Data Analysis and Processing From Remote Sensors for Detection Of Forest Fires Risk

Vitor Fernandes Marcelino

Work carried out under the guidance of:

Ana Isabel Pereira

José Luis Lima

Luis Paulo Fagundes

Master's degree in Industrial Engineering

2022-2023

Data Analysis and Processing From Remote Sensors for Detection Of Forest Fires Risk

Project Thesis

Master's degree in Industrial Engineering

Escola Superior de Tecnologia e Gestão

Vitor Fernandes Marcelino

2022-2023

Escola Superior de Tecnologia e de Gestão is not responsible for the opinions expressed in this document.

Certify that I have read this dissertation and that, in my opinion, is appropriate in content and form as a demonstrator of the developed work.

Vitor Fernandes Marcelino

Dedication

I dedicate this work to my family for their constant love and support, to my girlfriend, to my friends and colleagues, and to all my teachers at IPB and CEFET.

Acknowledgment

I would like to express my sincere gratitude to my family, my mother, Elivania, my father, Gilberto, my sister, Gabriela, and my godmother, Silvana, for all their support throughout this adventure. During this stage, I would also like to thank my girlfriend, Renata, for her continuous support and encouragement throughout this work. I would also like to thank my friends in Brazil for their constant support and conversation, and my friends who lived this whole experience with me during this period, especially the Team and Central da Copa. I would like to thank my partners at CEDRI, João Mendes and Thadeu Brito and at IPB for their priceless help with the data, code and trust, without which this study would not have been possible. I would also like to thank the teachers at CEFET, especially professor Luís Paulo Fagundes, for their support during the course and the project. This work was only possible to be accomplished thanks to my mentor and teacher Ana Isabel, who trusted in my potential and supported me throughout this project. Thank you Ana, for all the trust and encouraging conversations we had during this whole process.

Abstract

This work focuses on developing fire risk detection and prevention algorithms using data collected by sensors in the forest. The study involved a State of the Art review and a Theoretical Foundation, followed by Data Characterization and Data Analysis, which were divided into several sub-sections. The study developed regression models for different types of data and found that the random forest regression model was the best performing for transition times. The study compared different regression models, finding that the Support Vector Regression (SVR) model performed worse than the Gradient Boosting Regression (GBR) and Random Forest Regression (RFR) models.

The study concluded that using algorithms to identify periods of the day was a useful strategy for avoiding false alerts and that training the models for each individual module was the best strategy. Furthermore, the RFR and GBR regression models were found to be the most effective for the data available in this study. However, improvements are necessary to reduce false positives and facilitate abnormality detection.

Overall, this work provides insight into the most effective methods for analyzing and processing data collected by sensors in the forest for fire risk detection and prevention, with the potential to create alerts for those involved in fighting forest fires.

Keywords: Forest Fires, Machine Learning, Data Science, Classification

Resumo

Este trabalho centra-se no desenvolvimento de algoritmos de detecção e prevenção de riscos de incêndio utilizando dados recolhidos por sensores na floresta. O estudo envolveu uma revisão do estado da arte e uma Fundação Teórica, seguida de Caracterização e Análise de Dados, que foi dividida em várias subsecções. O estudo desenvolveu modelos de regressão para diferentes tipos de dados e descobriu que o modelo de random forest regression era o que tinha melhor desempenho para os dados de transição dia e noite. O estudo comparou diferentes modelos de regressão, descobrindo que o modelo de Support Vector Regression (SVR) teve pior desempenho do que os modelos de Gradient Boosting Regression (GBR) e de Random Forest Regression (RFR).

O estudo concluiu que a utilização de algoritmos para identificar períodos do dia era uma estratégia útil para evitar falsos alertas, e que o treino dos modelos para cada módulo individual era a melhor estratégia. Além disso, os modelos de regressão RFR e GBR foram considerados como os mais eficazes para os dados disponíveis neste estudo. No entanto, são necessárias melhorias para reduzir os falsos positivos e facilitar a detecção de anomalias.

Dessa forma, este trabalho fornece uma visão dos métodos mais eficazes para analisar e processar os dados recolhidos pelos sensores na floresta para detecção e prevenção de riscos de incêndio, com o potencial de criar alertas para os envolvidos no combate aos incêndios florestais.

Palavras-chave: Incêndios Florestais, Aprendizagem de Máquinas, Ciência de Dados, Classificação.

Contents

List of Tables

List of Figures

Siglas

AI Artificial Intelligence. 10, 11

ARI Adjusted Rand Index. 41

CARTs Classification and Regression Trees. 7

DL Deep Learning. 5, 12

DM Data mining. 12, 13

DT Decision Tree. 6, 13, 14, 38

GBR Gradient Boosting Regressor. 42–47, 49, 50, 53–60, 62–64, 70–73, 75, 77, 78

IBM International Business Machines Corporation. 11

IoT Internet of Things. 5, 7

IPB Polytechnic Institute of Bragança. 4, 21

MAE Mean Absolute Error. 20, 42–46, 54, 55, 59, 62, 64, 71

ML Machine Learning. 5–8, 11, 13, 38, 42

MSE Mean Squared Error. 20, 42–44, 46, 55–57, 59, 62–64, 71, 75

R-Square Coefficient of Determination. 20, 42–45, 52, 53, 58, 62, 66, 69, 70

RF Random Forest. 6, 7, 13, 15

RFR Random Forest Regressor. 42–45, 47–51, 53–55, 57, 59–64, 66, 70–75, 77, 78

RMSE Root Mean Squared Error. 20, 42–44, 57, 60, 62, 63, 72, 75

SAFe Forest Alert Monitoring System. 3, 4, 21

SVM Support Vector Machine. 7, 18, 47–49, 59, 62, 70

SVR Support Vector Regressor. 42, 43, 50, 54, 55, 59, 77

SWOT Strengths, Weaknesses, Opportunities, and Threats. 8

WCSS Within-Cluster Sum of Square. 17

WSN Wireless Sensor Network. 10

Chapter 1

Introduction

In this chapter, we present an introduction to this work, seeking to describe the main motivations that supported this study and the objectives it aims to accomplish. Forest fires are one of the most dangerous problems in the world. It can be spontaneous or not, but when the fire spreads, it is very difficult to control that damage. According to [1], 90% of forest fire incidents in the world have occurred as a result of human carelessness. Climate change has also contributed to the increase in forest fires worldwide, through more extreme weather conditions and droughts that increase the spread of fire [2], [3]. Forest fires are an imminent hazard for people and animals and cause air pollution, which can cause respiratory diseases more frequently. In 2022, the annual technical report of the Joint Research Centre (JRC), the European Commission's science and knowledge service, indicates that due to high summer temperatures, the countries bordering the Mediterranean Sea have recorded the most severe forest fires in Europe [4], [5].

Although the Mediterranean Sea does not surround Portugal, due to its geographical proximity, the country has similar climatic characteristics to other Mediterranean countries, meaning that with all the reported problems with forest fires, Portugal has also experienced similar forest fire increases [5], [6]. To solve the problem, the government has implemented measures such as aerial firefighting, increased surveillance, and forest management [5], [7].

Considering the importance of the issue, this problem is a common topic of discussion

and research in the world. The detection of forest fires has been the focus of a lot of research in the last decade due to the increase in forest fires, which can cause serious damage to society and the environment [1]. The detection of forest fires is one of the main strategies used to minimize the damage caused by these events, allowing combat teams to act quickly and efficiently [8], [9].

