KALLEB MORAIS DE MOURA ABREU

EXPLAINABLE MACHINE LEARNING FOR EFFECTIVE ALARM PREDICTION

Dissertation submitted to the Computer Science Graduate Program of the Universidade Federal de Viçosa in partial fulfillment of the requirements for the degree of *Magister Scientiae*.

Adviser: André Gustavo dos Santos

Co-advisers: Julio Cesar Soares dos Reis

VIÇOSA - MINAS GERAIS
2023

Ficha catalográfica elaborada pela Biblioteca Central da Universidade Federal de Viçosa - Campus Viçosa

T
A162e
2023

Abreu, Kalleb Morais de Moura, 1995-
Explainable machine learning for effective alarm prediction / Kalleb Morais de Moura Abreu. – Viçosa, MG, 2023.
1 dissertação eletrônica (65f.): il. (algumas color.).

Texto em inglês.
Orientador: André Gustavo dos Santos.
Dissertação (mestrado) - Universidade Federal de Viçosa, Departamento de Informática, 2023.
Referências bibliográficas: f. 62-65.
DOI: https://doi.org/10.47328/ufvbbt.2024.242
Modo de acesso: World Wide Web.

1. Aprendizado do computador. 2. Alarmes. 3. Análise por agrupamento. 4. Modelos e construção de modelos. I. Santos, André Gustavo dos, 1974-. II. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 22. ed. 006.31

Bibliotecário(a) responsável: Bruna Silva CRB-6/2552

KALLEB MORAIS DE MOURA ABREU

EXPLAINABLE MACHINE LEARNING FOR EFFECTIVE ALARM PREDICTION

Dissertation submitted to the Computer Science Graduate Program of the Universidade Federal de Viçosa in partial fulfillment of the requirements for the degree of *Magister Scientiae*.

APPROVED: December 20, 2023.

Assent:

Documento assinado digitalmente
gov.br
KALLEB MORAIS DE MOURA ABREU
Data: 11/07/2024 21:33:40-0300
Verifique em https://validar.iti.gov.br

Kalleb Morais de Moura Abreu
Author

Documento assinado digitalmente
gov.br
ANDRE GUSTAVO DOS SANTOS
Data: 11/07/2024 22:11:20-0300
Verifique em https://validar.iti.gov.br

André Gustavo dos Santos
Adviser

To my beloved family, who have always been my pillars of support and the source of my strength - thank you for your unwavering love, encouragement, and sacrifice. This dissertation is dedicated to you.

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to the following individuals and institutions, without whom this work would not have been possible:

To my family, for their unwavering support and encouragement throughout my life. Your love and belief in me have been my constant inspiration.

To my advisors, whose trust in my work and invaluable guidance have shaped this research. Your expertise and dedication have been instrumental in its completion.

I am grateful to the Federal University of Viçosa for providing me with the opportunity to pursue this postgraduate course and for the resources made available for my research.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

To the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), to granting the scholarship.

Special thanks to Coopservice for providing the data necessary for the execution of this work.

Lastly, I extend my sincere appreciation to all those who directly or indirectly contributed to the execution of this work.

“The important thing is not to stop questioning; curiosity has its own reason for existing.”

(Albert Einstein)

RESUMO

ABREU, Kalleb Morais de Moura, M.Sc., Universidade Federal de Viçosa, dezembro de 2023. **Explainable machine learning for effective alarm prediction**. Orientador: André Gustavo dos Santos. Coorientador: Julio Cesar Soares dos Reis.

Esta dissertação avalia doze modelos de aprendizado de máquina para a previsão de alarmes utilizando agrupamento geográfico, por meio dados de uma empresa italiana. Os modelos abrangem uma variedade de algoritmos, incluindo Naive Bayes (NB), XGBoost (XGB) e Perceptron Multicamadas (MLP), combinados com técnicas de codificação como Label/Ordinal Encoding (LOE) e Label/Ordinal/One-Hot Encoding (L2OE), e metodologias de agrupamento, nomeadamente Coopservice-2022 (COOP) e K-Means++ (KPP). O XGB destaca-se como o mais eficaz, proporcionando os maiores valores de AUC entre os modelos. Ajustes nas técnicas de codificação demonstram melhorias significativas para NB e MLP, com um impacto marginal para o XGB. A otimização de hiperparâmetros para modelos XGB revela que os valores padrão superam configurações variadas. As análises de valores SHAP destacam a influência significativa de atributos como um cluster específico e hora do dia. Experimentos de transferência de aprendizado confirmam a adaptabilidade do modelo entre províncias italianas, ressaltando a necessidade de monitoramento contínuo devido à sensibilidade aos rótulos de cluster. Desafios surgem ao lidar com desequilíbrios nos conjuntos de dados, impactando as previsões da classe minoritária de alarmes. Este trabalho estabelece uma base para futuras pesquisas sobre abordagens específicas para lidar com conjuntos de dados desequilibrados e algoritmos de uma única classe. O estudo advoga pela validação contínua em diversas províncias, enfatizando análises detalhadas e melhorias na robustez do modelo.

Palavras-chave: Alarmes; Aprendizado de máquina; Agrupamento; Explicabilidade de modelos; Transferência de aprendizado.

ABSTRACT

ABREU, Kalleb Morais de Moura, M.Sc., Universidade Federal de Viçosa, December, 2023. **Explainable machine learning for effective alarm prediction**. Adviser: André Gustavo dos Santos. Co-adviser: Julio Cesar Soares dos Reis.

This dissertation evaluates twelve machine learning models for the prediction of alarms using geographical clustering, leveraging data from an Italian company. The models encompass a spectrum of algorithms, including Naive Bayes (NB), XGBoost (XGB), and Multilayer Perceptron (MLP), coupled with encoding techniques such as Label/Ordinal Encoding (LOE) and Label/Ordinal/One-Hot Encoding (L2OE), and clustering methodologies, namely Coopservice-2022 (COOP) and K-Means++ (KPP). XGB emerges as the most effective, yielding the highest AUC values across models. Adjustments in encoding methods show significant improvements for NB and MLP, with a marginal impact for XGB. Hyperparameter tuning for XGB models reveals default values outperform varied configurations. The SHAP value analyses emphasize the significant impact of a specific cluster and hour of the day. Transfer learning experiments confirm the model's adaptability across Italian provinces, with continuous monitoring essential due to sensitivity to cluster labels. Challenges arise in handling dataset imbalances, impacting minority alarm class predictions. This work sets a foundation for further research on specific approaches for dealing with imbalanced datasets and one-class algorithms. The study advocates for ongoing validation across diverse provinces, emphasizing nuanced analyses and improvements in model robustness.

Keywords: Alarms; Machine learning; Clustering; Explainable models; Transfer learning.

LIST OF ILLUSTRATIONS

LIST OF TABLES

SUMMARY

1 INTRODUCTION

Alarm systems provide vital protection for individuals and valuable assets, acting as an integral part of modern society. They play a significant role in reducing the risk of burglary and theft, serving as a strong deterrent against potential intruders (Rutgers University, 2009). By promptly alerting security personnel or law enforcement authorities, alarms enable swift responses to security breaches, thereby enhancing overall safety and security. This not only acts as a deterrent for potential criminals and mitigates ongoing crimes but also enhances the probability of proactively preventing the occurrence of crimes (Rutgers University, 2009).

Alarm systems indicate events that occur at a specific moment, and sometimes it's not possible to take immediate action due to the time needed for a resolution. Therefore, alarm prediction can be a valid aid for companies that have to manage patrols because, by assigning the guard to a certain area, that has a risk of alarm, it is possible to reduce the intervention time. This is the reason why efforts have been made to apply machine learning techniques for alarm prediction in different types of scenarios.

In this context, several initiatives are focusing on developing machine learning models to improve the accuracy and efficiency of alarm prediction (MENG; KWOK, 2012; AU-YEUNG et al., 2019; QUINN, 2020; ZHUANG et al., 2020). By leveraging machine learning algorithms, these approaches enhance security measures and optimize resource allocation. For example, in network systems, they detect and predict network failures, enabling proactive response (ZHANG; WANG, 2019; LATEANO et al., 2023). In industries, they identify equipment failures and anomalies, facilitating timely maintenance and optimizing resource allocation (ZHU et al., 2016). In sum, these studies have yielded promising results and provided valuable insights for stakeholders involved in alarm-related activities. While previous studies in alarm prediction have made valuable contributions, they often overlook the geographic location of alarms and have limited exploration of model explainability for their models. In our work, we aim to fill this gap by explicitly considering the geographic location of alarms and placing emphasis on the explainability of the machine learning models employed.

In this work, we explore data from Coopservice[1], a large service provider

[1] <https://www.coopservice.it/>

company located in Italy, that offers a range of services including logistics, transportation, cleaning, maintenance, and security. The company also provides car patrolling services in several Italian provinces, utilizing a cluster system to efficiently serve its customers. Each cluster is assigned to a patrol that performs daily routes and executes the required services. Coopservice is committed to optimizing its routes and improving the efficiency and quality of its patrol services, as outlined in (ZUCCHI et al., 2022). The available dataset from Coopservice provides a suitable scenario, offering approximately three years of alarm occurrence data and their locations, spanning different provinces in the Italian territory.

When an alarm occurs, Coopservice needs to send a patrol to investigate the cause of the alarm. Since the alarms are not triggered according to a schedule, they often impact the routes defined by the company. To minimize the impact of these alarms on the defined routes, a study of alarm prediction within each cluster of the company is necessary, to take into consideration the probability of an alarm call incidence in a given region when generating routes. Due to the development of increasingly robust machine learning algorithms in solving prediction problems, this work addresses the use of such tools as a means to provide alarm prediction within Coopservice's operations.

1.1 Motivation

The motivation behind this dissertation comes from the increasing demand for efficient and effective security patrol operations in various industries, including the security services provided by Coopservice. Furthermore, one of the main challenges for alarm prediction is that their causes are diverse and unplanned. Alarms can occur at any time, and their frequency and distribution can vary significantly over time. Traditional patrol strategies often rely on reactive approaches, where patrol units respond to alarms after they occur. This reactive approach can lead to inefficiencies in resource allocation and response times. By leveraging predictive analytics and optimization techniques, there is an opportunity to transform patrol operations into proactive and preventive activities. Predicting alarm occurrences and optimizing patrol routes based on these predictions can significantly improve response times, reduce false alarms, and enhance overall security services. Last, the proposed solution not only addresses the specific challenges within Coopservice but also holds potential for broader application across diverse industries. By leveraging predictive analytics and optimization, the flexible framework can be adapted to various contexts, offering a scalable and adaptable approach to enhance security services beyond alarm prediction.

1.2 Hypotheses

In this section, we outline the hypotheses that guide our research and provide a framework for our analysis. The following hypotheses are formulated based on the problem statement and the objectives of our study:

Hypothesis 1: The occurrence of alarms is influenced by temporal factors such as the day of the week, month, and hour of the day.

Hypothesis 2: Implementing a classical clustering algorithm such as K-Means++ can lead to improved performance in predictive models compared to the patrolling practices currently employed by Coopservice.

Hypothesis 3: There are global characteristics in the context of alarm prediction that allow extending a regional model to other regions.

These hypotheses serve as a foundation for our analysis and will be tested and validated in the subsequent sections of this study.

1.3 Objectives

The primary objective of this research is to predict alarms within a time interval based on clusters. To achieve our primary objective, we set specific objectives for our study, which include:

Objective 1: Develop an understanding of the alarm patterns and factors influencing their occurrence within Coopservice's operational environment. This involves analyzing temporal factors such as the day of the week, month, and hour of the day.

Objective 2: Identify and compare different clustering approaches to define patrol zones within Coopservice's operations. This includes evaluating the existing clusters based on Coopservice's daily practices and applying the K-Means++ algorithm to identify alternative cluster formations.

Objective 3: Develop and evaluate machine learning models for alarm prediction based on historical alarm data. The models incorporate positive instances (alarms) and negative instances (non-alarms) to improve prediction performance.

Objective 4: Assess key features for alarm prediction, prioritize crucial ones, and explain the best model using methods like SHAP (SHapley Additive exPlanations) for clarity.

Objective 5: Investigate the applicability of the machine learning models trained in one province for predicting alarms in other provinces. Explore the potential of transfer learning to adapt and optimize models across different operational environments, thereby

enhancing the generalizability and scalability of the predictive system.

By achieving these objectives, this research aims to contribute to the field of alarm prediction and patrol route optimization, providing Coopservice with valuable insights and tools to enhance its operational efficiency and service delivery.

1.4 Contributions

The main contributions of this dissertation are summarized as follows:

- A model capable of predicting alarms in clusters.
- An analysis of the influence of clustering models on predicting events in clusters.
- An analysis of the impact of encoding strategies on the final results of predictive models.
- An investigation of the explainability of the models.
- An investigation of the impact of transfer learning from one province to another.

1.5 Dissertation Structure

The structure of this dissertation unfolds as follows. In Chapter 2, we present a theoretical foundation, delving into existing information relevant to our research domain. Chapter 3 builds upon this foundation by specifically examining related works to our research topic. Chapter 4 addresses the core elements of our study, formally articulating the challenges at hand and introducing the dataset that forms the basis of our analysis. Following this, Chapter 5 provides an overview of our methodology, encapsulating the problem features, machine learning models employed, the chosen scoring metric, and the framework for explainability. Chapter 6 covers transfer learning and discusses the adaptability of our models across different provinces. Last, Chapter 7 summarizes our findings and contributions and suggests potential avenues for future research.

2 THEORETICAL FOUNDATION

This chapter looks at the parts of our predictive modeling method. It starts with a study of clustering in Section 2.1, explaining the way used to find patterns in Coopservice's operational dataset. After this, Section 2.2 covers the variety of algorithms used for predictive analytics, showing their roles and help.

Subsequently, attention shifts to Section 2.3 where the evaluation landscape is dissected. The Confusion Matrix, ROC Curve, and AUC are important points, giving different views needed for improving predictive skills. Section 2.4 then goes into Blocked Cross-Validation, showing the details of a special way made to deal with time-series data.

Lastly, Section 2.5 brings interpretability to the forefront. Grounded in cooperative game theory, Shapley's values illuminate the contribution of each feature, demystifying the black box of machine learning models. This section encapsulates a comprehensive journey through the core pillars of our predictive analytics framework.

2.1 Clustering

Cluster analysis, commonly known as clustering, is a vital task in machine learning categorized under unsupervised learning. Its objective is to group a collection of objects into clusters, where objects within the same cluster exhibit higher similarity compared to those in different clusters (ROKACH; MAIMON, 2005). This technique allows the identification of inherent patterns or structures in data without relying on pre-existing labels or classifications. Cluster analysis finds numerous applications across various domains, including:

- **Customer Segmentation:** It is possible to cluster customers based on their purchases (KANSAL et al., 2018). This information is useful for adapting products to the market and targeting marketing campaigns for each segment.
- **Anomaly Detection:** Cluster analysis can be used to identify unusual patterns in the data, assisting in the detection of anomalies and atypical behaviors (AHMED; MAHMOOD; HU, 2016).

- **Semi-supervised Learning:** When the dataset has only a few labeled instances, cluster analysis can be employed to propagate these labels to all instances within the same cluster (CHAPELLE; WESTON; SCHöLKOPF, 2002). This allows for more efficient utilization of the available labels.
- **Image Segmentation:** Cluster analysis can be applied to segment an image into regions with similar characteristics, facilitating tasks related to image analysis and processing (NAZ; MAJEED; IRSHAD, 2010).

2.1.1 K-Means and K-Means++

The K-Means clustering algorithm is a method used to group a set of n observations into k clusters, where each observation belongs to the cluster with the nearest mean or centroid. The algorithm starts by randomly selecting the centroids in the data space. It then iteratively assigns each observation to the cluster whose centroid is closest and updates the centroids based on the new assignments. This process continues until the centroids no longer move significantly. (ROKACH; MAIMON, 2005)

The algorithm can be summarized in three steps as follows:

1. Initialization:
   - Randomly select k initial cluster centroids.
2. Iterative Process:
   - Assign each data point to the nearest centroid based on Euclidean distance.
   - Update the centroids by calculating the mean of the data points assigned to each centroid.
3. Convergence:
   - Repeat the assignment and centroid update steps until convergence or a predefined number of iterations.

Although the K-Means algorithm is guaranteed to converge, it is important to note that the solution it converges to may not be the global optimum. The quality of the solution depends on the initialization of the centroids. In practice, the algorithm can converge to suboptimal solutions if the random initialization step is not favorable. Considering that the result of K-Means is very sensitive to the initial centroids chosen, (ARTHUR; VASSILVITSKII, 2007) proposed K-Means++ as an optimized way of initializing centroids.

