unconcerned, fence sitters and advanced users. These profiles can later be used to assign the appropriate set of permissions to each individual user [15]. It is worth noting that the work of Lin et al. [15] was based on self-reported preferences, which, according to the privacy paradox, do not necessarily reflect actual privacy behavior.

On the contrary, Liu et al. collected real-world permission settings from 84 Android users worldwide [16]. Using the purpose of each permission, they successfully trained a machine learning prediction model, to derive a set of user profiles and assign each user the appropriate permission profile. Liu et al. represented the user data as a decision vector for each combination of the triple (application category, permission, purpose) and applied hierarchical clustering to generate 7 privacy profiles. They then propose to use these profiles in a personalized privacy assistant to support the user in configuring the permission settings of mobile applications [16].

Liu et al. in a previous work collected the privacy settings of 4.8 million LBE Privacy Guard users, mainly from mainland China [17]. Their work was based on users' privacy preferences of 12 permissions and used a vector with each triple combination (application, permission, decision) to represent each user. Thus, they generated 6 privacy profiles using *k*-means. With the resulting profiles, they were able to improve their predictions of users' permission preferences [17]. Biswas et al. used the users' profile in combination with their opinion about the purpose of using an application to selectively assign permissions [7]. In their work, they distinguished 3 profiles based on the knowledge level of the users: sophisticated, naive and general users [7]. Olejnik et al. grouped the users into three categories according to their privacy concerns (IUIPC scale), but as they mentioned, their dataset did not contain enough users to identify profiles by clustering [19].

Wijesekera et al. categorized users into two groups based on the denial rate of permissions prompts: the contextuals, who care about the surrounding context, and the defaulters, who don't seem to take the surrounding context into account when making privacy decisions [27]. They found that application visibility was a significant factor for users with a denial rate of 10–90%, i.e. contextuals users, but not for defaulters, i.e. users with a denial rate of 0–10% or 90–100% [27]. Andriotis et al. proposed the use of heat maps, as a visualization scheme for the representation of users' privacy profiles [4]. They suggested the creation of a privacy profile representative of each user. Each profile is a heatmap showing the percentage of apps (per device) that have been granted access to certain categories of dangerous permission groups.

Smullen et al. used machine learning to assign users to privacy profiles and then use these profiles to infer a set of permissions for each user [24]. They focused on 3 sensitive Android app permissions, calendar, location and contacts. They created agglomerative hierarchical clusters for similar individuals in their dataset and aggregated their preferences into privacy profiles. A profile is a model of (application class × permission) or, (application class × permission × purpose) recommendations. Smullen et al. [24] found that models that incorporate purpose can make more accurate predictions and also reduce the overall burden on the user, even when compared to other similar approaches [16].

The work of Mendes et al. differs from other works in that they take into account and evaluate the impact of contextual features and user expectations in combination with privacy profiles [18]. Thus, they go beyond the traditional privacy profiles created only with the requested application category and the requested permission [17], to integrate context awareness into personalization [18].

Specifically, Mendes et al. went beyond the category-permission combination, by adding features, that capture the similarity of user behavior in a more fine-grained way. So, they consider the user's expectations, as well as contextual features, such as the user's location, the visibility of the requesting application, and the network status, to build context-aware privacy profiles, the number of which was varied from 1 to 9 profiles for each combination of profiling and prediction [18].

In [9], the authors proposed a methodology to create privacy profiles and train neural networks to predict user's answers to permission requests in mobile devices, while guaranteeing user privacy even against a centralized server. Brandão, Mendes, & Vilela used the combination of category and permission to produce an average result for granting or denying a permission for each user [9]. They created the privacy profiles using hierarchical clustering and *k*-means, where each user has data points in one or more profiles. So, they decided to assign a percentage of each profile to each user. Then, to preserve the users' privacy, they used homomorphic encryption and secure aggregation, so that the server only sees the encrypted data.

5 Machine Learning and Other Artificial Intelligence Techniques in Predicting Users' Privacy Decisions

Several ways of supporting users in their privacy decisions have been proposed in the literature, including automating the configuration of privacy settings. Researchers have explored the possibility of automatically predicting and setting app permissions using various techniques, to reduce the users' burden [20]. Recent research on permission models has turned towards the use of Machine Learning (ML).