Thus, many innovative operations have been developed to monitor new areas of risk and strengthen existing surveillance systems. Predicting the wildfire will be a great step toward the preservation [10]. In this context, the application of machine learning techniques has shown promise, allowing the development of models capable of learning to detect and predict forest fires from data from various sources, such as satellite images, meteorological data, and information from sensors on the ground [8], [9], [11], [12].

1.1 Motivation

The Forest Alert Monitoring System (SAFe) Project aims to create and implement a set of innovative actions that will minimize the occurrence of forest fires and monitor flora and fauna. In order to contribute to the development of the Trás-os-Montes region in the district of Bragança [13]–[16].

In doing so, a set of sensors was implemented in the forest, where data regarding the most relevant variables for identification are collected from forest ignitions. Once this data is collected, it is filtered through an artificial intelligence system. In previous works [13]–[16], modules were developed, which are composed of several types of sensors, such as flame detection sensors, temperature sensors and humidity. Considering the region's size to be monitored, data transmission is carried out by LoRaWAN, it is based on low power wide area network

In this sense, this work is motivated to contribute to the advancement of the SAFe project with this studies.

1.2 Objective

The main objective of this work is to develop algorithms for fire risk detection and prevention based on data collected from the prototypes developed by SAFe, test data, with sensors fixed on campus and near the Polytechnic Institute of Bragança (IPB). The idea is to verify the behavior of algorithms for real data, checking possible improvements for false alarm problems. In particular, this work will explore:

- Regression techniques to identify patterns,
- Techniques to identify day and night periods,
- Techniques of clustering to identify similar situations among the sensors,
- Metrics for evaluating regression models.

1.3 Document Structure

The present work is organized into seven chapters. The first chapter aims to introduce the problem, motivation and objectives of this work. The second chapter has as its purpose to realize a state of the art on the studied theme. Based on the state of the art, the third chapter aims to illustrate the research on the subjects approached during this work. The fourth chapter is intended as a characterization of the data analyzed. The fifth chapter is the analysis of the data and results obtained. The sixth chapter aims to conclude the entire study and analysis carried out and, finally, the seventh chapter has suggestions for future work.

Chapter 2

State of Art

In this chapter, we present a review of research on the main theme of this project. Many scientists, in addition to looking for a solution to the causes of forest fires, also seek to predict these fires in order to minimize the damage caused [17]. Currently, technology is an ally in solving various problems in the world. Machine Learning (ML) algorithms are one technology used to predict events in many real-life activities, citing a few as an example: stock market crisis forecast [18], [19] and extreme weather events forecast [20], [21]. Predict the occurrence of such event can be beneficial in making decisions about how to deal with such an event and be prepared to prevent further wildfires [10].

Therefore, some criteria were used for the state-of-the-art investigation, the inclusion criteria were studies published in the last 10 years and studies related to forest fires. In Table 2.1 query results from Google Scholar, IEEE Xplore^{TM}, ScienceDirect and Scopus using keywords like ML, Forest Fires, Forest Fires Detection, Internet of Things (IoT), Deep Learning (DL) and Neural Network are presented. Only results from 2012 onwards were considered.

Table 2.1: Results of literature review.

Keywords	**Google Scholar**	**Scopus**	**Science Direct**	**IEEE**	**Web of Science**
Forest Fire	149000	20148	48390	1059	22443
Forest Fire Detection	52700	2048	13353	442	1822
Machine Learning	819000	375641	234482	124894	292268
Deep learning	1840000	250406	221421	96279	218891
Neural Network	1490000	510535	286009	272366	433297
IoT	550000	107472	32495	57366	78142
Forest Fire + Machine Learning	29400	506	4605	97	440
Forest Fire + Deep learning	28000	219	4296	74	178
Forest Fire + Neural Network	17300	408	3833	108	354
Forest Fire + IoT	10700	158	694	52	99
Forest Fire + Machine Learning + Deep learning	20500	77	2459	26	54
Forest Fire + Machine Learning + Neural Network	17500	120	2420	39	118
Forest Fire + Machine Learning + IoT	10300	13	450	4	7
Forest Fire + Deep learning + Neural Network	17300	113	1816	43	99
Forest Fire + Deep learning + IoT	8980	12	346	1	10
Forest Fire + Neural Network + IoT	6570	17	349	2	8
Forest Fire + Machine Learning + Deep learning + Neural Network	17700	44	1577	16	38
Forest Fire + Machine Learning + Deep learning + IoT	17700	2	328	1	2
Forest Fire + Deep learning + Neural Network + IoT	19000	6	273	1	2
Forest Fire + Machine Learning + Deep learning + Neural Network + IoT	20300	2	267	1	2

The data presented in Table 2.1 show that there is a large number of research being developed on the keyword ML, even when looking only for studies that involve the whole set (Machine Learning and Forest Fires Detection). In this way, the use of ML techniques to detect forest fires is a reality. Some models already developed using learning algorithms have already performed well in some tests [1], [22].

In addition to this research, using the software VosViewer and a search on the Scopus website, Figure 2.1, we can analyze a network of keywords related to the main theme of the work, forest fires. With this, it is possible to observe that the keywords: Decision Tree (DT), ML, Random Forest (RF) are the main ones used in works on this topic.

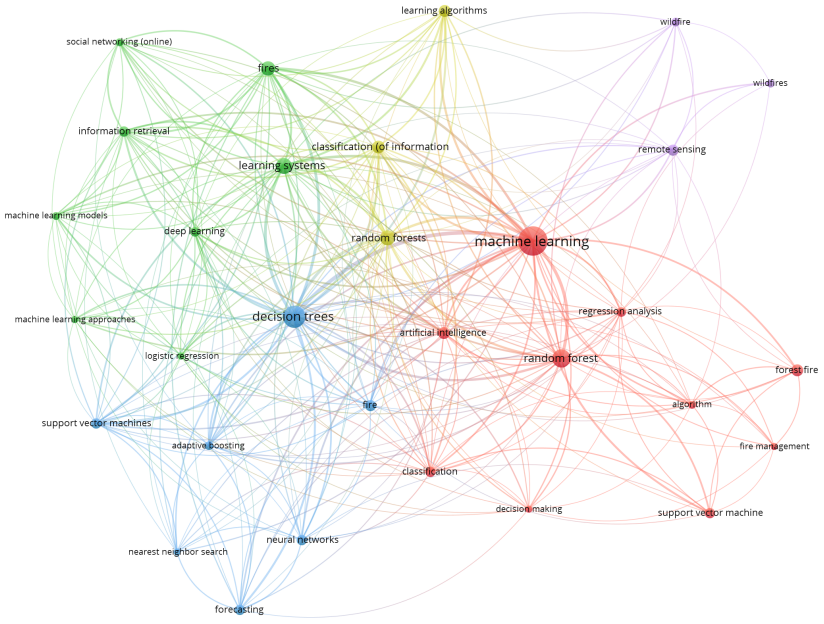


Figure 2.1: Network of Keywords Search in Scopus.

With the help of these highlighted keywords, the Web of Science search is used as a great basis for this work. Thus, it was possible to observe some works that show the use of ML algorithms in detecting forest fires. The work developed by Sharma [23], demonstrates how IoT combined with ML allows an increase in the efficiency of fire detection when compared to image analysis and rangers. Furthermore, the work shows performance comparisons between ML models.

The Zope's work (2020),[10], shows a forest fire detection framework using IoT sensors and ML algorithms, showing that simple logistic regression works very well for this type of problem. However, the project uses a Dense Neural Network created using Keras API. The accuracy of this developed model was 91%.