The K-Means++ algorithm improves the initial selection of cluster centers in the K-Means clustering process by selecting centers that are well spread out (ARTHUR; VASSILVITSKII, 2007) . This approach aims to enhance the convergence and quality of the final clustering result. Below are the necessary steps of the algorithm to improve the initial selection:

1. **Randomly choose the first center:** select the first cluster center uniformly at random from the available data points.
2. **Calculate distances and assign weights:** for each remaining data point, calculate the distance to the nearest already chosen center. Assign a weight to each data point, proportional to the nearest center.
3. **Select the next center:** choose the next cluster center randomly, but with a higher probability for points that have a larger squared distance. This probability is determined by the assigned weights.
4. **Repeat until k centers are chosen:** repeat steps 2 and 3 until k centers have been chosen for the clusters.

2.2 Machine Learning Strategies

Among the various machine learning approaches available, we present in the following sections the Naive Bayes (NB), the eXtreme Gradient Boosting (XGB), and the Multilayer Perceptron Classifier (MLP).

2.2.1 Naive Bayes (NB)

The Naive Bayes classifier is a probabilistic classifier based on applying Bayes' theorem with strong independence assumptions. It assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature. Although this assumption of independence (RISH et al., 2001) is generally unrealistic, Naive Bayes often competes well with more sophisticated classifiers. This makes it a simple and efficient classifier, especially in situations where the independence assumption holds reasonably well.

As described in (ZHANG, 2004) Bayes' theorem establishes a relationship between the class variable y and the dependent feature vector x_1 through x_n:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y)}{P(x_1, ..., x_n)} \tag{2.1}$$

To simplify the calculation, the naive Bayes classifier assumes a naive conditional independence assumption:

$$P(x_i|y, x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) = P(x_i|y) \tag{2.2}$$

Taking advantage of the constant value of $P(y|x_1, ..., x_n)$, the classification rule becomes:

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.3}$$

Therefore, the predicted class $\hat{y}$ is given by:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.4}$$

Estimating $P(y)$ and $P(x_i|y)$ can be achieved through Maximum A Posteriori (MAP) estimation. The relative frequency of class y in the training set serves as an estimate for $P(y)$. The different variants of the naive Bayes classifier arise from the assumptions made about the distribution of $P(x_i|y)$.

The Naive Bayes classifier finds applications in various domains such as text classification, spam filtering, and sentiment analysis due to its simplicity and good performance. Naive Bayes classifiers vary primarily based on the assumptions they make about the distribution of $P(x_i|y)$. We utilize the Gaussian distribution for classification purposes, where $P(x_i|y)$ is defined as follows:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \cdot \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{2.5}$$

2.2.2 eXtreme Gradient Boosting (XGB)

The eXtreme Gradient Boosting (CHEN; GUESTRIN, 2016) is a powerful machine learning algorithm used for a variety of tasks such as classification and regression. It works by creating many decision trees and combining their predictions to make a final prediction. Each tree is trained on a subset of the data, and the algorithm tries to learn from the errors made by the previous trees. XGB is known for its speed and accuracy, as it uses a variety of techniques to optimize the training process and handle large datasets efficiently. These techniques include sparsity-aware algorithms for dealing with sparse data, weighted quantile sketches for approximate tree learning, and cache-aware access patterns to reduce memory usage. The mathematical description of XGB (CHEN; GUESTRIN, 2016) involves the following key components:

- **Objective Function:** XGB aims to minimize a specific objective function that captures the trade-off between model complexity and accuracy. The objective function is defined as the sum of a loss function and a regularization term. The loss function measures the discrepancy between predicted and actual values, while the regularization term controls the complexity of the model to prevent overfitting.
- **Gradient Boosting Framework:** XGB builds an ensemble of weak learners (decision trees) in a sequential manner using the gradient-boosting framework. Each decision tree is trained to correct the errors made by the previous trees. The prediction of the ensemble is the weighted sum of predictions from all individual trees.
- **Tree Construction:** XGB greedily constructs decision trees, using a technique called gradient boosting. At each step, it calculates the gradient of the loss function to the predicted values and uses this information to fit a new tree. The algorithm iteratively adds new trees, optimizing the objective function.
- **Regularization:** XGB incorporates regularization techniques to control the complexity of the model and avoid overfitting. It includes parameters such as maximum depth of trees, minimum child weight, and regularization terms like L1 and L2 regularization. These parameters help in preventing the trees from growing too deep or too complex.
- **Feature Importance:** XGB provides a measure of feature importance, which indicates the relative importance of each input feature in the prediction process. This measure is computed based on the number of times a feature is used in the trees and the improvement in the objective function achieved by splitting on that feature.

Overall, XGB combines the strengths of gradient boosting and regularization techniques to build accurate and robust predictive models. It has gained popularity in various domains and machine learning competitions due to its effectiveness and efficiency (HSU et al., 2022).

2.2.3 Multilayer Perceptron (MLP)

The multilayer perceptron (MLP) (SUTER, 1990) belongs to the category of fully connected feedforward artificial neural networks (ANNs). Although the term MLP can refer to any feedforward ANN, it generally describes networks composed of multiple layers of perceptrons that have threshold activation. When MLPs have a single hidden layer, they are commonly referred to as “vanilla” neural networks. These neural networks

consist of at least three layers: an input layer, a hidden layer, and an output layer. Each node in the network, excluding the input nodes, is a neuron that employs a non-linear activation function. MLPs are trained using a supervised learning technique called backpropagation or reverse mode of automatic differentiation, which utilizes a chain rule. This technique uses multiple layers and non-linear activation to enable the network to distinguish non-linearly separable data. MLPs have widespread applications in various fields, including finance (WEYTJENS; LOHMANN; KLEINSTEUBER, 2021), healthcare (BUTT et al., 2021), image (TU et al., 2022) and speech recognition (ZHU et al., 2005), and more.

2.3 Metrics

For the machine learning models, our evaluation uses three metrics: the confusion matrix metrics, the ROC (Receiver Operating Characteristic) curve, and AUC (Area Under the Curve). The confusion matrix dissects the model's performance, breaking down true positives, true negatives, false positives, and false negatives, offering nuanced insights for targeted improvements. The ROC curve, a graphical portrayal of sensitivity versus specificity, illuminates the model's dexterity across various decision thresholds, aiming for the upper-left corner for optimal performance. Complementing this, the AUC quantifies overall performance, with 1 denoting perfection and 0.5 equating to randomness. Integrating these metrics guides informed decisions for refining our classification model's predictive accuracy.

2.3.1 Confusion Matrix

The confusion matrix is a useful tool for evaluating the performance of a classification model. It provides a tabular representation of the model's predictions compared to the actual class labels. The matrix is typically organized into four quadrants, as shown in Table 1.

Table 1 – Confusion matrix.

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

In the context of a confusion matrix, the rows correspond to the actual class labels, representing the ground truth. On the other hand, the columns represent the predicted class labels, indicating the model's output. Through an examination of the sections of the matrix, we gain insights into the performance and accuracy of the classification model by observing the various combinations of true and false predictions.

The correct predictions are reflected in the diagonal elements (TP and TN), whereas the off-diagonal elements (FP and FN) denote the incorrect predictions.

The confusion matrix provides valuable information for evaluating the performance of a classification model. It is used to calculate various evaluation metrics such as accuracy, precision, recall, and F1 score. These metrics help assess the model's ability to correctly classify instances from different classes.

By analyzing the confusion matrix, we can gain insights into the strengths and weaknesses of the classification model and make informed decisions for model improvement or adjustment of classification thresholds.

2.3.2 Evaluation Metrics

The confusion matrix is essential for evaluating classification model performance. Metrics derived from it, such as precision, recall, and F1-score provide insights into the model's effectiveness. They offer a nuanced assessment of the model's strengths and weaknesses. The most common metrics derived from the confusion matrix are listed below:

- **Accuracy:** The accuracy metric represents the proportion of correct predictions compared to the total number of predictions made by the model:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- **Precision:** Precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall (Sensitivity or True Positive Rate):** Recall calculates the proportion of correctly predicted positive instances out of all actual positive instances:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **Specificity (True Negative Rate):** Specificity measures the proportion of correctly predicted negative instances out of all actual negative instances:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

- **F1-score:** The F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

In scenarios where the dataset exhibits class imbalance, the F1-score is often preferred over accuracy as a performance metric. This is because accuracy alone can be misleading when the class distribution is uneven. In such cases, a classifier that simply predicts the majority class for all instances can achieve high accuracy, even though it fails to capture the minority class. The F1-score takes into account both precision and recall, which are computed based on true positive, false positive, and false negative rates. It provides a balanced measure of the model's performance by considering the trade-off between precision (the ability to correctly identify positive instances) and recall (the ability to capture all positive instances). By incorporating both precision and recall, the F1-score provides a comprehensive evaluation of a classifier's effectiveness in handling imbalanced datasets.

2.3.3 ROC and AUC

The ROC (Receiver Operating Characteristic) curve is a graphical representation of the performance of a binary classification model. It illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity) at various classification thresholds.

To construct the ROC curve, the model's predictions are sorted by their probability scores, and the threshold for classifying instances as positive or negative is varied. At each threshold, the true positive rate (TPR) and false positive rate (FPR) are calculated.

The AUC (Area Under the Curve) is a widely used metric in binary classification tasks. It quantifies the probability that a randomly selected positive instance will receive a higher predicted probability than a randomly selected negative instance. A perfect classifier achieves an AUC value of 1, indicating that all positive instances are ranked higher than negative instances. Conversely, a random classifier yields an AUC value of 0.5, implying that the model's predictions are no better than random chance. The AUC is a valuable measure of the model's discriminatory power and finds applications in various domains, including medical diagnostics and credit risk assessment.

The ROC curve and AUC are particularly useful when evaluating models in imbalanced datasets or when the misclassification costs of different classes are different, as emphasized by (BHOWAN; JOHNSTON; ZHANG, 2011). They provide a comprehensive assessment of the model's discriminatory power across different classification thresholds.

In Figure 1 we have an example of the ROC curve. The AUC values can be interpreted as follows:

- **AUC = 1:** Perfect classifier, all positive instances are ranked higher than negative

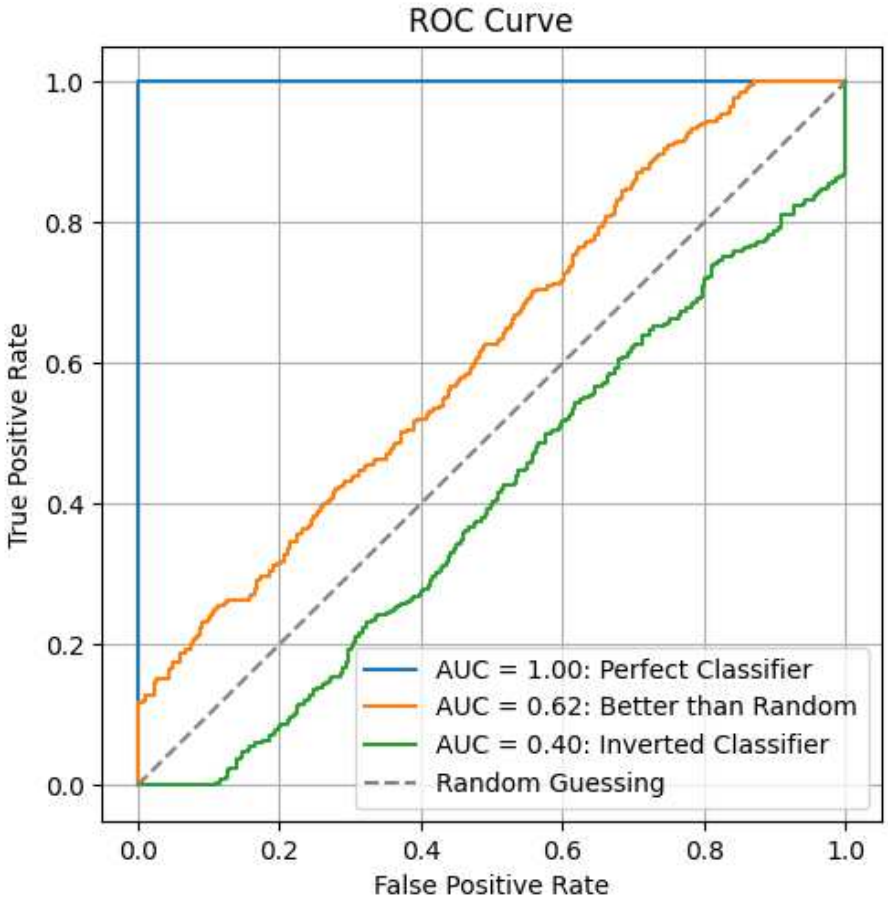


Figure 1 – The graphs show the ROC curve of 4 different classifiers.

instances.

- **AUC > 0.5:** Better than random, the model has some discriminative power.
- **AUC = 0.5:** Random classifier, the model performs as well as random guessing.
- **AUC < 0.5:** Inverted classifier, the model performs worse than random guessing.

2.4 Blocked Cross-Validation

Cross-validation stands as a pivotal technique for evaluating machine learning models, offering a robust estimate of their generalization capabilities to unseen data. In traditional methods like k-fold cross-validation, datasets are randomly split into training and validation sets. However, in the context of time series data, like the sequential occurrence of alarms, the temporal order becomes crucial. Disregarding this order can result in overly optimistic performance estimates.

To address this concern, we adopt a specialized technique known as Blocked Cross-Validation (SNIJDERS, 1988). This approach maintains the temporal order by

dividing the data into sequential blocks or folds. Each fold comprises a contiguous segment of the time series, ensuring that the validation set contains data only from later periods than the corresponding training set.

The concept of time series splits involves dividing the training set into two folds at each iteration, ensuring the validation set is consistently ahead of the training set. This respects the dependence inherent in time series data. However, this approach might introduce leakage from future data to the model, as it observes and potentially memorizes future patterns. Blocked cross-validation addresses this by adding margins at two positions. The first is between training and validation folds to prevent the model from observing lag values used both as a regressor and a response. The second is between folds used at each iteration to prevent the model from memorizing patterns from one iteration to the next.

A prevalent method for time series cross-validation is the rolling window approach. Here, a fixed-size window traverses the time series, creating consecutive folds. At each iteration, the window advances by a fixed number of time steps. This process allows the model to be trained on past data and validated on future data, simulating real-world scenarios where predictions are made on unseen future events.

Employing time series cross-validation enables a more realistic evaluation of our alarm detection model's performance. It offers insights into the model's ability to handle temporal patterns and identifies potential issues, such as overfitting or underperformance during specific periods, in a meaningful manner.

2.5 SHapley Additive exPlanations (SHAP)

Interpreting the predictions of complex machine learning models has become increasingly important in many applications, where understanding why a certain prediction was made can be just as crucial as the accuracy of the prediction itself. However, the highest accuracy for modern datasets is often achieved by complex models that are difficult to interpret, creating a tension between accuracy and interpretability. To address this issue, various methods have been proposed recently to help interpret the predictions of complex models, including the SHapley Additive exPlanations (SHAP) framework by (LUNDBERG; LEE, 2017).

SHAP assigns an importance value to each feature for a particular prediction and unifies six existing methods into a new class of additive feature importance measures. The framework provides a unique solution with desirable properties and has theoretical results to support it. In addition to what has been mentioned previously, the SHAP framework is a widely popular method for improving the interpretability of machine learning models and has been explored in previous efforts such as (LI, 2022),

(EKANAYAKE; MEDDAGE; RATHNAYAKE, 2022), and (MANGALATHU; HWANG; JEON, 2020) to deal with the black-box nature of machine learning models in different contexts.

The SHAP framework is based on the concept of Shapley values from cooperative game theory. Shapley values allocate a fair distribution of the total contribution among players in a cooperative game. In the context of machine learning interpretability, SHAP extends this idea to allocate the contribution of each feature to the prediction for a specific instance.

The key idea behind SHAP is to consider all possible coalitions of features and measure their impact on the prediction. By doing so, SHAP captures the interactions and dependencies among features, providing a more comprehensive understanding of the model's decision-making process.

SHAP assigns an importance value to each feature, indicating its contribution to the prediction compared to a baseline reference. The importance values can be positive or negative, representing whether a feature's presence increases or decreases the prediction, respectively.

One of the strengths of SHAP is its ability to handle both additive and non-additive models. For additive models, SHAP provides global feature importance scores that sum up the model's output. For non-additive models, SHAP offers local feature importance scores specific to each instance, considering the interactions between features.

The SHAP framework has found extensive applications in various domains, enabling interpretable machine-learning models in complex scenarios. Some of the notable applications include:

- **Healthcare:** SHAP has been utilized to interpret predictive models in healthcare (MOHANTY; MISHRA, 2022), providing insights into the factors driving individual predictions and aiding in clinical decision-making processes.
- **Finance:** In the financial industry, SHAP has been employed to interpret credit risk models (GRAMEGNA; GIUDICI, 2021), allowing financial institutions to understand the factors influencing creditworthiness and explain model predictions to customers.
- **Natural Language Processing:** SHAP has been applied to interpret models in natural language processing tasks (ZHAO et al., 2020), such as sentiment analysis, fake news detection (REIS et al., 2019), and text classification, shedding light on the most influential features for predicting sentiment or categorizing text.