Lin et al. used machine learning techniques for data analysis, specifically clustering techniques to identify privacy profiles [15]. Liu et al. also clustered the users into privacy profiles based on their privacy preferences regarding app permissions [17]. They then used a linear support vector machine (SVM) to predict whether or not the app would access the user's sensitive data. In a later paper [16], Liu et al. again used machine learning, but this time dynamically generated decision trees, to build privacy profiles for users' permission settings. They leveraged these profiles into a Personalized Privacy Assistant (PPA) to support users in configuring permission settings for mobile applications. The profile-based recommended settings were generated by a scalable SVM classifier.

Olejnik et al. developed SmarPer, an advanced mechanism for predicting Android user permission decisions based on contextual information and machine learning methods [19]. SmarPer used a Bayesian linear regression (BLR) model and mimics the user decisions based on the decision patterns. Wijesekera et al. constructed several statistical models, such as mixed effects binary logistic regression models, to examine whether the user's desire to block certain permission requests could be predicted using contextual data [26]. A classifier could then be used to determine when to deny potentially

unexpected permission requests without user intervention, and only prompt the user for questionable data requests.

In a subsequent article [27], Wijesekera et al. incorporated elements of the environmental context into a machine learning model. They trained an offline classifier by using participants' responses to runtime prompts. Their model better approximates user decisions by finding factors relevant to users that are not encapsulated by the AOFU model. Their SVM model with an RBF kernel produced the best accuracy [27]. Later, Wijesekera et al. implemented a Support Vector Machine (SVM) classifier and integrated it into the Android operating system as a system service [29]. In [29], they implemented the first context-aware permission system that dynamically performs permission denial, which is an improvement over previous work that only performed offline learning or did not regulate permissions in real time. In [28], Wijesekera et al. built an ML model with comparable accuracy to the status quo AOFU, using only passively observable features.

Raber & Krueger used machine learning techniques to predict users' privacy settings based on their personalities [20]. In their work, they tried out SVM and several other classification methods, and achieved the best results with a KNeighbors implementation with two as the number of neighbors [20]. Smullen et al. attempted to improve privacy preference recommendation models by using privacy profiles that incorporate a combination of supervised and unsupervised machine learning (agglomerative hierarchical clusters and conditional inference trees) [24]. They also performed a logistic regression analysis to confirm the significant effect of purpose on participants' preferences for app permissions [24].

Mendes et al. attempted to automate user privacy decisions using machine learning approaches, combining privacy profiles, context-awareness and user expectation [18]. They experimented using various models from the literature, such as SVM with linear and Radial Basis Function (RBF) kernels, decision trees, bagging, ada boosting, random forest and a neural network. Furthermore, Mendes et al. built privacy profiles using the K-means clustering algorithm instead of the hierarchical clustering, traditionally used to build privacy profiles [18].

Brandão, Mendes, & Vilela present a strategy for predicting user responses to a permission request using federated learning, while preserving user privacy [9]. Federate learning provides them that possibility by training a neural network model locally, on each smartphone, using only local data, and then sharing only the neural network weights with a central server on each iteration. Their system generated privacy profiles in a secure manner, using privacy-preserving distributed hierarchical clustering and efficient privacy-preserving distributed k-means for non-IID data [9].

The use of machine learning techniques has not only enabled higher accuracy, but also helps to reduce user involvement, which is important to prevent habituation [28]. By using ML techniques, future systems can not only reduce the number of decisions users have to make, but also increase the likelihood that privacy settings will be more closely aligned with users' preferences and expectations [28].

6 Discussion

Managing information privacy is an increasingly challenging task for the average internet user, as the literature shows [24]. Users appear to be overwhelmed and increasingly unable to make informed privacy decisions due to time constraints, motivation, and cognitive decision-making skills. Many attempts have been made to overcome this user burden by automating the process of granting permissions to apps.

Researchers have studied various factors in an attempt to predict users' privacy preferences. Our research has shed light on the different factors that have been found in the literature to play a significant role in predicting users' app request decisions. From the systematic literature review we conducted, we have summarised the factors that influence individuals' privacy decisions about app permissions. We used a tabular format to present these factors, providing a structured overview.

The findings of the individual studies differ from each other in terms of the influence of the various factors on users' decisions. This is probably due to the different context of each work and the different datasets they used. There are articles that have investigated real-world data and others that have relied on self-reported data of the participating users, papers that have examined a wide range of applications and others that have focused on specific categories of applications and permissions.