In the work published by Nosouhi (2022),[24], in addition to proposing a data analysis with IoT and ML sensors, the author mentions an important objective in works with algorithms, which is to reduce the number of false positive detections. Classification and Regression Trees (CARTs), RF and Support Vector Machine (SVM) were used.

Another relevant research publication was that of Xie and Peng (2018), [25] which shows comparisons between models of forest fire detection algorithms. The main works present in the research report is the use of temperature and humidity sensors, both data are used during the present work. In this context, with this information, the use of different types of regression algorithms was observed to predict the information from the sensors.

These academic researches plus the work developed at IPB [13]–[16], serve as a basis for the work developed during this project.Thus, it is necessary to carry out a research on the reasoning of the keywords.

During the research was created a Strengths, Weaknesses, Opportunities, and Threats (SWOT) analysis. ML might fit into each category for the application of predicting and responding to forest fires. The strengths can be that ML algorithms can analyze large amounts of data quickly and accurately, potentially identifying patterns and trends and can be used to make predictions about the probability of a forest fire occurring. The weaknesses can be that ML algorithms are only as good as the data they are trained on, the predictions made by the algorithm may not be accurate according to the data and ML also can be complex and difficult to understand. Furthermore, using ML may present opportunities for growth, expansion, or improvement for the project. They may include changes in the process of detecting forest fires.

Chapter 3

Theoretical Foundation

In this chapter, we present a theoretical foundation in order to support the conception of this work. Forest fires are one of the most serious problems in the world. They can be natural or not, but when the fire spreads, it is very hard to keep it under control. This problem is a common topic of discussions and studies in the world, therefore, there are a lot of techniques to try to control this situation. This chapter shows some technologies that can be applied to minimize this damage.

3.1 Detection Techniques

Nowadays due to the advance in technology, there are some methods used to identify forest ignitions [25]. They are:

Camera Surveillance

This type of sensor works according to the images produced by a camera. In this case, the camera produces an image analyzed by a program to check if it contains smoke or fire [25]. However, it is very common to happen some fake alarms, like during fog or sunlight, which consequently can affect the reliability of the result [13].

Wireless Sensor Network

Wireless Sensor Network (WSN) are nowadays receiving more attention when we talk about studies in forest fire detection [25]. According to [13], the WSN is a set of autonomous sensors from energy and data transmission. With these sensors, we can monitor the parameters such as temperature, humidity, and ultraviolet radiation, among others.

3.2 SAFe Sensor Modules

The SAFe sensor modules were developed internally at IPB in the project SAFe. The modules have five flame sensors, a humidity and temperature sensor. The module is controlled through an Atmega328P as the main hardware of the system. All communication between the module and the database is carried out with LoRa communication modules. The LoRa communication system is a high-bandwidth wide area network technology with low power consumption, enabling long-range transmissions, about 10 kilometers in rural areas. [13]–[16]. Figure 3.1 shows a side view of the module.

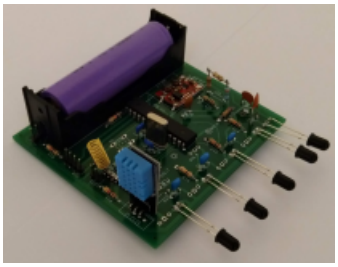


Figure 3.1: Sensor Module SAFe [13], [14].

3.3 Machine Learning

Given all this data generated by such sensors, there is a need for data processing to detect forest fires. With this big data comes the possibility of using technologies with more development, such as Artificial Intelligence (AI) systems. According to [26]–[28], AI

is anything capable of mimicking human behavior. AI these days is a term that describes solutions that learn on their own.

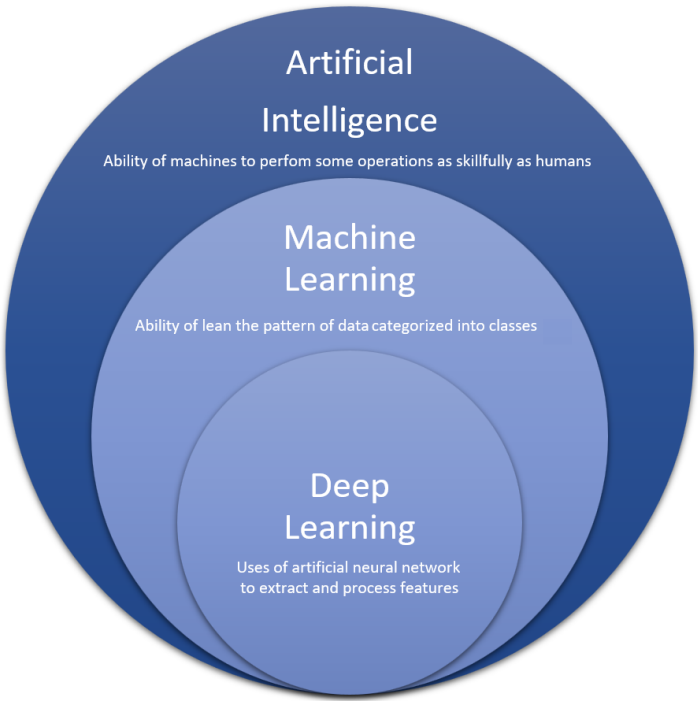


Figure 3.2: Specialization of AI algorithms Adapted from [26].

The Figure 3.2 shows a representation of AI and its sub-division into Machine Learning and deep learning. Regarding Figure 3.2, ML is a sub-division of AI, this term was created in 1969 by Arthur Samuel at the International Business Machines Corporation (IBM). Arthur Samuel defined ML as the "field of study that gives computers the ability to learn without being explicitly programmed" [29]. ML algorithms can be simplified as mathematical functions that contain many parameters that map features or inputs into one or more outputs or targets. In order to train a ML algorithm is necessary to optimize the parameters to map the features to the targets accurately [30].

According to the input data, this type of algorithm can be defined into four subdivisions:

- Supervised learning

  This type is defined by the use of labeled datasets to train the ML algorithms, with

training, an already known output is expected [31]. All the knowledge is obtained through examples. In a nutshell, supervised learning models can be divided into two types of problems, Regression and Classification.

- Semi-supervised learning

  According to [32], semi-supervised learning trains the labeled data using unlabeled data to compensate for the lack of labeled data. The semi-supervised classifier is better than the classifier that trains only labeled data

- Unsupervised learning

  This kind is a powerful tool to identify latent structures in unlabeled time series data, the goal is to organize as much as possible the data into clusters, which could significantly reduce the computational efforts [33].

- Reinforcement learning

  The subdivisions in the previous topics learn to perceive, encode, predict or sort patterns, but do not learn to act in the more general sense of Reinforcement Learning. These codes must figure out how to interact with a dynamic environment, initially unknown, to maximize its cumulative expected reward signals [34].

Regarding Figure 3.2, DL is a subset of machine learning where there are artificial neural networks algorithms inspired by the human brain. Due to the ability to produce faster and more accurate results when the amount of data is increased, these algorithms have a certain advantage in some applications. There is some work that explores the use of DL algorithms which use various artificial neural network architectures to extract and process features into the data [13], [26].