The SHapley Additive exPlanations (SHAP) framework offers a powerful solution

for interpreting complex machine learning models. By assigning importance values to features and capturing their interactions, SHAP provides valuable insights into the decision-making process of the models. With its wide applicability and robust theoretical foundation, SHAP has gained popularity in various domains, enabling interpretable machine learning models in scenarios where accuracy and interpretability are equally important.

3 RELATED WORK

This chapter presents some related works in the field of alarm prediction. There are several stakeholders interested in alarm-related works, as we will see, to minimize the time between the alarm and remedial action or implement preventive measures and avoid unnecessary operational costs with the handling of false alarms. We can group the machine learning research lines applied to alarms into two sets: (i) filtering and (ii) prediction.

Within the (i) filtering research works, we observe studies developed to identify false alarms. Meng e Kwok (2012) addresses the issue of false alarms in intrusion detection systems (IDSs) by proposing an adaptive false alarm filter using machine learning. The study compares six machine learning schemes, highlighting their varying performance, and introduces an adaptive filter that selects the best algorithm based on network contexts. The evaluation demonstrates the effectiveness of the adaptive filter, achieving a stable reduction rate of over 80% in false alarms. The proposed approach provides a practical solution to enhance the accuracy and efficiency of intrusion detection systems in real deployment scenarios.

Au-Yeung et al. (2019) introduces the issue of false alarms generated by bedside monitors in the ICU. The methodology involves signal processing, feature extraction, and optimized machine learning, with a focus on reducing false arrhythmia alarms. The approach incorporates signal quality indices (SQIs) and arrhythmia-specific features, applying Random Forest for classification. Using the PhysioNet Challenge 2015 dataset, the proposed method achieved the highest score in the real-time category on the hidden test set. The work emphasizes the importance of domain knowledge in feature design, the optimization of machine learning algorithm performance through hyperparameter tuning, and the potential of advanced algorithms and feature engineering to enhance alarm classification in ICU settings.

The works that follow the (ii) prediction line mainly serve the industry and network systems. Zhu et al. (2016) addresses a dynamic alarm prediction algorithm for industrial plants, focusing on predicting critical alarms. The probabilistic model utilizes real-time alarm data, employing the n-gram model to calculate the occurrence probability of critical alarms, exemplified by the COT.LL alarm. The algorithm achieves accurate predictions, with a 0% missed prediction rate and a 14.3% false prediction

rate using a threshold of 0.55. The model provides operators with approximately 3 minutes of advance notice, allowing timely preventive actions. While the n-gram model proves effective, it has limitations in capturing long-distance correlations. The proposed probabilistic model demonstrates efficiency, accuracy, and real-time applicability for dynamic alarm prediction in industrial processes.

Quinn (2020) focuses on alarm forecasting methods for natural gas production pipelines, aiming to facilitate the transition from reactive to predictive maintenance. Four real-time alarm prediction methods are explored to detect system degradation onset and ensure flow assurance in the pipeline. The developed forecasting models serve as powerful tools for natural gas production companies, enabling predictive maintenance and avoiding shut-ins. The artificial neural network, in particular, outperforms the forecaster used by the production company, successfully predicting 82.7% of alarms 30 minutes in advance, contributing to significant resource savings for production companies.

Pezze et al. (2022) introduce FORMULA, a novel deep learning-based approach for Alarm Forecasting framed as a multi-label classification task, offering a low-cost alternative or supportive tool for sensor-based Predictive Maintenance. Leveraging Transformer architecture and addressing alarm imbalance with Weighted Focal Loss, FORMULA outperforms classic multilabel techniques and recurrent neural networks in predicting future alarms. The proposed approach not only demonstrates superior performance on a real-world industrial dataset from the packaging industry but also exhibits a lower computational burden, making it a valuable tool for timely corrective actions in industrial scenarios.

Zhang e Wang (2019) address the challenges posed by the increasing number of alarms in Optical Transport Networks (OTNs) and the critical need for failure prediction and analysis. The proposed scheme combines machine learning (ML) and deep learning (DL) algorithms to predict device failures using support vector machine (SVM) or long short-term memory (LSTM) and to analyze alarms through Time series segmentation, Time sliding window (TT), K-means, Back propagation neural network (KB) and modified Apriori with weights (W-Apriori). The evaluation, based on actual data from provincial telecommunications operators, demonstrates the effectiveness of the developed methods.

Zhuang et al. (2020) focus on leveraging machine learning (ML) for alarm prediction in large-scale optical transport networks, addressing challenges related to the quality of collected data. The proposed G-MAP algorithm, coupled with a self-optimizing data augmentation method based on generative adversarial networks (GANs), demonstrates significant improvements in prediction accuracy compared to existing methods. The experimental results on a commercial backbone synchronous

digital hierarchy (SDH) network underscore the effectiveness of the G-MAP algorithm, showcasing a 3.5% higher accuracy rate than benchmark algorithms.

Last, Lateano et al. (2023) introduce a modular machine learning (ML) framework for predicting failure root causes in a nationwide microwave network, addressing Quality of Services (QoS) requirements. Two workflows, designed for short- and long-term predictions, leverage real equipment alarms to achieve over 95% accuracy within an hour. The system learns from human experience, forecasts alarms, detects future failures and identifies root causes. The proposed ML-based framework demonstrates high-performance metrics and negligible execution times, offering operational benefits for microwave network operators.

However, the works in both groups (i) and (ii) do not address scenarios where it is important to determine not only the occurrence of the alarm but also its geographic location. Unlike previous studies, these scenarios are extremely relevant to our problem, given that it is crucial in determining the impact of alarms on the routes practiced in car patrolling. For example, a car is moving according to the defined route to patrol the area, but an alarm is triggered during the displacement. Since alarms represent a possibility of security vulnerability, the patrol must respond to the call, deviating from the planned route. The impact of these trajectory alterations over the year can mean a considerable increase in the operation cost. Moreover, to the best of our knowledge, there is a lack of research exploring the explainability of models within this context. Thus, different from previous efforts, this work seeks to determine machine learning algorithms that can assist in the predictability of these alarms within clusters while investigating their explainability, which would be a significant contribution to the field of alarm prediction.

4 PROBLEM DEFINITION AND DATASET

This chapter introduces the problem of alarm detection considering geographical location, utilizing data from Coopservice's patrol activities in Italy. In Section 4.1, the problem is formally defined as the development of a model to detect alarms and assign likelihood scores to alarm occurrences within specific clusters and time intervals. In Section 4.2, we introduced the Coopservice dataset with a focus on the studied province and key alarm-related information. The distribution of alarms is analyzed based on variations in month and hour. Section 4.3 outlines the process of clustering alarms from the perspective of Coopservice, as well as using the K-Means++ algorithm for theoretical comparison. The creation of negative instances (non-occurrence of alarms) and the generation of features based on date and time information are discussed. Additionally, the encoding method for each feature is described. These sections provide a comprehensive overview of the problem definition, dataset, and preprocessing steps, setting the foundation for subsequent analysis and model development.

4.1 Definition

Formally, we can define the problem of alarm detection considering geographical location as follows:

Definition 4.1. **(Alarms Detection)** Given a cluster $c \in C$, and a time interval $t \in T$, a model for alarm detection assigns a score $S(c,t) \in [0,1]$ indicating the extent to which the pair c,t is believed to have an alarm. A threshold τ can be defined such that the prediction function $F : C,T \rightarrow \{\text{alarm, not alarm}\}$ is:

$$F(c,t) = \begin{cases} \text{alarm}, & \text{if } S(c,t) > \tau \\ \text{not alarm}, & \text{otherwise.} \end{cases}$$

4.2 Overview of Coopservice's Dataset

The car patrol activities dataset, collected by Coopservice during their daily service operations in Italy, represents a rich source of information. It comprises data

from over 150 patrols, operating in more than 30 cities, and includes a total of over 150,000 alarms recorded between January 1, 2020, and November 6, 2022. The dataset encompasses key fields such as the date, geographic location (latitude and longitude), type of activity, and the corresponding car patrol. These fields collectively offer a detailed and comprehensive perspective on the car patrol activities conducted by Coopservice.

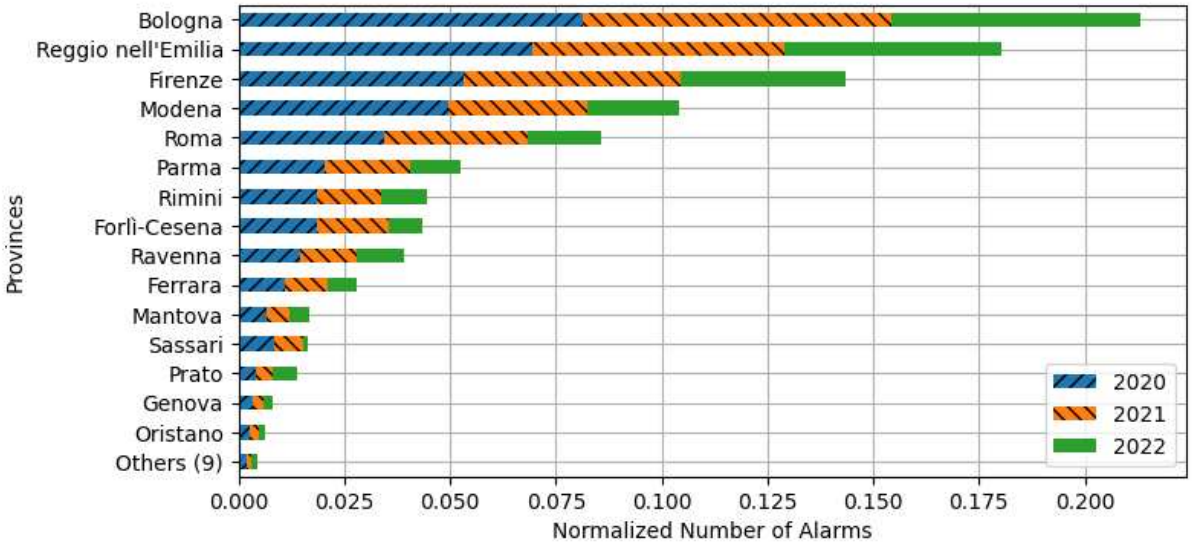


Figure 2 – Number of alarms by province. The graph show the number of alarms per province using the min-max normalization technique. This process ensures that the values are scaled between 0 and 1.

In this study, our analysis will be limited to the province level as patrols are assigned at this level. We chose the province of Reggio Emilia for our investigation due to its significant alarm activity, which is the second highest, as shown in Figure 2, and its strategic importance to the company. Being the location of the company's headquarters and having one of the highest occurrences of alarms among the areas covered by the patrol service, Reggio Emilia provides a suitable context for our study. In addition to Reggio Emilia, other provinces such as Bologna, Firenze, Modena, and Roma are used to assess how the best model trained in Reggio Emilia performs when applied to other regions.

The distribution of alarms across Reggio Emilia over the years is illustrated in Figure 3a, where a consistent pattern is observed except for August 2022. For November 2022, information is incomplete, covering only the first 6 days, and there is no data available for December 2022, which explains the absence of bars in the graph. According to the results presented in Figure 3a graph, between April and October, there is an increase in alarm occurrences as the maximum temperature of each month rises. Similarly, we observe a decrease in alarm occurrences during this same period as the temperature drops. One possible reason is that in colder seasons, people tend to spend more time at home, reducing the circulation of people and consequently the incidence

of alarms. In warmer seasons, we observe the opposite, with an increase in people's mobility. Additionally, during the hotter periods, we have the Italian summer holidays, which involve a significant movement of people between cities, particularly from the interior to the coastal areas.

Furthermore, Figure 3b shows that the incidence of alarms is higher during the night and early morning, peaking at 23 hours. However, it is crucial to take into account that the data was collected during the COVID-19 pandemic, and the protective measures adopted during this period had a significant impact on this distribution. The pandemic has undoubtedly affected the number of alarms.

4.3 Dataset Preprocessing

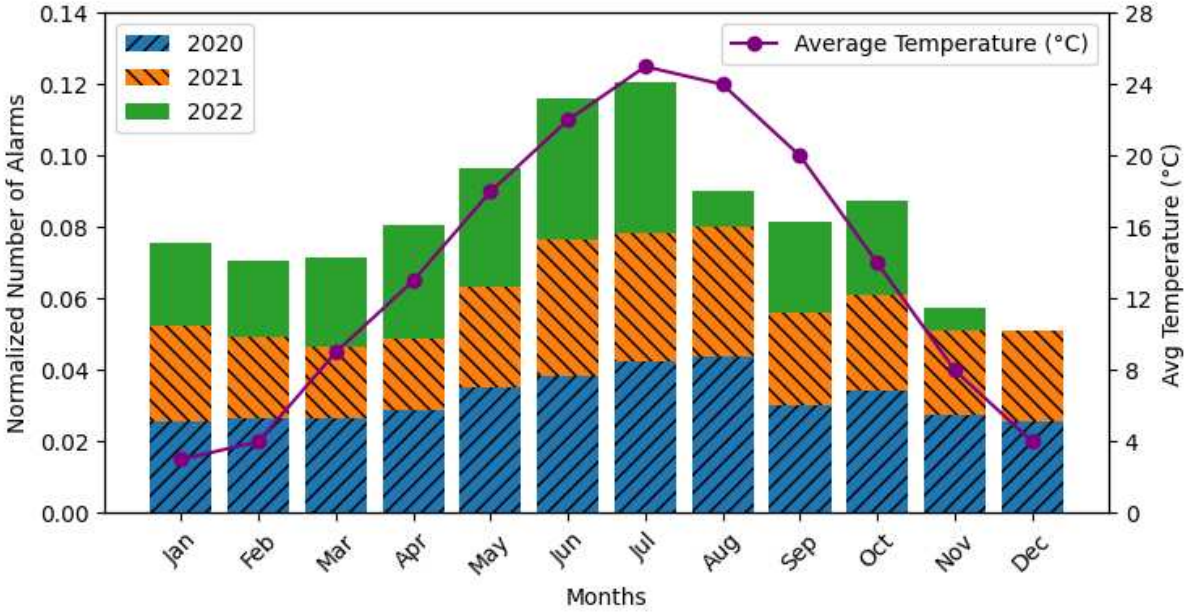
To make our original dataset suitable for making predictions, we perform a series of actions to prepare it in the ideal format for use in the selected machine learning approaches. Among the activities carried out, we work on an essential data cleaning step to remove or fill missing values and remove duplicates. Additionally, more steps are performed, which are further detailed in the next sections of this chapter:

- Dataset Clustering (Section 4.3.1);
- Non-alarm Instances (Section 4.3.2);
- Feature Definition and Encoding (Section 4.3.3);
- Data Processing Pipeline (Section 4.3.4).

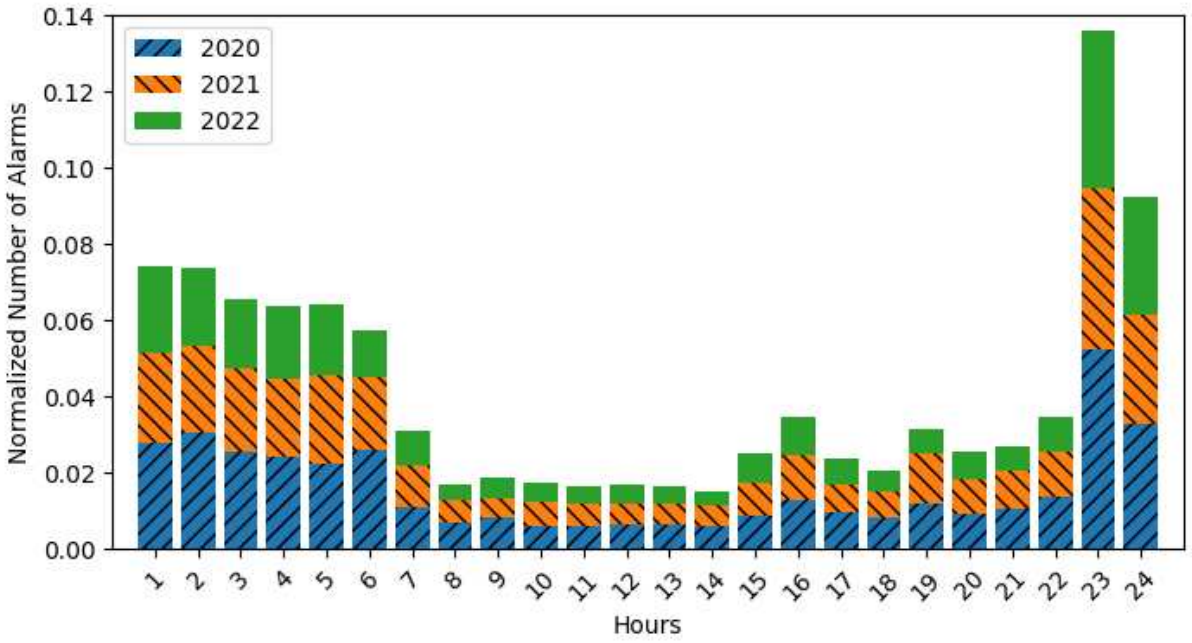
4.3.1 Dataset Clustering

According to Definition 4.1, presented at the beginning of this chapter, our problem involves classifying a specific event within a time interval and a cluster into either an alarm or non-alarm. Therefore, information about the alarm clusters is crucial for our work. As aforementioned, the alarm predictions are used in conjunction with Coopservice's route planning system to determine the optimal route for the daily activities of the patrols and to respond to any potential alarms. However, the current alarm response process at Coopservice does not consider this approach, and consequently, neither does our data. The alarm response is based on the patrol closest to the event at that moment, disregarding the predefined cluster for that patrol.