One of the most widely used techniques to predict users' privacy preferences is the use of machine learning. Machine-learning (ML) techniques have shown great promise for designing systems that automatically make privacy decisions on behalf of the user [25]. Indeed, by creating privacy decision models based on machine learning, it is possible to capture people's privacy preferences more accurately while reducing the burden on users, as highlighted in the literature [24]. It has also been pointed out that ML reduces the involvement of the user and thus, minimises the habituation effect [27]. Using machine learning to automate users' privacy decisions provides a balance between accurately implementing user's privacy preferences and overburdening them with too many decisions [27].

Furthermore, the current permissions systems are not well aligned with users' privacy expectations [28]. The situation has improved with Android 6.0, where users can grant or deny permissions to apps during run time. But there is still not enough information to help users make informed decisions. It appears that users make decisions based on context, change their mind based on circumstances [28], and consider much more than the name of the application and the permission type when deciding whether to accept permission. An advantage of ML models is their ability to incorporate nuanced contextual data to predict user preferences. These approaches have shown much higher accuracy than the status quo, i.e. AOFU [25]. To improve the effectiveness of permission managers in protecting user privacy, automation is of paramount importance, which should take into account personal preferences in each context [1].

We argue that an important first step towards empowering users to make more contextual privacy decisions privacy has already been taken. However, the full extent of the impact of the surrounding context goes beyond the mere visibility of the requesting application [25]. Further work is needed to understand the different contextual factors and their respective impact on users' privacy decisions. In addition, more research should be done towards passively observable features for predicting users' privacy decisions,

that do not require user prompting [28]. More sophisticated learning techniques could potentially further improve the accuracy of such predictions, so that little to no user involvement is needed.

Understanding the factors that influence users' privacy decisions can provide a fertile ground for further research in academia. We also aim to provide guidance to researchers, as well as developers and platform designers, so that they can implement more appropriate permission-granting mechanisms, and thus, help the users make more informed privacy decisions. Increasing users' awareness of the data collected by their apps and empower them to control their privacy more effectively and make more self-protective and privacy-conscious decisions about personal data should be a high priority [5].

7 Conclusion

The number and complexity of apps and permissions on mobile devices, combined with the lack of adequate knowledge and awareness among users regarding their data privacy, require automated approaches to effective user privacy protection, such as automatic granting of permissions. In trying to predict users' privacy preferences, researchers have studied various factors and used different approaches.

We collected relevant studies to aggregate the factors that have been identified as predictive of users' decisions and that play a crucial role in predicting users' decision to accept or reject app requests, as well as the methods they have used, focusing on machine learning approaches and constructing privacy profiles. From the systematic literature review that we have conducted, we have aggregated the factors that influence individuals' privacy decisions and provided a structured overview of them, by using a tabular format.

Our work provides a basis for further research in an attempt to understand users' privacy decisions. Nonetheless, increasing users' privacy awareness and empowering them to better control their privacy by making informed consent decisions about application permissions should be the ultimate goal.

Acknowledgement. The authors gratefully acknowledge the financial support of the Ionian University, Greece.

References

1. Acquisti, A., Brandimarte, L., Loewenstein, G.: Privacy and human behavior in the age of information. Science **347**(6221), 509–514 (2015)
2. Alani, M.M.: Android users privacy awareness survey. Int. J. Interact. Mobile Technologies **11**(3) (2017)
3. Almuhimedi, H., et al.: Your location has been shared 5,398 times! a field study on mobile app privacy nudging. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, pp. 787–796, April 2015
4. Andriotis, P., Li, S., Spyridopoulos, T., Stringhini, G.: A comparative study of android users' privacy preferences under the runtime permission model. In: Tryfonas, T. (eds.) Human Aspects of Information Security, Privacy and Trust. HAS 2017. Lecture Notes in Computer Science, vol. 10292. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58460-7_42