3.4 Data Mining

Data mining (DM) is the process of discovering and extracting undiscovered information and patterns previously. DM is widely used in large databases. In this context, this

process makes it possible to obtain meaning from the information in order to generate values for making complex decisions [35], [36]. It is common to confuse terms such as ML and DM, due to overlapping data. In this way, DM has a different focus. The purpose of DM is not necessarily to induce global models of the system through the relationship between input and output variables. DM is more explanatory and used to discover patterns in a dataset that are generally descriptive rather than predictive. DM also places emphasis on analyzing very large datasets and therefore on aspects of scalability and efficiency [37]. Several other fields are closely related to ML and DM.

3.5 Machine Learning Techniques

ML techniques, such as DT and RF, are often used in the analysis of sensor data for forest fire detection and prevention. The goal is to develop machine learning models capable of identifying patterns and anomalies in the data collected by sensors. The use of ML techniques such as DT and RF can offer several advantages over traditional data analysis methods. With all these technological advances, several data analysis techniques have been developed. Thus, some of them stand out, they are presented in the following subsections [38], [39].

3.5.1 Decision Tree

Decision Tree (DT) method is a supervised ML algorithm that is widely used in ML problems such as geosciences. A DT model is developed through an algorithm that unfolds the space of input variables into subspaces using a hierarchical decision set [40]. As the name implies, the DL algorithm uses a tree format for decision making. This format allows viewing conditions and chances of the outputs of each node [41]. A DT structure is composed by nodes and branches. Each node will lead two branches that correspond to the different possible outputs. Figure 3.3 shows an example of a decision tree structure that classifies animals.

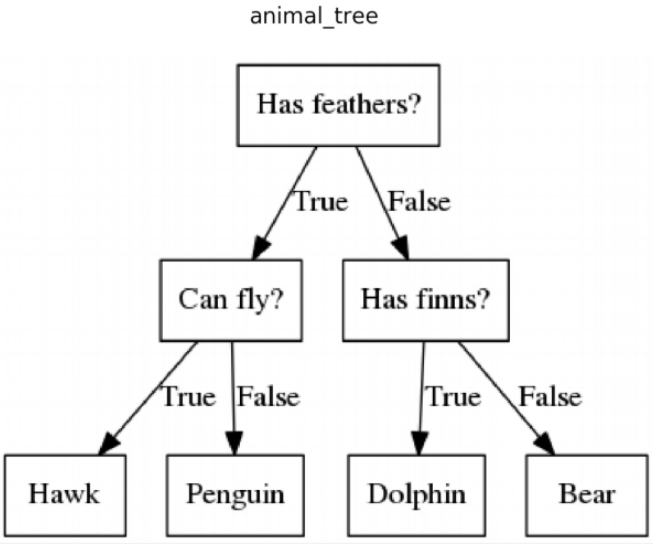


Figure 3.3: Example of a Decision Tree [42].

A DT starts with a single node, which branches into possible outputs and each one of those outputs leads to others nodes, which branch into other possibilities. This type of method can be used for both regression and classification problems.

- Regression:

  In this case, training the algorithm requires continuous outputs, thus the DT predicts continuous values instead of classes.

- Classification:

  In this case, the DT is trained on output data that is categorized into classes and can predict classes for unseen data.

According to [41], generating a DT requires a lot of computational effort. As long as more divisions are performed, the data becomes more and more split. With this in mind, a major concern in DT algorithms is efficiency and scalability when applied to large databases. Most decision tree algorithms have the constraint that training tuples must reside in the main memory. Therefore, this constraint limits the scalability of such algorithms [35].

At each iteration or node, to decide which feature leads to the purest branch, we need to measure the purity of a dataset. The most famous metrics for measuring purity are information gain, entropy, gini index, and gain ratio [35], [41], [43].

Entropy

Entropy is an information gain through impurity characterization of the data. Therefore, It is a measure that indicates the inhomogeneity of the data.

With this indicator, it is possible to verify that the less pure the data, the more confusing it is. However, the more impure, the more diversification of information.

Information Gain

Information gain is a metric to check how good a feature is for splitting. For this calculation, it is necessary to calculate the entropy before the separation and after the separation of the data, through the sum of the output entropies weighted by the size of the subsets [41], [44].

Gini Index

Another measurement for base purity is the Gini Index. The Gini index, is then defined as the weighted sum of the Gini impurity of the different subsets [43], [45]. The feature with the lowest gini index is used as the next feature to be split.

3.5.2 Random Forest

According to [46], the (RF) is initially proposed by Breiman L et al [47], this method can be considered simply as multiple decision trees. Figure 3.4 shows that RF is a set of decision trees.

The random forests are able to handle supervised classification and regression tasks [48]. According to [47], [49], the accuracy of a random forest depends on the strength of the individuals tree classifiers and a measure of the dependence between them.

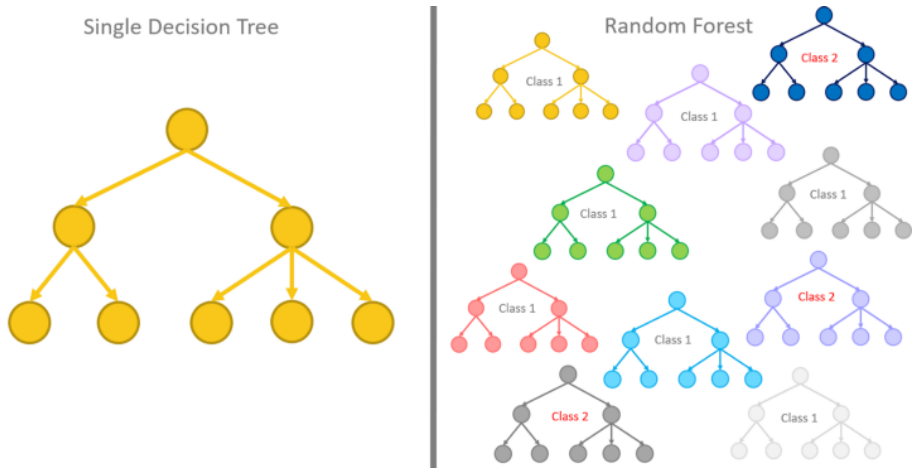


Figure 3.4: Example of a Random Forest [43].

3.5.3 K-Means

The technique known as k-means is a clustering technique. This technique was proposed by MacQueen, 1967, [50]. The clustering technique aims to cluster and divide. This algorithm initially considers all the individual objects in the system, and as the code progresses, groups them and divides them into larger groups, k groups. This value of k is one of the objectives of optimizing the code, because using some existing techniques we can find the ideal number of groups, k. [51], [52]. Figure 3.5 shows an example of a k-means process.

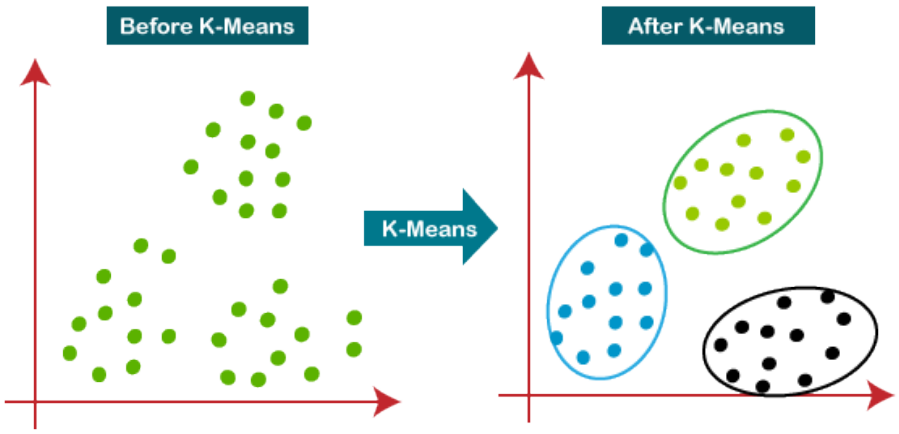


Figure 3.5: Example of a kmeans [53].