Through our dataset, we have information about the cluster values for the patrols during non-alarm activities. By considering the coordinates associated with each cluster, we define the convex region that represents it using the convex hull algorithm. In the



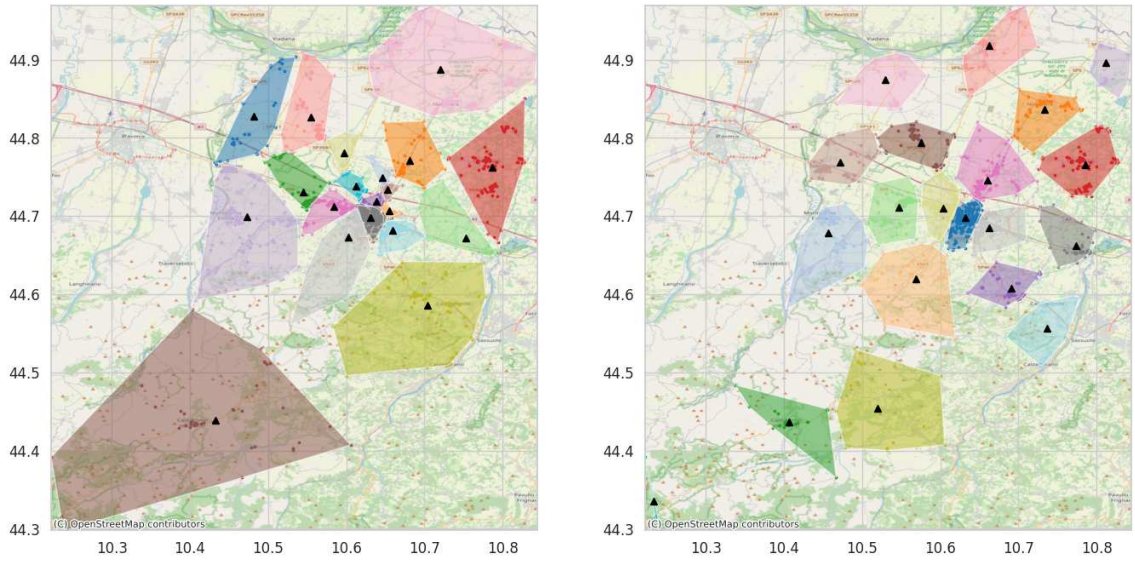
(a) Number of alarms and max temperature by months in Reggio Emilia.



(b) Number of alarms by hours.

Figure 3 – The graphs show the number of alarms per hour and month, with the y-axis values normalized using the min-max normalization technique. This process ensures that the values are scaled between 0 and 1.

field of computational geometry, the convex hull of a set of points in a plane is defined as the smallest possible convex polygon that encompasses all the given points. It can be visualized as the “outer boundary” of these points, forming a closed shape with no indentations or concavities. Using this convex region, we define that the alarm belongs to the convex region with the centroid closest to the alarm. This strategy allows us to work with the clusters already practiced by Coopservice.



(a) Convex regions obtained through Coopservice patrols.

(b) Convex regions obtained through K-Means++.

Figure 4 – Convex regions of clusters formed through geographical locations where alarms occur. The centroid of the points in each convex region is represented by a black triangle.

Coopservice operates with 20 patrols across Reggio Emilia province. The visual representation in Figure 4a showcases the distinct convex regions, each symbolizing a dedicated patrol unit. In our problem, each region represents a cluster for alarm response by car patrols. It is possible to observe that there is a significant variation in the size of the clusters and the number of locations each cluster encompasses. The definition and distribution of these clusters are solely based on what Coopservice already practices in handling other patrol activities. Consequently, there might be overloaded patrols in cases of a high number of locations to be covered or a long distance to travel. Conversely, some patrols might be less occupied due to an inadequate number of locations or a small patrolling area.

To provide a basis for comparison, we employ another clustering approach. For this purpose, we use the K-Means++ algorithm, detailed in Section 2.1.1, a popular choice that often yields good results for clustering problems involving geographic coordinates. While conventional studies determine the ideal number of clusters using methods like the elbow and silhouette, we opt for a value of k equal to 20 for comparison purposes, mirroring the number of clusters (patrols) used by Coopservice in the province of Reggio Emilia. The results obtained from K-Means++ are observable in Figure 4b.

The convex regions resulting from the two methods differ significantly in terms of area covered, number of locations, and alarm distribution, as shown in Figure 5. It is observed that K-Means++ obtained clusters with smaller areas to be covered for each patrol, which may lead to a shorter travel time. However, when evaluating the distribution

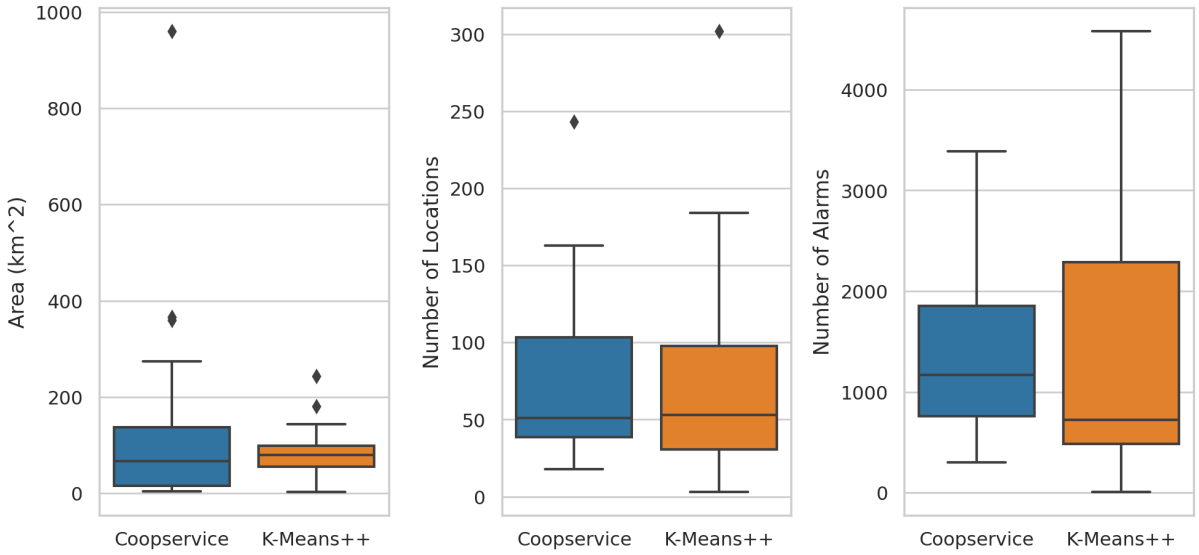


Figure 5 – Boxplot comparing covered area, number of locations, and alarm distribution of convex regions generated by Coopservice and K-Means++ methods.

of locations to be addressed and the number of alarms, we see that K-Means++ have extremes at the lower and upper ends. This can result in clusters with high alarm and location activity intensity and others much more idle compared to what is observed in the distribution already established by Coopservice.

4.3.2 Non-alarm Instances

In the previous chapter, we described how we defined the clusters for our work. Thus, we have two datasets: one clustering based on Coopservice's daily practices (COOP) and another clustering based on K-Means++ with k equal to 20 (KPP). Both datasets have only two columns at this stage: datetime (the moment the alarm occurred in the format yyyy-MM-dd hh-mm-ss) and cluster_code. Our prediction function, as defined in Definition 4.1, returns two possible outcomes: alarm and not alarm, depending on a defined threshold.

However, our dataset contains only occurrences of alarms, consisting of positive instances and no negative instances. To address this issue, we introduce negative occurrences into our dataset. The procedure consists of three steps and is detailed in the next paragraphs.

Figure 6 shows a scheme with the steps we perform to generate our dataset with positive and negative occurrences, called BD (binary dataset) in the picture. To focus on the one-hour time interval, our first step is to remove the minutes and seconds from the datetime. We do this by rounding the datetime by hour. We refer to this new dataset as RD (rounded dataset).

The second step is to take the first and last day of the dataset and create an hourly datetime entry for each cluster through the RD. We call this new dataset the CD (combinations dataset). For example, if we consider a 10-day interval with five clusters, this combination gives us a total of 1200 instances (10 days x 24 hours x 5 clusters). Finally, we join the RD and CD datasets, retaining all the entries from the CD. We assign the label alarm to the entries that are in both datasets and the label no alarm to the ones that are only in the CD. This is because the RD only has the alarm occurrences, so any entry that is missing from it is a non-occurrence. We obtain the BD dataset with two classes: alarm and no alarm, containing 475,152 non-alarm instances, following the same steps for COOP and KPP.

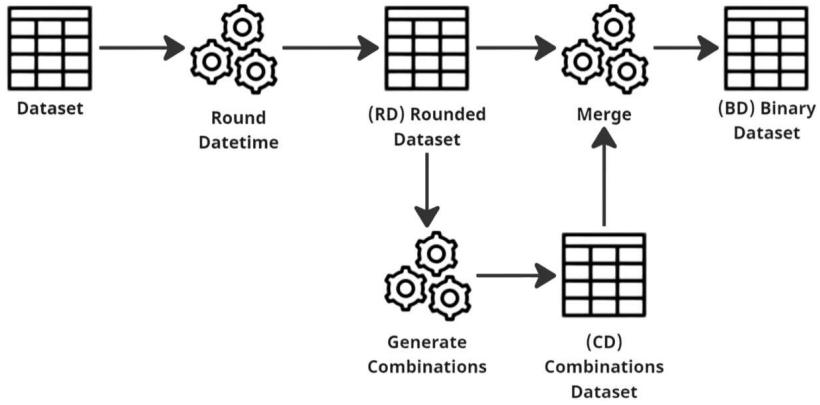


Figure 6 – Steps to generate the binary dataset BD that includes both classes: alarm and no alarm.

4.3.3 Feature Definition and Encoding

The last step in our data processing involves feature definition and encoding. As mentioned earlier, our datasets consist of two columns: datetime and cluster_code. Through the datetime column, we generate temporal features: year, month, day, hour, shift, and day_of_week. These features will help us better understand which individual time characteristic is more relevant for our model or if the model evaluates them together. The cluster_code has values based on the province code, for instance, RE for Reggio Emilia, followed by a number. If it ends with K, it is a KPP cluster; if it ends without K, it is a COOP cluster. Table 2 shows the features of our models, the type of each data, the encoding used, as well as the value ranges utilized. Notably, for cluster_code, we use two types of encoding to accommodate models across different provinces. This ensures compatibility and allows us to handle variations in the number of clusters per province.

Table 2 – Properties and values of the features for the datasets.

Feature	Type	Encoding	Values
year	Discrete	Label/Ordinal Encoding	2020 to 2022
month	Discrete	Label/Ordinal Encoding	1 to 12
day	Discrete	Label/Ordinal Encoding	1 to 31
hour	Discrete	Label/Ordinal Encoding	1 to 24
shift	Ordinal	Label/Ordinal Encoding	dawn, morning, afternoon and night
day_of_week	Ordinal	Label/Ordinal Encoding	1 to 7
cluster_code	Nominal	One-hot Encoding Label/Ordinal Encoding	RE00 to RE19 or RE00K to RE19K

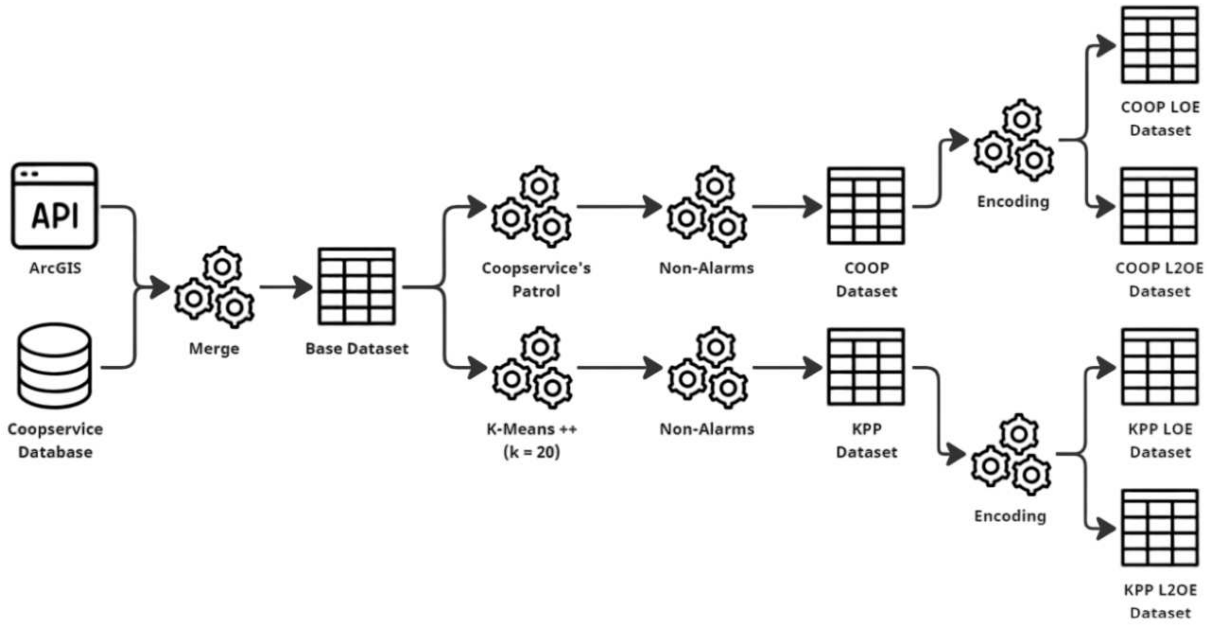


Figure 7 – Illustration of the data processing pipeline.

4.3.4 Data Processing Pipeline

We generate four distinct datasets, incorporating two clustering models (COOP and KPP), and employing two encoding methods for the cluster_code: Label/Ordinal Encoding (LOE) and Label/Ordinal/One-hot Encoding (L2OE). Figure 7 illustrates the sequential steps involved in producing the final datasets through our data processing pipeline. Initially, we access the Coopservice database to extract patrol data. Concurrently, using the ArcGIS API[1], we collect address information to transform the latitude and longitude data into city, province, and other relevant details. These combined pieces of information form the Base Dataset. Applying Coopservice's clustering methods and K-Means++, and integrating a non-alarm creation function, we create the COOP and KPP datasets. Following this, through the application of encoding techniques, we produce the four final datasets that serve as the training data for our models.

[1] https://developers.arcgis.com/python/

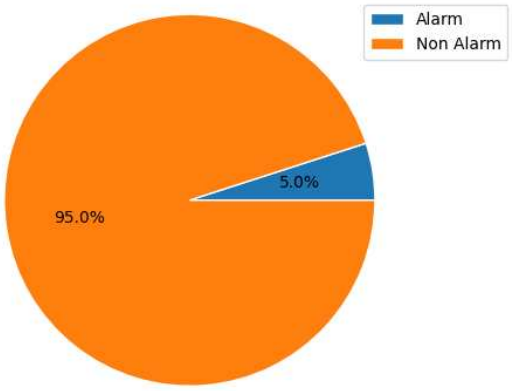


Figure 8 – Percentage distribution of alarms and non-alarms in the datasets.

The datasets consist of 499,680 rows, encompassing instances of both alarm occurrences and non-occurrences. The LOE datasets contain 7 features, while the L2OE datasets boast 26 features. The disparity arises from the utilization of One-Hot Encoding, which transforms our cluster_code feature into 20 features, each corresponding to a distinct cluster. Lastly, Figure 8 illustrates the datasets' imbalance, revealing only 5% instances of alarm occurrences, representing a ratio of 1 occurrence for every 19 non-occurrences. Note that the alarm occurrences described in this dataset do not represent the total number of alarms attended by Coopservice. Since a cluster comprises a set of locations, an alarm in any of the locations within a specific cluster was defined as an instance with an alarm in our problem. Thus, a cluster can have, for example, 10 alarms and another 5 in a time interval, and this does not affect our model, as we consider an alarm occurrence for a cluster if the number of alarms is greater than 0; otherwise, we consider it a non-occurrence.