5. Barth, S., de Jong, M.D., Junger, M., Hartel, P.H., Roppelt, J.C.: Putting the privacy paradox to the test: Online privacy and security behaviors among users with technical knowledge, privacy awareness, and financial resources. Telematics Inform. **41**, 55–69 (2019)
6. Betzing, J.H., Tietz, M., vom Brocke, J., Becker, J.: The impact of transparency on mobile privacy decision making. Electron. Mark. **30**, 607–625 (2020)
7. Biswas, S., Haipeng, W., Rashid, J.: Android permissions management at app installing. Int. J. Secur. Appl. **10**(3), 223–232 (2016)
8. Bonné, B., Peddinti, S.T., Bilogrevic, I., Taft, N.: Exploring decision making with {Android's} runtime permission dialogs using in-context surveys. In: Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017), pp. 195–210 (2017)
9. Brandão, A., Mendes, R., Vilela, J.P.: Prediction of mobile app privacy preferences with user profiles via federated learning. In: Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy, pp. 89–100, April 2022
10. Cha, Y., Pak, W.: Protecting contacts against privacy leaks in smartphones. PLoS ONE **13**(7), e0191502 (2018)
11. Felt, A.P., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D.: Android permissions: User attention, comprehension, and behavior. In: Proceedings of the Eighth Symposium on Usable Privacy and Security, pp. 1–14, July 2012
12. Gerber, N., et al.: FoxIT: enhancing mobile users' privacy behavior by increasing knowledge and awareness. In: Proceedings of the 7th Workshop on Socio-Technical Aspects in Security and Trust, pp. 53–63, December 2018
13. Lee, H., Kobsa, A.: Privacy preference modeling and prediction in a simulated campuswide IoT environment. In: 2017 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 276–285. IEEE, March 2017
14. Lin, J., Amini, S., Hong, J.I., Sadeh, N., Lindqvist, J., Zhang, J.: Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, pp. 501–510, September 2012
15. Lin, J., Liu, B., Sadeh, N., Hong, J.I.: Modeling {Users'} mobile app privacy preferences: restoring usability in a sea of permission settings. In: 10th Symposium on Usable Privacy and Security (SOUPS 2014), pp. 199–212 (2014)
16. Liu, B., et al.: Follow my recommendations: a personalized privacy assistant for mobile app permissions. In: Twelfth Symposium on Usable Privacy and Security (SOUPS 2016), pp. 27–41 (2016)
17. Liu, B., Lin, J., Sadeh, N.: Reconciling mobile app privacy and usability on smartphones: Could user privacy profiles help?. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 201–212, April 2014
18. Mendes, R., Cunha, M., Vilela, J.P., Beresford, A.R.: Enhancing user privacy in mobile devices through prediction of privacy preferences. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) Computer Security – ESORICS 2022. ESORICS 2022. Lecture Notes in Computer Science, vol. 13554. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17140-6_8
19. Olejnik, K., Dacosta, I., Machado, J.S., Huguenin, K., Khan, M.E., Hubaux, J.P.: Smarper: context-aware and automatic runtime-permissions for mobile devices. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 1058–1076. IEEE, May 2017
20. Raber, F., Krueger, A.: Towards understanding the influence of personality on mobile app permission settings. In: Human-Computer Interaction–INTERACT 2017: 16th IFIP TC 13 International Conference, Mumbai, India, 25–29 September 2017, Proceedings, Part IV 16, pp. 62–82. Springer International Publishing (2017)
21. Regulation, P.: Regulation (EU) 2016/679 of the European Parliament and of the Council. Regulation (eu) **679**, 2016 (2016)

22. Rossenberg, M., Confessore, N., Cadwalladr, C.: How Trump consultants exploited the Facebook data of million. The New York Times (2018)
23. Shklovski, I., Mainwaring, S.D., Skúladóttir, H.H., Borgthorsson, H.: Leakiness and creepiness in app space: perceptions of privacy and mobile app use. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2347–2356, April 2014
24. Smullen, D., Feng, Y., Zhang, S., Sadeh, N.M.: The best of both worlds: mitigating trade-offs between accuracy and user burden in capturing mobile app privacy preferences. Proc. Priv. Enhancing Technol. **2020**(1), 195–215 (2020)
25. Tsai, L., et al.: Turtle guard: Helping android users apply contextual privacy preferences. In: Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017), pp. 145–162 (2017)
26. Wijesekera, P., Baokar, A., Hosseini, A., Egelman, S., Wagner, D., Beznosov, K.: Android permissions remystified: a field study on contextual integrity. In: 24th USENIX Security Symposium (USENIX Security 15), pp. 499–514 (2015)
27. Wijesekera, P., et al.: The feasibility of dynamically granted permissions: aligning mobile privacy with user preferences. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 1077–1093. IEEE, May 2017
28. Wijesekera, P., et al.: Dynamically regulating mobile application permissions. IEEE Secur. Priv. **16**(1), 64–71 (2018)
29. Wijesekera, P., et al.: Contextualizing privacy decisions for better prediction (and protection). In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pp. 1–13, April 2018

Author Index

Published by Springer Nature Switzerland AG 2024
I. Maglogiannis et al. (Eds.): AIAI 2024, IFIP AICT 712, pp. 361–362, 2024.
https://doi.org/10.1007/978-3-031-63215-0