The K-means algorithm randomly determines the initial cluster centers. After that, the remaining data points are assigned to centers according to some predefined minimum distance to center criterion. According to [52], it's the Euclidean distance of the object to the center of the cluster. In Python Code, there is a library named `scikit-learn` that has several types of clustered techniques, including k.means, `sklearn.cluster.KMeans`. As mentioned before, there are some techniques for identifying the optimal number of groups, k groups, the Elbow Method and the Silhouette Index.

Elbow Method

The Elbow method consists of a plot of the values of cluster and Within-Cluster Sum of Square (WCSS) numbers, thus choosing the elbow of the graph is possible to obtain the optimal cluster numbers for the system [54]. This value of WCSS is the sum of the squares of the distance between each data point and the cluster centroid, multiplied by the number of clusters. [55]. As the number of clusters increases, the WCSS value will start to decrease [56]. Figure 3.6 shows an example of an elbow method in a plot view.

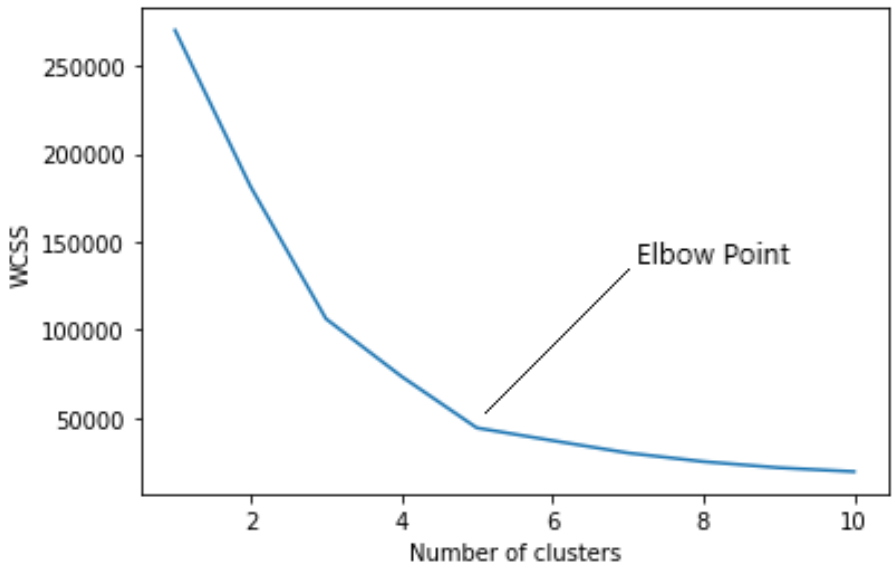


Figure 3.6: Example of an elbow method. [57]

This method can present problems when situations of having no elbow or having more

than one elbow occur [56]. In such cases, the solution is to use other cluster evaluation techniques, such as the Silhouette index.

Silhouette Index

Another method for identifying the optimal number of k is the silhouette method. This other method of comparing distance within a cluster says that the greater the difference in distance, the better the fit. This index, width of the silhouette, was introduced by Kaufman and Rousseeuw (1990) relating the difference between proximity within the cluster and separation from the rest [56], [58].

This value has a range between -1 and 1. If the silhouette value is close to 0, it means that this data can be considered in another centroid. In this sense, the system did not find the best. If the value is close to -1, it means that it does not belong to this cluster. Finally, if the silhouette value is close to 1, the data is well classified.

3.5.4 Support Vector Machine

A SVM is a supervised learning algorithm that can be used for classification and regression tasks. The basic idea behind SVM is to find a hyperplane in a high-dimensional feature space that maximizes the separation of different classes. In other words, it tries to find the best boundary (or hyperplane) that separates data points of different classes [59], [60].

SVM is especially useful when the data is feature-rich and the classes are not linearly separable. The algorithm works by transforming the input data into a higher dimensional feature space where linear boundaries can be found to separate the classes. It does this through a process called the kernel trick, which maps input data into a higher-dimensional space without actually computing the coordinates of the data in that space [59], [60].

3.5.5 Gradient Boosting

Gradient Boosting is an ensemble learning technique that combines multiple models to form a stronger model. This is an iterative process where new base learners are added to the ensemble to improve the overall performance of the model. The basic idea behind gradient boosting is to train a sequence of base models, where each model is trained to correct the errors of previous models. The base learner is usually a decision tree, but other types of models can also be used [61], [62].

During each iteration, the gradient boosting algorithm tries to minimize a loss function, which measures the difference between the predicted value and the true value, by adjusting the predictions of the current set of base learners. This is done by computing the gradient of the loss function with respect to the predictions of the current ensemble, and then training a new base learner to correct for residual negative gradients [61], [62].

3.5.6 Linear Regression

Linear regression is a viable method to predict the burned area of forest fires using meteorological data [63]. Linear regression is a powerful statistical technique that can be used to understand the relationship between different variables in predicting forest fires. The method involves finding the best-fitting line or equation that describes the relationship between the burned area of forest fires, and meteorological variables such as temperature, humidity, and wind speed. It is based on the assumption that there is a linear relationship between these variables. In the past, studies have used linear regression to predict the burned area of forest fires with a high degree of accuracy [63], [64].

3.5.7 Metrics to Evaluate the Performance

There are several metrics that can be used to evaluate the performance of a regression model. The choice of metric will depend on the specific goals. These are some used metrics for evaluating regression models [65]–[68].

- Mean Absolute Error (MAE):

  MAE is the average difference between the predicted values and the true values. It is easy to understand and interpret, and is appropriate for datasets with a wide range of values.

- Mean Squared Error (MSE):

  MSE is the average of the squared differences between the predicted values and the true values. It is more sensitive to outliers than MAE.

- Root Mean Squared Error (RMSE):

  RMSE is the square root of the MSE, and is used to interpret the error values in the same units as the original data.

- Coefficient of Determination (R-Square):

  R-Square is the proportion of the variance in the dependent variable that is predictable from the independent variable. It is important to note that the R-Square can be negative if the regression model does not fit the data well, indicating that the predicted variance of the dependent variable cannot be explained by the available data or the model in question.

The best metric to use will depend on the characteristics of the data and the specific problem. It is often a good idea to use multiple metrics to evaluate the performance of a regression or classification model.

Chapter 4

Data Characterization

In this chapter, the data characterization is presented, the data were obtained from previous SAFe projects, specifically from the months of September and October 2020. In this context, modules were installed to obtain the data, the five attachment points were chosen based on the soil type and trees around the IPB campus.

4.1 Data Identification

The nomenclature of the modules was not the same as the files obtained from the database. Thus, the pair formed by modules 1 and 2 will be associated with trees near the building. Modules 3 and 4 are attached to the trunks of a small orchard, and modules 5 and 6 are located on medium-sized trees around the Fervenza River that runs through the campus. On the other hand, modules 7 and 8 are fixed off-campus with a high density of trees. And the last module 9 is set on a tree that is planted in a yard; that is, it does not share other elements around it.

Figure 4.1: Map with the sensor modules distribution [14].