5 MODELING AND ANALYSIS

This chapter explores the process of model selection, evaluation, and refinement. Three machine learning models, namely Naive Bayes (NB), eXtreme Gradient Boosting (XGB), and Multilayer Perceptron Classifier (MLP), are being scrutinized concerning their efficiency, with a focus on alarm clustering methods: Coopservice-2022 (COOP) and K-Means++ (KPP). Thorough model assessments, predominantly utilizing the AUC metric, allow a deeper understanding of not just accuracy but also the models' sensitivity in predicting alarm occurrences.

Furthermore, hyperparameter optimization, particularly for the XGB algorithm, is being explored to enhance model performance. The chapter also investigates advanced interpretative techniques such as the SHAP framework, offering insights into feature importance and the influence of various factors on alarm classification. In subsequent sections, we detail model selection and evaluation in Section 5.1, hyperparameter optimization in Section 5.2, and the analysis of the best models in Section 5.3.

5.1 Model Selection

The dataset used comprises 1041 days of data from January 1, 2020, to November 6, 2022. The models are currently being trained using 720 days of the dataset, representing 69.09% of the total. The data is separated to consider the temporal nature of the information, ensuring that the test data consists entirely of dates after the training set; in other words, no future data is used to predict the past. The remaining portion is solely used for model evaluation and plays no role in training the models under study. This approach assists in bias reduction. Models had been trained for both types of clustering applied: Coopservice-2022 (COOP) and K-Means++ (KPP).

For this study, we select three machine learning approaches: Naive Bayes (NB), eXtreme Gradient Boosting (XGB), and Multilayer Perceptron Classifier (MLP), mainly for their rapid training speed and the robustness of results obtained from preliminary training. Additionally, these algorithms represent different families of machine learning. NB is a probabilistic algorithm, XGB is an ensemble algorithm, and MLP belongs to the family of artificial neural networks. All training is conducted on a platform similar to the one used by (BRAVO et al., 2022) in their work, namely, the Google Colab

platform (GOOGLE, 2019), using the Python 3 programming language. The NB and MLP algorithms are imported from the Scikit-Learn library (PEDREGOSA et al., 2011), while XGB is imported from the xgboost library (CHO, 2023).

To ensure a fair comparison, we examine them using their default configurations provided by the libraries used. Additionally, they are compared and trained under the same conditions, utilizing the same training and test sets. In this evaluation, we employ the area under the ROC curve (AUC) metric, widely acknowledged in the literature as suitable for training models on imbalanced datasets, as emphasized by (BHOWAN; JOHNSTON; ZHANG, 2011). The AUC metric captures the overall performance of the models by taking into account the trade-off between the true positive rate and the false positive rate. It is important to mention that metrics such as accuracy are not good in this context because they are biased toward the majority class. Additionally, we can determine the ROC curve and vary the probability threshold to determine the label that each probability will be assigned to, allowing stakeholders to determine the best threshold in the practical application of the model in the company.

Table 3 provides an overview of the training process and model configuration. It is important to emphasize that at this stage, we are not considering hyperparameter selection for the trained models; we are using the default values defined by the Python libraries we have employed.

Table 3 – Summary of training data and model configuration.

Properties	**Values**
Provincy	Reggio nell'Emilia (RE)
Number of days	1041
Number of clusters	20
Time granularity	hourly
Dataset size	499,680
Training percentage	69.09% (720 days)
Algorithms	- Naive Bayes (NB) - eXtreme Gradient Boosting (XGB) - Multilayer Perceptron Classifier (MLPC)
Encoding strategies	- Label/Ordinal Encoding (LOE) - Label/Ordinal/One-hot Encoding (L2OE)
Clustering methods	- Coopservice-2022 (COOP) - K-Means++ (KPP)
Number of models	12 = (3 algorithms) x (2 encodings) x (2 clusterings)

The models are named according to the criteria: {ALGORITHM}_{PROVINCE}_{CLUSTERING}_{ENCODING}. For instance, NB_RE_COOP_LOE represents a model that employs the Naive Bayes algorithm, trained with data from the province of Reggio Emilia, using the Coopservice alarm clustering method, and following the encoding strategy LOE, which involves Label/Ordinal Encoding exclusively.

After training the 12 models, we initially evaluate using the metrics explained in Section 2.3.2. Table 4 presents the values obtained considering a threshold of 0.5, meaning that if there is a chance higher than 0.5 of an alarm, we classify it as an actual alarm; for lower values, we classify it as no alarm. Rows with 1 represent the classification of alarm occurrences, and 0 denotes non-occurrence.

Table 4 – Results obtained for multiple models: comparison of precision, recall, and F1-score.

	nb_re_coop_loe (MF1-score: 0.490)		
	Precision	**Recall**	**F1-score**
0	0.95	1.00	0.98
1	0.00	0.00	0.00

	nb_re_kpp_loe (MF1-score: 0.490)		
	Precision	**Recall**	**F1-score**
0	0.95	1.00	0.98
1	0.00	0.00	0.00

	nb_re_coop_l2oe (MF1-score: 0.475)		
	Precision	**Recall**	**F1-score**
0	0.97	0.69	0.80
1	0.09	0.60	0.15

	nb_re_kpp_l2oe (MF1-score: 0.465)		
	Precision	**Recall**	**F1-score**
0	0.98	0.62	0.76
1	0.09	0.79	0.17

	xgb_re_coop_loe (MF1-score: 0.500)		
	Precision	**Recall**	**F1-score**
0	0.95	1.00	0.98
1	0.28	0.01	0.02

	xgb_re_kpp_loe (MF1-score: 0.510)		
	Precision	**Recall**	**F1-score**
0	0.95	1.00	0.98
1	0.37	0.02	0.04

	xgb_re_coop_l2oe (MF1-score: 0.500)		
	Precision	**Recall**	**F1-score**
0	0.95	1.00	0.98
1	0.31	0.01	0.02

	xgb_re_kpp_l2oe (MF1-score: 0.510)		
	Precision	**Recall**	**F1-score**
0	0.95	1.00	0.98
1	0.37	0.02	0.04

	mlp_re_coop_loe (MF1-score: 0.490)		
	Precision	**Recall**	**F1-score**
0	0.95	1.00	0.98
1	0.00	0.00	0.00

	mlp_re_kpp_loe (MF1-score: 0.490)		
	Precision	**Recall**	**F1-score**
0	0.95	1.00	0.98
1	0.00	0.00	0.00

	mlp_re_coop_lo2e (MF1-score: 0.490)		
	Precision	**Recall**	**F1-score**
0	0.95	1.00	0.98
1	0.00	0.00	0.00

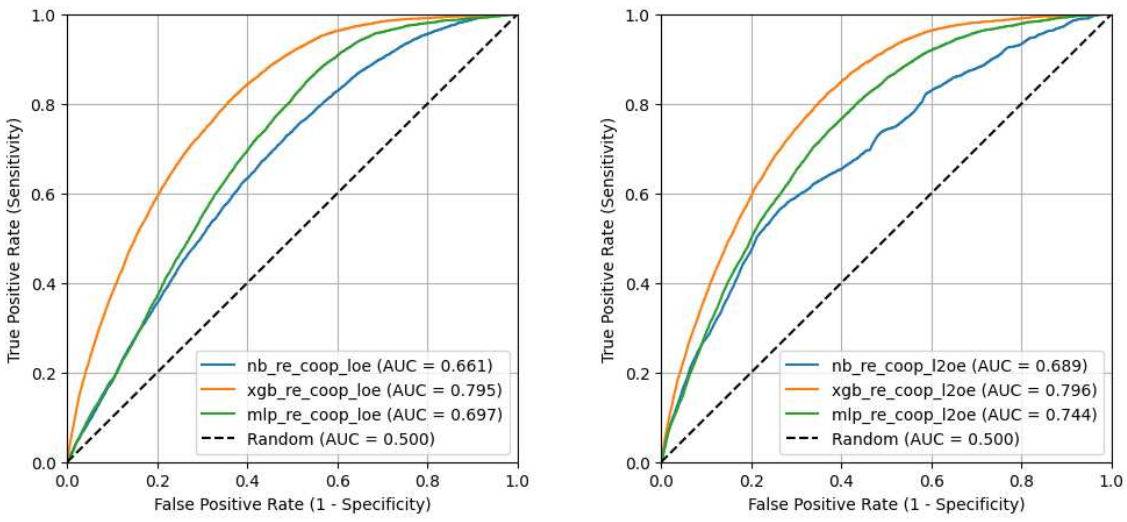
	mlp_re_kpp_l2oe (MF1-score: 0.490)		
	Precision	**Recall**	**F1-score**
0	0.95	1.00	0.98
1	0.00	0.00	0.00

Evaluating by the MF1-score metric (Macro Average F1-score), we observe a variation of 0.045 between the worst model, nb_re_kpp_l2oe, with an MF1-score of 0.465, and the best models, xgb_re_kpp_l2oe and xgb_re_kpp_l2oe, with an MF1-score of 0.510.

Some models, such as MLP and NB, focus solely on the absence of alarms and completely disregard alarm occurrences. Although these models have a performance similar to the best models when it comes to evaluating the F1-score of non-alarm (0.98), they do not effectively address the problem we aim to solve. Alarm occurrences happen much less frequently than their absence, with a ratio of 1/20 for an occurrence to the

absence of alarms. As a result, they have the worst performances regarding the F1-score of alarm occurrences. The best model, when evaluated based on the F1-score of alarm occurrences, is nb_re_kpp_l2oe, which has an F1-score of 0.17, despite having the lowest MF1-score. While these metrics are a good starting point, it is essential to investigate further because these values are calculated for a threshold of 0.5, and it is possible to alter the threshold to seek better results.

To comprehend the impact of threshold variations on our models, we employ ROC curves and AUC values. Figure 9 illustrates the AUC values for models trained using the Coopservice alarm distribution. From the AUC values shown in each graph, it is evident that XGB takes the lead. Figure 9a shows that for the LOE encoding, XGB is above all other models at almost all points. In the range from 0 to 0.2, it is observed that the MLP and NB models are very close, making it challenging to distinguish which is superior. However, from 0.2 onwards, the MLP outperforms the NB. Moving to Figure 9b for L2OE encoding models, again, XGB is dominant across almost the entire range, but the MLP is much closer to it than in the previous encoding. The NB and MLP are close in the 0 to 0.2 range, but afterward, the MLP maintains an advantage. When comparing the two XGB models (LOE and L2OE), we observe a near tie, with just a 0.001 difference in AUC between the models.



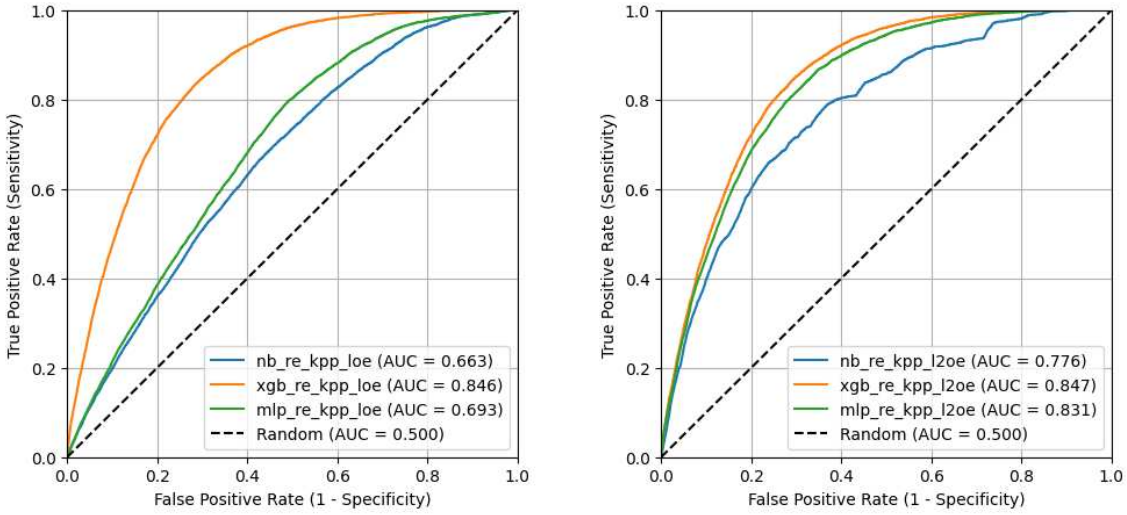
(a) LOE models.

(b) L2OE models.

Figure 9 – ROC curves and AUC for the COOP models compared to a random model as reference.

Similarly, one observes in Figure 10 that the XGB models excel in predicting alarm distribution using K-Means++. When employing LOE encoding, a significant disparity emerges between XGB and the other models, while this distinction becomes less pronounced among models using L2OE encoding. The graphs illustrate the XGB models' strong performance in predicting alarm distributions, particularly when leveraging the K-

Means++. This discrepancy in performance underscores the importance of selecting the appropriate clustering method, as it significantly influences model outcomes. Moreover, the choice of encoding strategy also plays a role in model performance, with LOE encoding showing distinct results compared to L2OE encoding.



(a) LOE models.

(b) L2OE models.

Figure 10 – ROC curves and AUC for the KPP models compared to a random model as reference.

In Table 5, the AUC values for the 12 trained models are presented. The XGB performs the best in all scenarios. When comparing XGB LOE and XGB L2OE for both COOP and KPP, there is no significant difference, indicating that for XGB, applying One-Hot encoding to the cluster_code feature does not result in an improvement in the model. However, for other algorithms, the difference is relevant, especially for the MLP KPP, leaping from 0.693 in LOE to 0.831 in L2OE. Thus, for XGB-based models, the encoding performance does not appear to be a determining factor in model choice.

Assessing the clustering methods for the XGB models reveals a jump of 0.051 between COOP and KPP, regardless of the encoding. This leap is even more significant among the MLP L2OE models, shifting from 0.744 in COOP to 0.831 in KPP, a difference of 0.087. Hence, it is evident that clustering significantly impacted our best models. Therefore, based on the available information, the best models are obtained through XGB.

In this way, models with better AUC provide us with greater flexibility to vary the threshold and achieve good prediction results. Since the goal is to use the prediction model in an optimization engine that determines the best patrolling route, the ideal threshold value is chosen based on Coopservice's specific needs. We can consider two possible scenarios:

Table 5 – Comparison of AUC scores for models trained with COOP and KPP clustering.

COOP		KPP	
Model	**AUC**	**Model**	**AUC**
nb_re_coop_loe	0.661	nb_re_kpp_loe	0.663
xgb_re_coop_loe	0.795	xgb_re_kpp_loe	0.846
mlp_re_coop_loe	0.697	mlp_re_kpp_loe	0.693
nb_re_coop_l2oe	0.689	nb_re_kpp_l2oe	0.776
xgb_re_coop_l2oe	**0.796**	xgb_re_kpp_l2oe	**0.847**
mlp_re_coop_l2oe	0.744	mlp_re_kpp_l2oe	0.831

1. **Maximizing Precision**: This strategy aims to maximize the number of alarm predictions that actually turn out to be alarms, emphasizing the precision metric. It helps reduce costs by avoiding unnecessary deployments to regions prone to false positives. However, it may lead to non-responses to some actual alarm occurrences, as the model tends to be more conservative in classifying alarm occurrences.

2. **Maximizing Recall**: This strategy focuses on ensuring that the maximum number of alarm occurrences are predicted as alarm occurrences, emphasizing the recall metric. This ensures that most alarms are responded to, even at the cost of deploying to regions where no alarm occurrence has actually occurred.

Based on earlier discussions, we choose to proceed with the XGB algorithm. We chose this algorithm due to its superior AUC score and faster training time than MLP, the second best. Additionally, the XGB model is known for its interpretability, making it easier to collaborate with Coopservice's patrol experts.

5.2 Hyperparameter Optimization

Following the selection of our algorithm, we proceed to explore ways to achieve higher AUC scores. To attain this objective, we conduct hyperparameter tuning to construct more robust models. As previously mentioned, our primary focus is on the XGB algorithm, as it was the approach with the best results when evaluating the AUC score, and we also assess the impact of two clustering methods and two encoding strategies.

In the last section, we divide our data into two sets: train and test. For hyper-parameter selection, we split the training data into folds, with each fold having its distribution for training, validation, and testing. The training, testing, and validation sets together encompass a maximum of 720 days from the dataset, as illustrated in the distribution shown in Figure 11. We use a 10-fold cross-validation approach where each fold represents 72 days, with 36 days for training, 18 days for validation, and 18 days

for testing. The ratio between training, validation, and testing is 2-1-1. The objective of this separation is to train the model on small samples of our dataset to ensure that the hyperparameters selected in the end are not strongly dependent on the data in our training set. This helps create a model that can generalize well to new information and perform satisfactorily with data that was not used in its training, avoiding problems with overfitting.