For each module, there are flame sensors and a sensor that acquires the temperature and relative humidity of the air. These sensor modules are designed to be flexible in terms of sensor replacement or addition. With this in mind, to transmit the data to a database, the sensor modules use a LoRa Communication method because of the long-range and low consumption.

The data was stored in a database in a table with the following columns: Time, *sen1* (sensor1), *sen2* (sensor2), *sen3* (sensor3), *sen4* (sensor4), *sen5* (sensor5), *h* (humidity), *tem* (temperature).

In this work, the data is used in Excel format and it comes from the SAFe database. The work uses `Pandas 1.5.1`, which is a powerful library for data manipulation and analysis in Python. It provides useful data structures such as the dataframe, which is used to store and manipulate data in a tabular format, similar to a spreadsheet. In this code, Pandas is used to read the `.csv` files using the `read_csv()` function, which takes a file path as a parameter and returns a dataframe containing the data in the file.

As mentioned before, this work will deal with modules with nomenclature between 01

and 09. The code reads in 09 `.csv` files, specifically those with the file names `01.csv`, `02.csv`, etc. up to `09.csv` and assigns the resulting dataframe for each file to a variable with the same name as the file but replacing the `.csv` extension by module, for example, `01.csv` is assigned to `dados_module_01`.

4.2 Data Description

Using the Python code, all datasets of each sensor were imported for data analysis on the Jupyter platform. In the first part of the characterization, a statistical analysis of the data was obtained using the `describe()` python method. The description contains the following information for each column of the dataframe [69]:

- count - The number of not-empty values.
- mean - The average (mean) value.
- std - The standard deviation.
- 25% - The 25% percentile*.
- 50% - The 50% percentile*.
- 75% - The 75% percentile*.
- max - The maximum value.

*: how many of the values are less than the given percentile.

Table 4.1 shows an example of the data using the described method. The displayed data refer to the module 01.

Table 4.1: Describe of the module 1.

describe	h (%)	sen1 (10bits)	sen2 (10bits)	sen3 (10bits)	sen4 (10bits)	sen5 (10bits)	tem (°C)
count	42078	42078	42078	42078	42078	42078	42078
mean	71.628	567.371	581.06	566.52	603.46	598.19	16.13
std	22.19	477.270	472.69	477.75	464.43	468.03	6.10
min	10	26	0	26	0	0	5
25%	56	40	43	39	49	45	12
50%	82	950	968	950	993	992	15
75%	90	1023	1023	1023	1023	1023	19
max	93	1023	1023	1023	1023	1023	38

Statistical searches of all available modules and sensors were carried out. Thus, with these data, it was observed that not all sensors count the same amount of data, Figure 4.2 can show the difference between all the sensors 1.

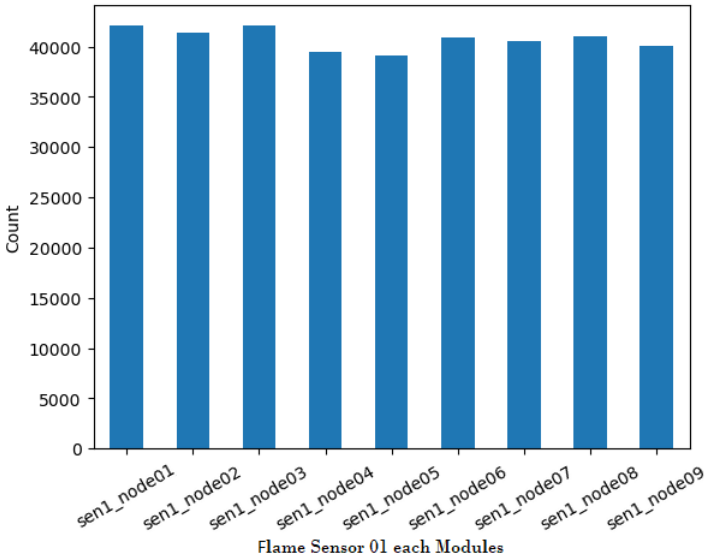


Figure 4.2: Count of the numbers of rows for similar sensors.

The reason for this difference in the data is due to a loss of communication that did exist in the period of the analyzed data. Other differences between the modules can be found when we analyze the data by sensor type.

4.3 Data Characterization According to Sensor Type

Table 4.2 shows that the average of the flame sensors of modules 01, 02, 03, 04 and 05 have values considerably lower than the values of the sensors of modules 06, 07, 08 and 09. This difference can be explained by the model of the sensors installed in each of the modules. Therefore, it is possible to separate the sensors into two groups of models, those that reaches minimum values close to zero and another group that reach minimum values in the range of 337 and 974.

Table 4.2: Describe of flame sensors

Module	*sen1 (10bits)*			*sen2 (10bits)*			*sen3 (10bits)*			*sen4 (10bits)*			*sen5 (10bits)*		
	mean	**min**	**max**	**mean**	**min**	**max**	**mean**	**min**	**max**	**mean**	**min**	**max**	**mean**	**min**	**max**
01	51.51	26.0	1023.0	59.30	21.0	907.0	50.75	26.0	1023.0	84.95	0.0	1023.0	75.17	0.0	956.0
02	59.11	29.0	932.0	44.84	12.0	879.0	44.36	16.0	870.0	51.44	26.0	910.0	48.17	21.0	895.0
03	52.98	18.0	929.0	48.17	25.0	1023.0	47.06	0.0	908.0	51.30	26.0	927.0	55.43	0.0	938.0
04	93.12	2.0	1023.0	88.22	25.0	1021.0	87.64	25.0	1022.0	132.13	0.0	1009.0	84.46	30.0	1023.0
05	126.11	32.0	1023.0	162.91	0.0	1023.0	154.53	34.0	1023.0	127.93	0.0	1021.0	107.11	24.0	1022.0
06	1011.70	670.0	1023.0	1007.98	428.0	1023.0	1006.78	337.0	1023.0	1018.01	974.0	1023.0	1003.21	843.0	1023.0
07	1005.51	793.0	1023.0	1010.55	616.0	1023.0	1008.95	445.0	1023.0	1017.60	970.0	1023.0	1012.18	976.0	1023.0
08	981.33	582.0	1023.0	980.77	512.0	1023.0	974.76	513.0	1023.0	999.80	901.0	1023.0	1002.31	944.0	1023.0
09	982.93	636.0	1023.0	992.61	842.0	1023.0	991.11	818.0	1023.0	1009.18	970.0	1023.0	996.48	940.0	1023.0

This variation behavior is observed only in the values of the flame sensors. Table 4.3 shows that the humidity and temperature values do not show such explicit differences in the maximum and minimum values.