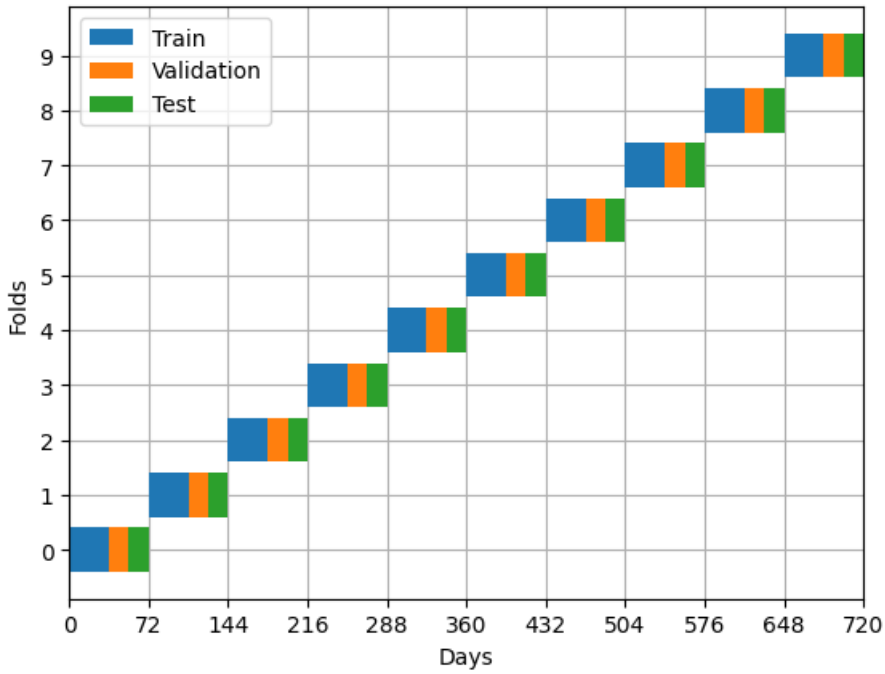


Figure 11 – The distribution of the 720 days of data used for hyperparameter selection.

The remaining 321 days are set aside for the final AUC score of the models. Considering the time series nature of the problem, we choose to perform isolated block-wise splits. This means that we train our models on one portion of the dataset and evaluate them on the chronologically subsequent portion, thus ensuring that we do not use future information to make predictions. In a real-world scenario, we have access only to historical data, not future information. This practice ensures a robust evaluation of our models.

Among the various hyperparameters considered, we utilize the following parameters for their common applicability in XGB projects and other tree-based models:

- **max_depth**: This hyperparameter controls the maximum depth of each decision tree in the XGB model. Deeper trees capture more complex data relationships but must be adequately limited to prevent over-fitting.
- **learning_rate**: The learning rate determines the size of steps the XGB model takes during the optimization process. A lower learning rate results in more precise

training but demands more iterations. Conversely, a higher learning rate speeds up training but might lead to sub-optimal solutions.

- **n_estimators**: This hyperparameter represents the number of decision trees created during the XGB model's training. A higher value increases model complexity and make it more robust. However, a huge number of trees can lead to over-fitting.
- **early_stopping_rounds**: It is an integer value that defines the number of iterations the algorithm will continue training after not observing an improvement in performance on a validation set. This technique is used during machine learning algorithm training, such as XGB, to prevent over-fitting and optimize the selection of the ideal number of iterations (the number of trees in the case of XGB). It has a direct relationship with n_estimators, often set to 10% of the n_estimators value.
- **eval_metric**: This function assesses the model's performance during training. In the context of XGB, it is usually a metric like AUC (Area Under the ROC Curve) for classification problems or RMSE (Root Mean Squared Error) for regression problems. This metric is crucial in determining the model's quality.

Table 6 outlines the hyperparameter search space and the pre-selected values for each hyperparameter. Each model is evaluated across 125 different configurations. In other words, 5 maximum depths (max_depth), 5 learning rates (learning_rate), and 5 pairs of n_estimators and early_stopping_rounds are explored, resulting in 125 possible combinations. These combinations were explored exhaustively through the grid search of the Scikit-learn library.

Throughout the 10-fold cross-validation process, the configuration yielding the best average AUC among the tests in the folds is considered the optimal set of hyperparameters. The selection of hyperparameters is based on the available information in the XGB documentation (DEVELOPERS, 2023) and the blog post, "Mastering XGBoost Parameter Tuning: A Complete Guide with Python Codes", from the Analytics Vidhya blog (JAIN, 2016). This blog post is recommended by Awesome XGB[1] (DMLC, 2023), which is a curated list of examples, tutorials, and blogs dedicated to XGB use cases.

Table 7 presents an overview of the hyperparameter values that have been chosen for the four trained models. These models feature distinct combinations of hyperparameters, thereby emphasizing the importance of tailoring each one to suit the specific characteristics of the dataset. For example, the xgb_re_coop_loe model boasts a max_depth of 3, a learning_rate of 0.20, a n_estimators setting of 400, and an early_stopping_rounds value of 40. It is notable that although there is some overlap in

[1] Awesome XGB is mentioned in the documentation as a source of resources on XGB, available at <https://xgboost.readthedocs.io/en/stable/tutorials/index.html>.

Table 6 – Hyperparameter search space.

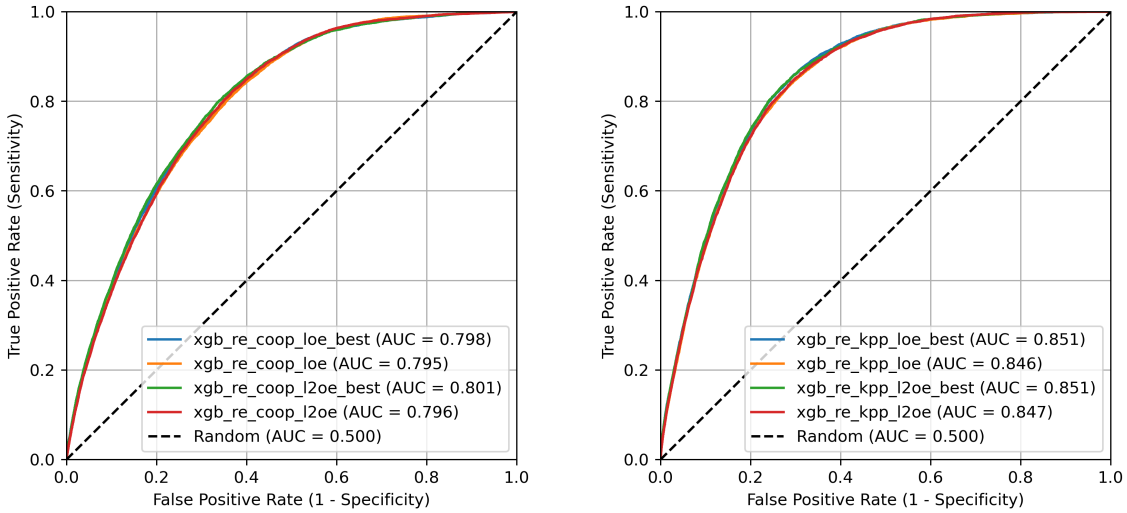
Hyperparameters	**Values**
max_depth	3, 4, 5, 6, 9
learning_rate	0.05, 0.10, 0.15, 0.20, 0.25
n_estimators	100, 200, 400, 800, 1600
early_stopping_rounds	10, 20, 40, 80, 160
eval_metric	auc

the values of individual hyperparameters across these models, the final hyperparameter combinations for each model are unique.

Table 7 – Best hyperparameter values for XGB trained models.

	Hyperparameters			
Models	**max_depth**	**learning_rate**	**n_estimators**	**early_stopping_rounds**
xgb_re_coop_loe	3	0.20	400	40
xgb_re_coop_l2oe	3	0.15	800	80
xgb_re_kpp_loe	5	0.10	400	40
xgb_re_kpp_l2oe	4	0.15	800	80

In Figure 12a, the behavior of models trained with the COOP clustering and tuned hyperparameters is compared to models with default hyperparameters. All models perform closely, with almost complete overlap in the graph's curves. Similarly, the encoding method does not appear to have a substantial impact on the results. Figure 12b reinforces the lack of influence of tuning on the KPP-clustered models and the encoding method.



(a) COOP-clustered models.

(b) KPP-clustered models.

Figure 12 – Comparison of ROC curves for XGB models: tuned vs. default hyperparameters.

According to Table 8, it becomes evident that the models do not exhibit significant improvements in their performance. In the best scenario, there is only a 0.6% increase

in AUC for the KPP models. In contrast, it is noteworthy that the choice of clustering method has a more substantial impact on the models. This results in an increase of approximately 5% in AUC for the KPP models compared to the COOP models.

Even though we may not find hyperparameter values that bring considerably better results, we cannot guarantee that these values do not exist since the search space we use can be expanded. We do not dedicate more time and effort to this stage because our final goal in this work is not to achieve the best possible model but to explore the possibilities that Coopservice can adopt to connect the patrol optimizer to predictive alarm models.

Table 8 – AUC comparison for XGB models trained with distinct clustering and encoding methods.

	AUC		
Models	**Default**	**Tuned**	**Improvement**
xgb_re_coop_loe	0.795	0.799	0.4%
xgb_re_coop_l2oe	0.796	0.801	0.5%
xgb_re_kpp_loe	0.846	0.852	0.6%
xgb_re_kpp_l2oe	0.847	0.853	0.6%

Once, for our XGB models, the addition of One-Hot Encoding does not bring any significant performance improvement, we proceed with LOE encoding exclusively. Among the reasons, it is a simpler model to evaluate since the number of features is smaller. It is possible to use XGB models without One-Hot Encoding in other datasets since the number of features remains the same. With One-Hot Encoding, we have one feature per cluster. For provinces with different patrol numbers, it would not be possible to apply the already-trained models.

In the context of data analysis, these findings suggest that our choice of LOE encoding provides a straightforward and efficient approach, reducing the complexity of the model and ensuring versatility across different datasets with consistent feature numbers. Additionally, it paves the way for potential applications of transfer learning in scenarios where provinces may have varying patrol numbers, preserving the models' adaptability and predictive capabilities.

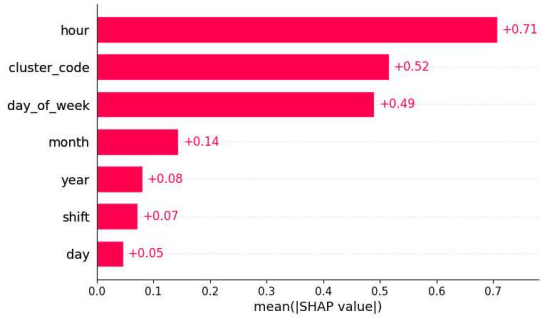
5.3 Analysis of Optimal Models

Interpreting the predictions of complex machine learning models becomes increasingly important in many applications. Understanding why a certain prediction is made becomes as crucial as the prediction's accuracy. However, the highest accuracy for modern datasets is often achieved by complex models challenging to interpret, creating a tension between accuracy and interpretability. To address this issue, various methods

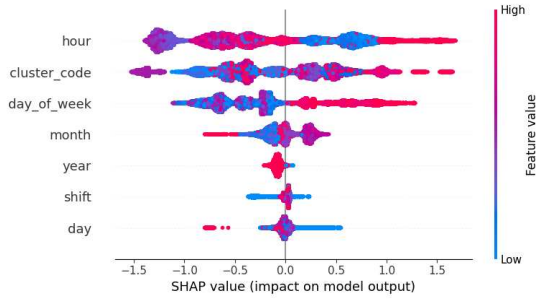
have been proposed recently to help interpret the predictions of complex models, including the SHapley Additive exPlanations (SHAP) framework by (LUNDBERG; LEE, 2017).

SHAP assigns an importance value to each feature for a specific prediction and unifies six existing methods into a new class of additive feature importance measures. The framework provides a unique solution with desirable properties and theoretical support. In addition to its widespread use for improving the interpretability of machine learning models, the SHAP framework is chosen in this work. It has been explored in previous efforts, such as (LI, 2022), (EKANAYAKE; MEDDAGE; RATHNAYAKE, 2022), and (MANGALATHU; HWANG; JEON, 2020), dealing with the black-box nature of machine learning models in different contexts.

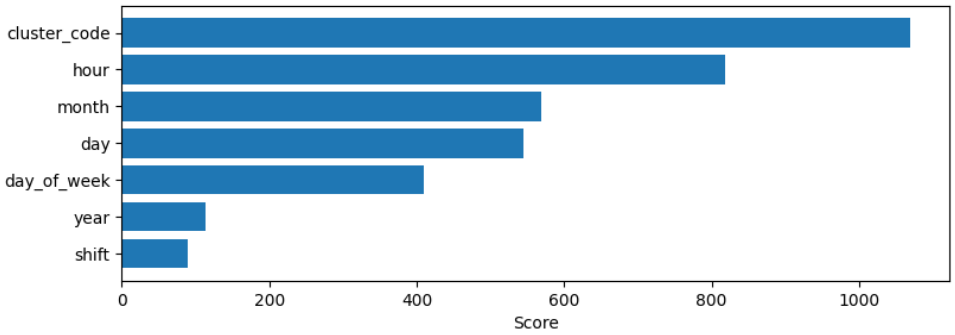
For the xgb_re_coop_loe model, the information is displayed in Figure 13. Figure 13a presents the ranking of the model's features based on the absolute value of the SHAP metric. These values indicate the impact these features have on the model. Since the values are in absolute terms, it is not possible to distinguish if they are for the classification of alarm occurrences or non-occurrences. From the values, we observe that hour, cluster_code, and day_of_week are the most crucial features for this model and possess significantly higher values compared to others.



(a) Feature ranking based on the absolute value of SHAP scores for COOP model.



(b) Summary of feature distribution versus SHAP values for the COOP model.

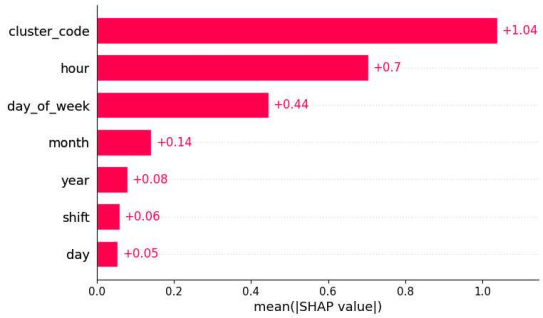


(c) Feature importance generated by XGB for the COOP model.

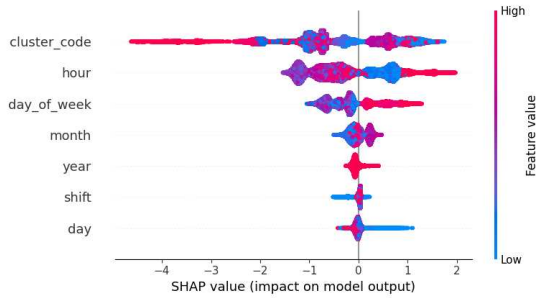
Figure 13 – Showcasing feature importance and distribution analysis using SHAP scores for the COOP model's alarm prediction analysis.

Figure 13b illustrates the distribution of feature values against SHAP values. This distribution identifies whether the feature values impact the classification of alarm occurrences or non-occurrences. Based on the distribution in Figure 3 and the SHAP values, it is observed that extreme hours (both low and high), affect alarm occurrences — periods with minimal pedestrian traffic, typically during sleeping hours. Additionally, weekends also positively impact the model.

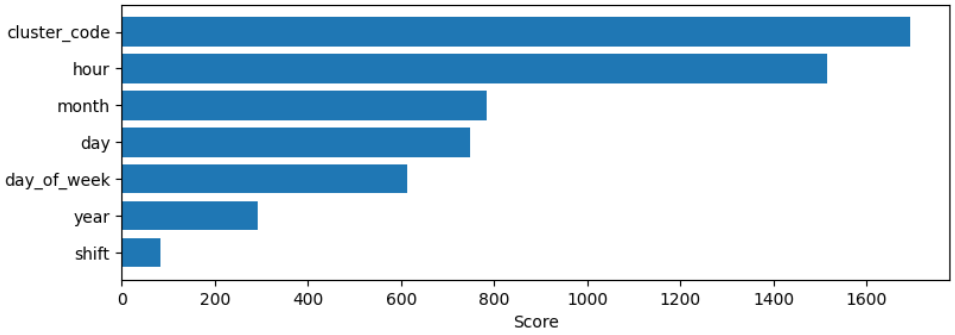
As aforementioned in Section 2.2.2, XGB has an internal feature importance mechanism for the generated models. Figure 13c presents the values obtained. The configuration obtained directly by XGB is quite different from that obtained by SHAP. All features appear in different positions, but it is observed in both that cluster_code and hour are the most important and contribute the most to the model's results in both methodologies.