Table 4.3: Characterization of temperature and humidity sensors

Module	*tem (°C)*			*h (%)*		
	mean	**min**	**max**	**mean**	**min**	**max**
01	19.95	7.0	38.0	59.21	10.0	93.0
02	21.03	7.0	39.0	55.47	11.0	95.0
03	20.51	7.0	33.0	63.51	27.0	95.0
04	19.99	7.0	35.0	64.98	20.0	94.0
05	18.54	7.0	30.0	69.07	28.0	94.0
06	17.73	7.0	34.0	64.15	19.0	93.0
07	17.23	6.0	30.0	65.73	19.0	93.0
08	18.13	7.0	30.0	63.55	23.0	93.0
09	17.88	0.0	31.0	66.32	20.0	93.0

Although module 09 has a value of 0 as the minimum, this behavior does not influence the mean values, as it was identified as a data outlier. This behavior can be seen in Figure

4.3. The justification for this could be a failure in the sensor scanning.

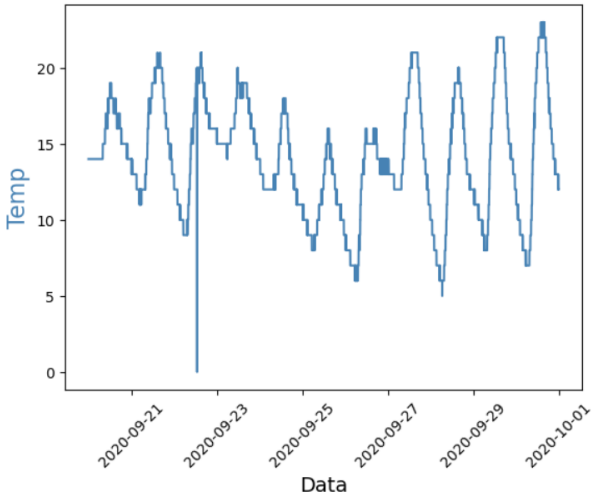


Figure 4.3: Module 09 Temperature Values.

When analyzing the data of each of the sensors, it was observed that despite the difference between the minimum values of the flame sensors, they present similar behaviors. Figure 4.4 shows the behavior of all five flame sensors of module 01. When analyzing the behavior, it is possible to notice that the data of the sensors present values close to or greater than 1000. During daytime the values displayed are less than 1000, this mode can be explained by the characteristics of the sensors, when the sensor was close to the flame, the values tended to be close to 0. On the other hand, when the sensor moved away from the flame, the values tended to approach 1023. Thus, sunlight influences the values of each of the sensors.

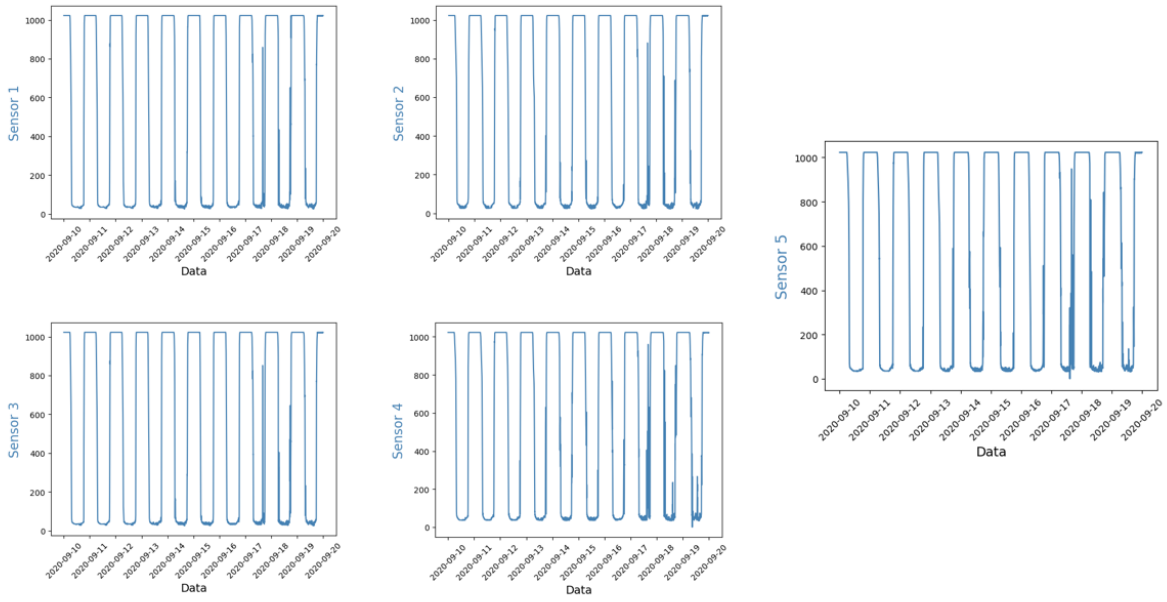


Figure 4.4: Module 01 Flame Sensors.

The temperature data from the sensors were analyzed individually and in groups. Thus, it was observed that the sensors present a decrease in values between the beginning of the available data and the last data of the dataframe, this is due to the beginning of autumn in the region. This behavior can be seen in Figure 4.5 which shows the daily average of all nine temperature sensors analyzed.

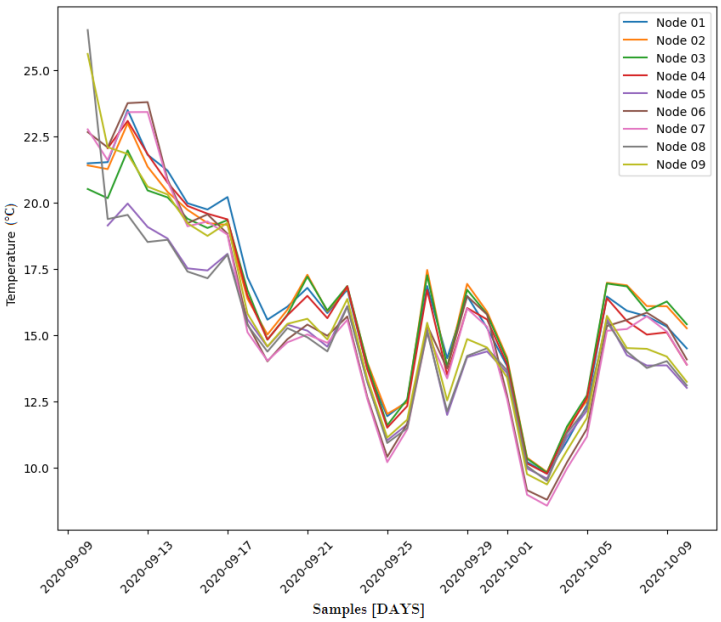


Figure 4.5: Average Temperature of Modules.

The last sensors to be analyzed were the humidity sensors. Similar behaviors were observed in these sensors as well. However, unlike temperature sensors, humidity sensors show an increase in values over time between September and October. The performance of the humidity sensors can be verified with the help of Figure 4.6, where it is possible to observe the increase in the daily average in all modules.

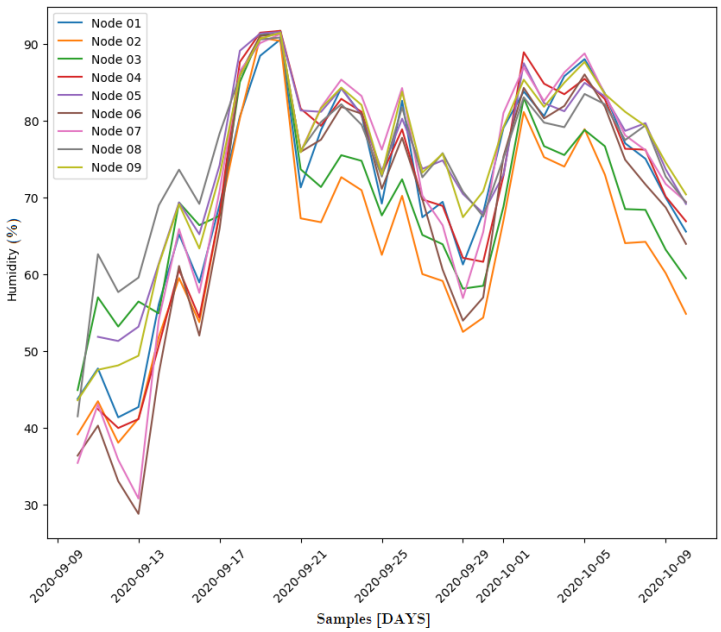


Figure 4.6: Average Humidity of Modules.