(a) Feature ranking based on the absolute value of SHAP scores for KPP model.



(b) Summary of feature distribution versus SHAP values for the KPP model.



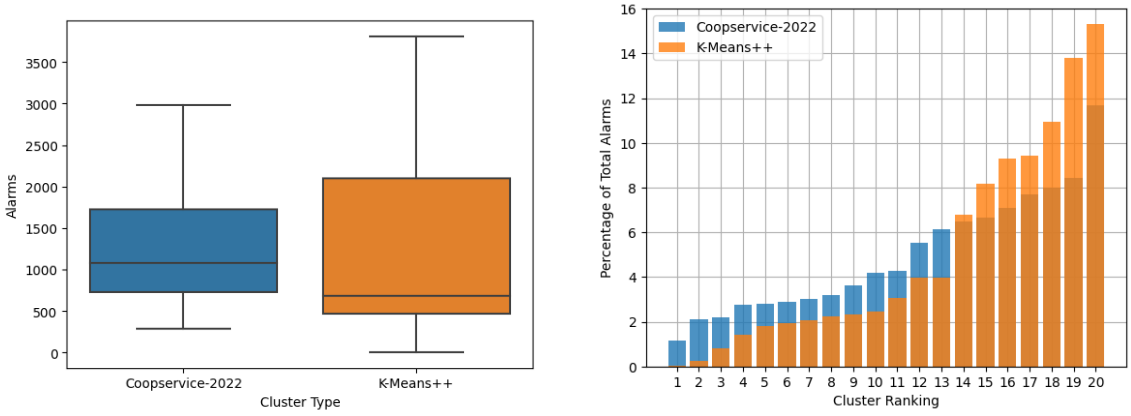
(c) Feature importance generated by XGB for the KPP model.

Figure 14 – Showcasing feature importance and distribution analysis using SHAP scores for the KPP model's alarm prediction analysis.

In Figure 14a, the same analysis is carried out for the xgb_re_kpp_loe model as was done for the previous model. It reveals that for KPP, the model prioritizes times with low pedestrian traffic and weekends. However, a significant difference emerges as the most important feature becomes the cluster_code, providing further evidence of how alarm distribution can enhance the model's performance. Notably, the contribution of each feature to the models shows very close values, with only one feature

—cluster_code— exhibiting a notably higher difference. This difference probably occurs because cluster_code is the only feature modeled differently between the two datasets.

Similarly to what occurred with COOP, Figure 14c shows that the order of importance of features is quite different when comparing the SHAP and XGB methodologies. However, this time both methodologies indicate that cluster_code is the most important feature, followed by hour. In a scenario with a high number of features, it would be interesting to compare the results of models with only the best features selected by each methodology.



(a) Boxplot displaying the distribution of alarms across different cluster types.

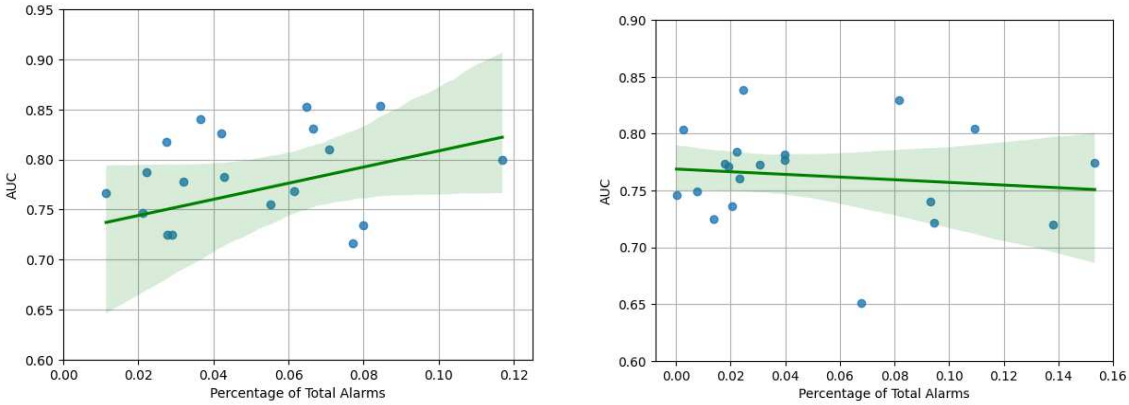
(b) Comparative graph ranking clusters based on alarm occurrences.

Figure 15 – Comparison of alarm distributions across cluster types.

Figure 15a shows a distinct difference that emerges in the alarm distribution across clusters. Coopservice-2022 showcases a more balanced distribution, significantly reducing the range of alarm numbers compared to K-Means++. The mean in COOP is higher than in KPP. Figure 15b shows that in 13 of 20 clusters, COOP has a higher number of alarms than KPP after ordering them by alarm numbers. Additionally, some KPP clusters exhibit a notably low number of alarms.

Upon analyzing the correlation between the model's AUC for each cluster and the percentage of alarms within clusters, as shown in Figure 16, we observe some intriguing relationships. Beginning with COOP, Figure 16a presents a positive correlation of 0.252. Here, clusters with a higher number of alarms are more readily assessed by the XGB model. In contrast, for KPP clustering, as highlighted by Figure 16b, there is a slight negative correlation of -0.128, where higher alarm numbers indicate a poorer model performance according to the AUC.

These observations shed light on the varying impacts of alarm distribution within clusters on model performance. In the case of COOP, higher alarm occurrences appear to enhance the model's assessment, while for KPP, more alarms seem to challenge the model's evaluative capability. Such contrasting correlations highlight the model's



(a) Correlation between AUC and alarm percentage for COOP clusters.

(b) Correlation between AUC and alarm percentage for KPP clusters.

Figure 16 – Comparative analysis of AUC-alarm percentage correlations for COOP and KPP clusters. These contrasting correlations highlight the model's sensitivity to distinct alarm patterns within clusters, influencing their performance differently based on the clustering method.

sensitivity to cluster-specific alarm patterns and their subsequent influence on AUC performance.

Perhaps we could explore an alternative approach to enhance collaboration between the route optimizer and the alarm predictor. One idea could be to assign an AUC score to each cluster and specific time slots. This approach might improve work quality by allowing the optimizer to prioritize alarm predictions for clusters and time slots with higher scores, while de-emphasizing those with lower accuracy.

Implementing such a strategy could optimize the allocation of resources and enhance the overall efficiency of the alarm response system. By directing patrol teams to areas where alarm predictions are more reliable, we could reduce unnecessary deployments to regions with lower prediction accuracy. Incorporating AUC scores into the decision-making process could help fine-tune the system to the unique characteristics of each cluster and the variations in alarm occurrence patterns throughout the day.

6 TRANSFER LEARNING

The province of Reggio Emilia was chosen for the study because it serves as the headquarters of Coopservice and registers a high number of alarms, as previously mentioned. However, the company operates in several other provinces and aims to apply similar models to those tested in different locations. If a province already holds a substantial database, one may replicate the analysis conducted for Reggio Emilia, crafting an exclusive model for the province. In cases where the company lacks historical information, utilizing the model from existing provinces through transfer learning becomes an alternative. In other words, the goal is to assess how well our best model, trained in Reggio Emilia, can be generalized and evaluate its performance in a scenario where there is still no data available for training. We cover this approach in this chapter.

According to Figure 2, the top five provinces with the highest number of alarms, from highest to lowest, are Bologna, Reggio Emilia, Firenze, Modena, and Roma. In this chapter, we exclusively use the most successful approach, which is the model xgb_re_kpp_loe. It was trained with data from Reggio Emilia and will be used to predict the number of alarms for the other provinces in the top 5.

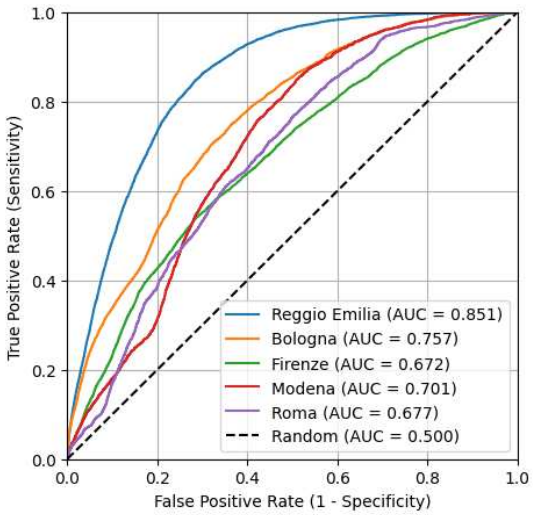
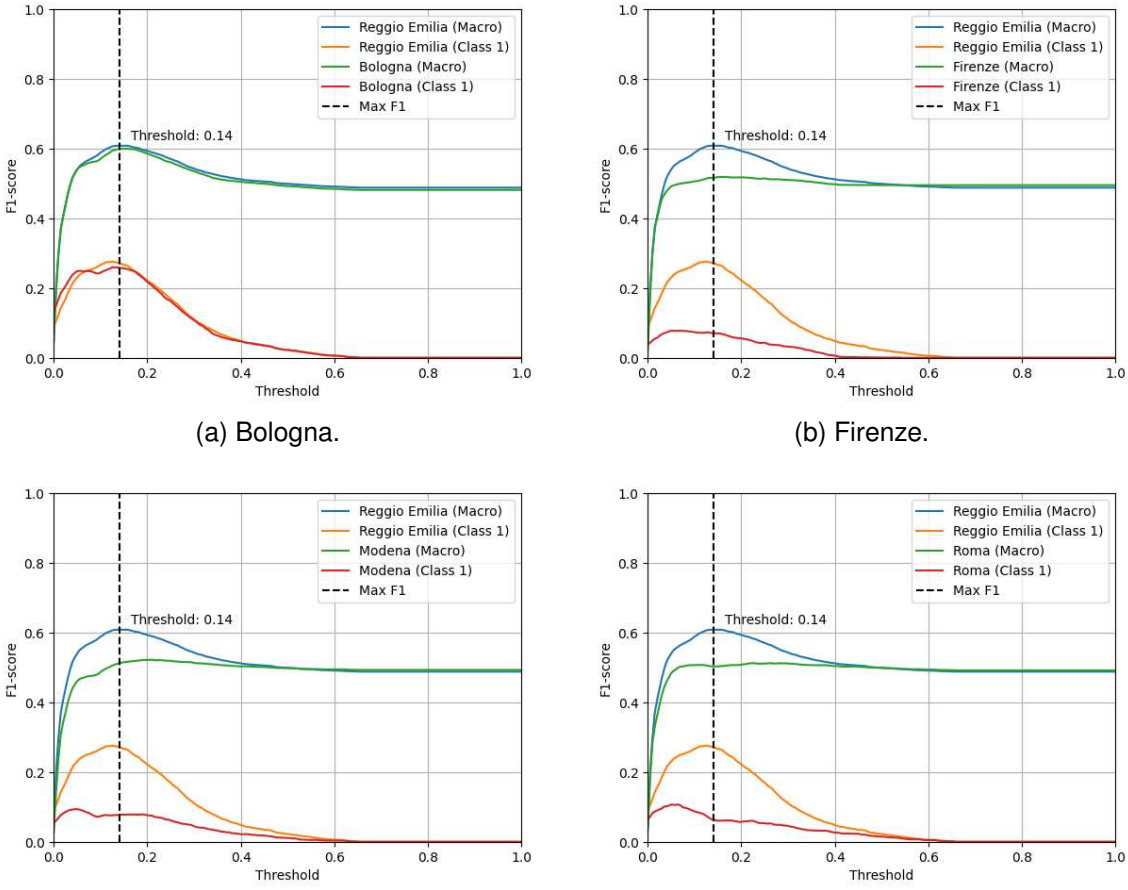


Figure 17 – Comparison of ROC curves for the XGB model trained in Reggio Emilia and applied to different provinces, including Bologna, Firenze, Modena, and Roma.

Figure 17 presents the ROC curve and AUC values for our best model, the one trained in Reggio Emilia with LOE encoding and KPP clustering, applied in the provinces of Bologna, Firenze, Modena, and Roma. It shows that the model adapts well to Bologna with an AUC of 0.756 but for the Firenze, Modena, and Roma, it ranges from 0.700 or lower. Nevertheless, in the worst cases, Firenze and Roma, it exceeds the random model by 1.720 in the AUC score. Despite the results not being very promising, except for Bologna, it is a positive sign that for all the provinces the model can outperform the random model by a significant margin.



(a) Bologna.

(b) Firenze.

(c) Modena.

(d) Roma.

Figure 18 – F1-score by thresholds for all provinces compared to Reggio Emilia. The black vertical dashed line represents the threshold achieving the maximum Macro F1-score.

As seen in Section 5.1 of the previous chapter, we obtained good F1-score values for the non-alarm class using XGB. However, we did not find satisfactory F1-score values for the alarm occurrence class, and consequently, low macro F1-score values as well. Figure 18 illustrates how our best model performs for the selected provinces compared to Reggio Emilia. We achieve the maximum MF1-score and also the maximum F1-score for class 1 (alarm occurrence) when setting the threshold to 0.14. Bologna's behavior is very close to what we obtain in Reggio Emilia, reinforcing the validity of transfer

learning for this province. On the other hand, the remaining provinces obtained values significantly lower than Reggio Emilia, coupled with poor AUC scores, making them unsuitable for use with the Reggio Emilia model.

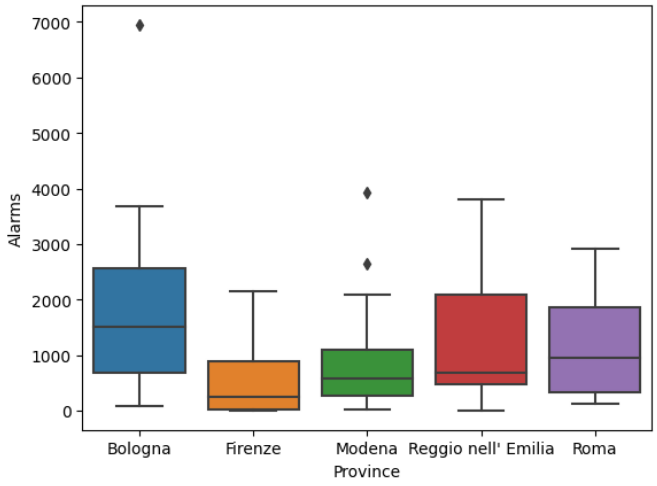
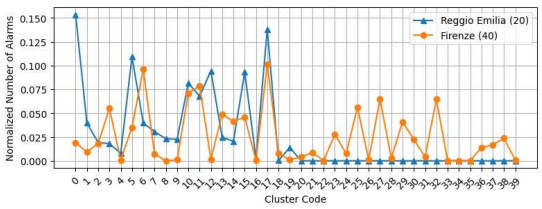


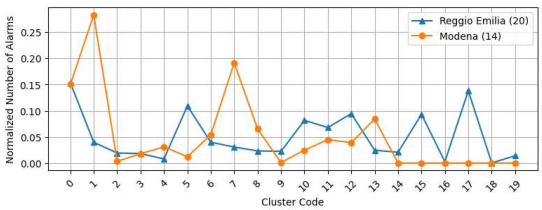
Figure 19 – Boxplot illustrating alarm occurrence distributions across different provinces.

To attempt to explain the behavior of AUC, we decide, through Figure 19, to evaluate the alarm distribution in these provinces. Initially, one might consider that provinces with alarm distributions similar to Reggio Emilia might yield better AUC values, but that's not what we find. Bologna, having a similar distribution, produced the best results, but Roma also had a comparable distribution, yet the model obtained the worst AUC among all the provinces, tied with Firenze.

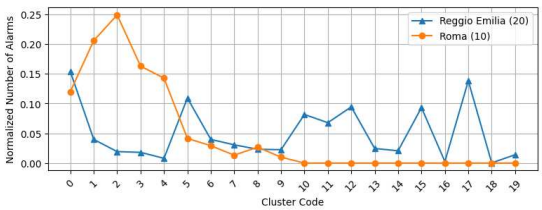
(a) Bologna.



(b) Firenze.



(c) Modena.

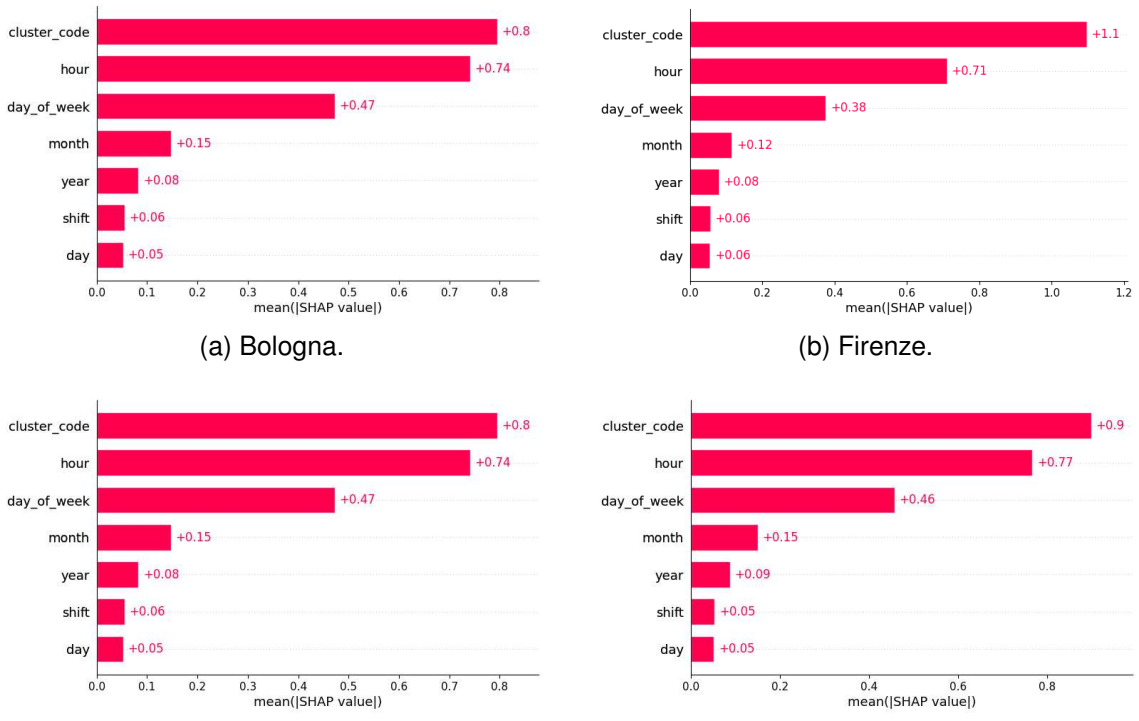


(d) Roma.

Figure 20 – Distribution of alarms by cluster for the provinces of Bologna, Firenze, Modena, and Roma compared to Reggio Emilia.The values in parentheses represent the total number of clusters in each region.

There is a random factor that we cannot control in this transfer learning process, which is how the identification value for each cluster_code will be determined. Analyzing

the values of the clusters in Reggio and Bologna through Figure 20, one can observe the behavior of the curves. It is evident that there is a certain proximity between the patterns, following the falls and rises. However, this behavior is coincidental and may affect the model's performance. If, during the distribution of cluster values, cluster 2 of Reggio Emilia is assigned to a cluster with a low number of alarms, the model will be trained to believe in this behavior. Looking at cluster 2 of Bologna has several alarms very close to what we expect in Reggio, and this can assist the model in making predictions. In Rome, the same cluster 2 exhibits a behavior opposite to Reggio, which can impair the model's predictions. Since the goal is to apply the model from one province to another without historical data, we cannot directly address this randomness as we do not know the behavior of the clusters in the target province. It is worth noting that as new information becomes available, it is possible to adapt the distribution of numerical cluster identifications to maximize the similarity between the alarm distribution, potentially leading to models that generate better results.



(a) Bologna.

(b) Firenze.

(c) Modena.

(d) Roma.

Figure 21 – Feature ranking based on the absolute value of SHAP scores for different provinces.

In Figure 21, we observe that the model trained in Reggio Emilia defines the same feature importance order when applied to the new provinces, following the pattern observed previously for clustering in KPP. All provinces exhibit analogous behaviors, predominantly differing in the specific SHAP values allocated. The prominence of cluster_code over hour, particularly in Firenze, stands out as a unique deviation. Unlike Firenze, other provinces have fewer than 20 clusters, Modena and Bologna with 14

each, and Roma with 10, all included in the Reggio Emilia model. Firenze's 40 clusters may lead to model extrapolation issues, affecting SHAP values and AUC.

Based on the obtained results, it is evident that there is a possibility of applying models trained in one province to another, ensuring a generalization of the problem in some cases. However, it is important to conduct recurrent analyses to assess the results and behavior of new provinces, as good results are not always guaranteed.

7 CONCLUSIONS

In this dissertation, we evaluated 12 machine learning models to assess their quality in predicting alarms based on geographical location through clustering using data from an Italian company. These models are divided into 3 algorithms: Naive Bayes (NB), XGBoost (XGB), and Multilayer Perceptron (MLP), 2 encoding methods: Label/Ordinal Encoding (LOE) and Label/Ordinal/One-Hot Encoding (L2OE), and 2 clustering methods: COOP (Coopservice-2022) and K-Means++ (KPP). The results showed that XGB is the strategy that provided the highest AUC values, 0.795 and 0.796 for COOP-clustered models, and 0.846 and 0.847 for KPP-clustered models. Regarding encoding methods, it was observed that for XGB, there was no significant change in AUC values. Still, for models derived from NB and MLP, a significant improvement was obtained by applying L2OE instead of LOE. The use of K-Means++ for clustering alarms significantly impacted the dataset, with an approximate gain of 0.051 in AUC for XGB models for both encoding strategies.

Hyperparameter tuning was performed for the 4 XGB models, where it was not possible to determine a set of hyperparameters that significantly outperformed the default values. SHAP value analyses showed that for XGB models applying LOE, the most important features are cluster_code (cluster identification) and the hour when the alarm occurred or not. The XGB COOP model selected hour as more relevant, and the XGB KPP model selected cluster_code, providing further evidence that the alarm clustering approach impacts predictive model results.

To determine the model's generalization capability, a transfer learning study was conducted, demonstrating that models trained in one province can be applied to another. It is important to note that constant monitoring is required when using models from one province to another, as the model is highly sensitive to the label assigned to each cluster. Clusters with similar behavior in different provinces labeled with the same cluster_code can positively impact the model's performance.

Transfer learning is also directly linked to the original model's ability to make predictions. Therefore, it is essential to work on improving its performance before applying it to other provinces. In conclusion, more studies are needed, as the model trained and applied in the same province showed good overall performance considering both classes (alarm and non-alarm). However, when analyzing the alarm class, the

model does not perform well, unable to exceed an F1-score of 0.3. The imbalanced nature of the dataset contributes to the models favoring the majority class (non-alarm) which represents 95% of the dataset and not giving as much importance to the minority class.

The contributions of this work pave the way for several potential avenues of future research, providing a ground for enhancements and refinements. Some potential directions for subsequent investigations include:

- **Specialized Techniques for Imbalanced Data:** exploring methods specialized in handling imbalanced datasets, such as oversampling and undersampling, is a promising area. Strategies like SMOTE (Synthetic Minority Over-sampling Technique) can be employed to create synthetic instances of the minority class, potentially improving model performance by assigning equitable importance to both classes during training.
- **One-Class Algorithms:** Investigating approaches more focused on one-class problems, like One-Class SVM and Isolation Forest, may yield valuable insights. By concentrating solely on occurrences of the positive class (alarms), we can tailor modeling strategies to more effectively deal with the imbalanced nature of the dataset.

REFERENCES

AHMED, M.; MAHMOOD, A. N.; HU, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, Elsevier, v. 60, p. 19–31, 2016. Citado na página 17.

ARTHUR, D.; VASSILVITSKII, S. K-means++: The advantages of careful seeding. In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. USA: Society for Industrial and Applied Mathematics, 2007. (SODA '07), p. 1027–1035. ISBN 9780898716245. Citado 2 vezes nas páginas 18 and 19.

AU-YEUNG, W.-T. M. et al. Reduction of false alarms in the intensive care unit using an optimized machine learning based approach. *npj Digital Medicine*, v. 2, n. 1, p. 86, Sep 2019. ISSN 2398-6352. Disponível em: <https://doi.org/10.1038/s41746-019-0160-7>. Citado 2 vezes nas páginas 13 and 29.

BHOWAN, U.; JOHNSTON, M.; ZHANG, M. Developing new fitness functions in genetic programming for classification with unbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 42, n. 2, p. 406–421, 2011. Citado 2 vezes nas páginas 24 and 42.

BRAVO, F. et al. Machine learning aplicado à predição da obrigação do investimento em p,di. In: *Anais do XXXVII Simpósio Brasileiro de Bancos de Dados*. Porto Alegre, RS, Brasil: SBC, 2022. p. 216–228. ISSN 2763-8979. Disponível em: <https://sol.sbc.org.br/index.php/sbbd/article/view/21808>. Citado na página 41.

BUTT, U. M. et al. Machine learning based diabetes classification and prediction for healthcare applications. *Journal of healthcare engineering*, Hindawi, v. 2021, 2021. Citado na página 22.

CHAPELLE, O.; WESTON, J.; SCHöLKOPF, B. Cluster kernels for semi-supervised learning. In: BECKER, S.; THRUN, S.; OBERMAYER, K. (Ed.). *Advances in Neural Information Processing Systems*. MIT Press, 2002. v. 15. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2002/file/d6288499d0083cc34e60a077b7c4b3e1-Paper.pdf>. Citado na página 18.

CHEN, T.; GUESTRIN, C. XGBoost. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. Disponível em: <https://doi.org/10.1145%2F2939672.2939785>. Citado na página 20.

CHO, H. *xgboost 1.7.5*. 2023. <https://pypi.org/project/xgboost/>. Accessed on: May 1, 2023. Citado na página 42.

DEVELOPERS, x. *XGBoost Documentation*. 2023. <https://xgboost.readthedocs.io/en/stable/index.html>. Accessed on: November 30, 2023. Citado na página 48.

DMLC. *XGBoost: Scalable and Flexible Gradient Boosting*. 2023. <https://github.com/dmlc/xgboost>. Citado na página 48.

EKANAYAKE, I.; MEDDAGE, D.; RATHNAYAKE, U. A novel approach to explain the black-box nature of machine learning in compressive strength predictions of concrete using shapley additive explanations (shap). *Case Studies in Construction Materials*, v. 16, p. e01059, 2022. ISSN 2214-5095. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214509522001917>. Citado 2 vezes nas páginas 27 and 51.

GOOGLE. *Google Colaboratory*. 2019. <https://colab.research.google.com>. Accessed on: May 1, 2023. Citado na página 42.

GRAMEGNA, A.; GIUDICI, P. Shap and lime: an evaluation of discriminative power in credit risk. *Frontiers in Artificial Intelligence*, Frontiers Media SA, v. 4, p. 752558, 2021. Citado na página 27.

HSU, A. et al. Predicting european cities' climate mitigation performance using machine learning. *Nature Communications*, Nature Publishing Group UK London, v. 13, n. 1, p. 7487, 2022. Citado na página 21.

JAIN, A. *Mastering XGBoost Parameter Tuning: A Complete Guide with Python Codes*. 2016. Accessed on 2023-11-30. Disponível em: <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/>. Citado na página 48.

KANSAL, T. et al. Customer segmentation using k-means clustering. In: *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*. [S.l.: s.n.], 2018. p. 135–139. Citado na página 17.

LATEANO, F. et al. Machine-learning-assisted failure prediction in microwave networks based on equipment alarms. In: *2023 19th International Conference on the Design of Reliable Communication Networks (DRCN)*. [S.l.: s.n.], 2023. p. 1–7. Citado 2 vezes nas páginas 13 and 31.

LI, Z. Extracting spatial effects from machine learning model using local interpretation method: An example of shap and xgboost. *Computers, Environment and Urban Systems*, v. 96, p. 101845, 2022. ISSN 0198-9715. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0198971522000898>. Citado 2 vezes nas páginas 26 and 51.

LUNDBERG, S. M.; LEE, S.-I. A unified approach to interpreting model predictions. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 4768–4777. ISBN 9781510860964. Citado 2 vezes nas páginas 26 and 51.

MANGALATHU, S.; HWANG, S.-H.; JEON, J.-S. Failure mode and effects analysis of rc members based on machine-learning-based shapley additive explanations (shap) approach. *Engineering Structures*, v. 219, p. 110927, 2020. ISSN 0141-0296. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0141029620307513>. Citado 2 vezes nas páginas 27 and 51.

MENG, Y.; KWOK, L.-f. Adaptive false alarm filter using machine learning in intrusion detection. In: WANG, Y.; LI, T. (Ed.). *Practical Applications of Intelligent Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 573–584. ISBN 978-3-642-25658-5. Citado 2 vezes nas páginas 13 and 29.

MOHANTY, A.; MISHRA, S. A comprehensive study of explainable artificial intelligence in healthcare. In: *Augmented Intelligence in Healthcare: A Pragmatic and Integrated Analysis*. [S.l.]: Springer, 2022. p. 475–502. Citado na página 27.

NAZ, S.; MAJEED, H.; IRSHAD, H. Image segmentation using fuzzy clustering: A survey. In: IEEE. *2010 6th international conference on emerging technologies (ICET)*. [S.l.], 2010. p. 181–186. Citado na página 18.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 42.

PEZZE, D. D. et al. Formula: A deep learning approach for rare alarms predictions in industrial equipment. *IEEE Transactions on Automation Science and Engineering*, v. 19, n. 3, p. 1491–1502, 2022. Citado na página 30.

QUINN, C. *ALARM FORECASTING IN NATURAL GAS PIPELINES*. Dissertação (Mestrado), Milwaukee, Wisconsin, 2020. Disponível em: <https://epublications.marquette.edu/theses_open/577/>. Citado 2 vezes nas páginas 13 and 30.

REIS, J. C. et al. Explainable machine learning for fake news detection. In: *Proceedings of the 10th ACM conference on Web Science*. [S.l.: s.n.], 2019. p. 17–26. Citado na página 27.

RISH, I. et al. An empirical study of the naive bayes classifier. In: *IJCAI 2001 workshop on empirical methods in artificial intelligence*. [S.l.: s.n.], 2001. v. 3, n. 22, p. 41–46. Citado na página 19.

ROKACH, L.; MAIMON, O. Clustering methods. In: ____. *Data Mining and Knowledge Discovery Handbook*. Boston, MA: Springer US, 2005. p. 321–352. ISBN 978-0-387-25465-4. Disponível em: <https://doi.org/10.1007/0-387-25465-X_15>. Citado 2 vezes nas páginas 17 and 18.

Rutgers University. *Rutgers Study Finds Alarm Systems Are Valuable Crime-Fighting Tool*. 2009. <https://www.rutgers.edu/news/rutgers-study-finds-alarm-systems-are-valuable-crime-fighting-tool>. Accessed on May 15, 2023. Citado na página 13.

SNIJDERS, T. A. B. On cross-validation for predictor evaluation in time series. In: DIJKSTRA, T. K. (Ed.). *On Model Uncertainty and its Statistical Implications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988. p. 56–69. ISBN 978-3-642-61564-1. Citado na página 25.

SUTER, B. W. The multilayer perceptron as an approximation to a bayes optimal discriminant function. *IEEE transactions on neural networks*, v. 1, n. 4, p. 291, 1990. Citado na página 21.

TU, Z. et al. Maxim: Multi-axis mlp for image processing. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2022. p. 5769–5780. Citado na página 22.

WEYTJENS, H.; LOHMANN, E.; KLEINSTEUBER, M. Cash flow prediction: Mlp and lstm compared to arima and prophet. *Electronic Commerce Research*, Springer, v. 21, n. 2, p. 371–391, 2021. Citado na página 22.

ZHANG, H. The optimality of naive bayes. In: . [S.l.: s.n.], 2004. v. 2. Citado na página 19.

ZHANG, M.; WANG, D. Machine learning based alarm analysis and failure forecast in optical networks. In: *2019 24th OptoElectronics and Communications Conference (OECC) and 2019 International Conference on Photonics in Switching and Computing (PSC)*. [S.l.: s.n.], 2019. p. 1–3. Citado 2 vezes nas páginas 13 and 30.

ZHAO, W. et al. SHAP values for explaining cnn-based text classification models. *CoRR*, abs/2008.11825, 2020. Disponível em: <https://arxiv.org/abs/2008.11825>. Citado na página 27.

ZHU, J. et al. Dynamic alarm prediction for critical alarms using a probabilistic model. *Chinese Journal of Chemical Engineering*, v. 24, n. 7, p. 881–885, 2016. ISSN 1004-9541. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1004954116303044>. Citado 2 vezes nas páginas 13 and 29.

ZHU, Q. et al. Using mlp features in sri's conversational speech recognition system. In: CITESEER. *Ninth European Conference on Speech Communication and Technology*. [S.l.], 2005. Citado na página 22.

ZHUANG, H. et al. Machine-learning-based alarm prediction with gans-based self-optimizing data augmentation in large-scale optical transport networks. In: *2020 International Conference on Computing, Networking and Communications (ICNC)*. [S.l.: s.n.], 2020. p. 294–298. Citado 2 vezes nas páginas 13 and 30.

ZUCCHI, G. et al. A metaheuristic algorithm for a multi-period orienteering problem arising in a car patrolling application. In: *INOC*. [S.l.: s.n.], 2022. p. 99–104. Citado na página 14.