After all the data and its behavior were identified, another identification analysis was performed on the data columns. For this analysis, a code was developed that aims to find and report the percentage of values in each column that are considered outliers. This code is useful for pre-processing and data cleaning, as it allows the identification and potentially removal of outliers from the data that may distort the results of subsequent analyses.

The code iterates through each column in the data frame and computes the quartile 1 (Q1) and quartile 3 (Q3) of the values in the column and can then calculate the interquartile range (IQR) which is the difference between Q3 and Q1, Figure 4.7 shows the results each sensor. The selection of the factor 1.5 to define the limits of the IQR is based on common conventions in descriptive statistics. The recommendation to use a factor of 1.5 to identify outliers was proposed by John Tukey [70]. In this book, Tukey suggests that values that are above the Q3 plus 1.5 times the IQR, or below the Q1 minus 1.5 times the IQR, can be considered outliers, but also argues that points out that the choice of the exact value of the factor 1.5 is arbitrary and may depend on the context and the

goals of the analysis.

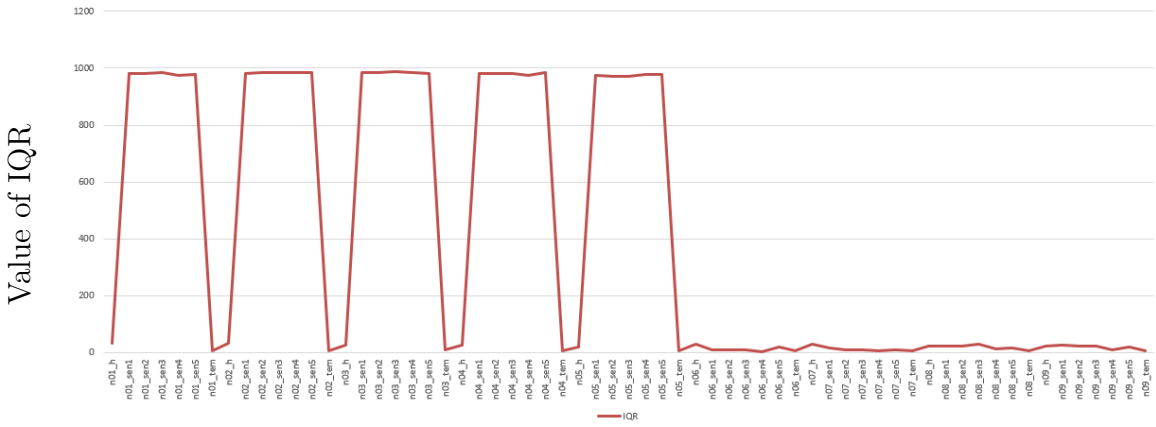


Figure 4.7: IQR for all sensors.

With these values it is possible to identify the lower and upper limits of the values, subtracting 1.5 times the IQR from Q1 and adding 1.5 times the IQR to Q3, respectively. In this way, the code performs the filtering of the values in the column that are between the lower and upper limits. Thus, outlier analysis was performed for factors of 1.5, 2 and 3 * IQR. Figure 4.8 shows the outlier values in percent for all the sensors analyzed. In it we can see a larger amount of outlier in module 8.

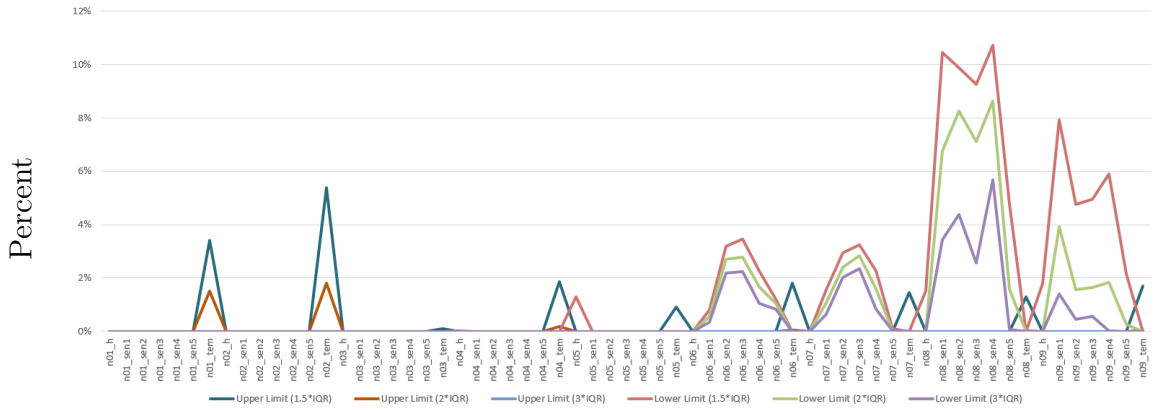


Figure 4.8: Outlier Percentage.

All this outlier treatment is useful for the next step of the project, the analysis of the data pattern identification models.

Chapter 5

Data Analysis and Results

In this chapter, the analysis of the data that was used in this on in this project will be presented, which were analyzed in some aspects. The first aspect analyzed was the question of the difference in available data between sensors and modules. This difference, which was explained in the previous chapter, was dealt with during the process of merging the databases of each of the modules.

5.1 Pre-Analysis of the Data

The purpose of this part is to maintain the pattern and trends shown in the data of a module after the data from all the modules are merged into a single dataset. Figure 5.1 shows what the pattern and trends of all flame sensors from module 01 has during the interval of September 10th to 12th, analyzing the data in the separate dataframe of the module.

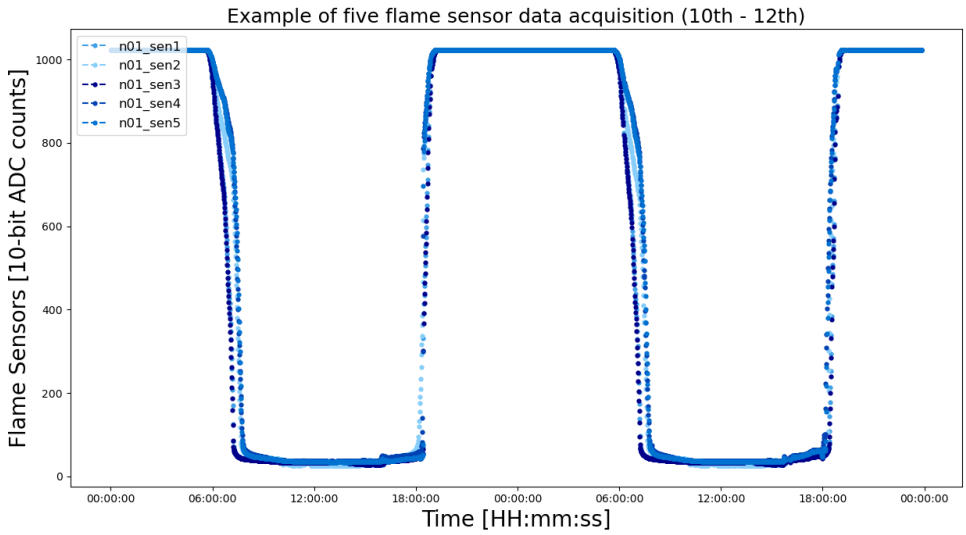


Figure 5.1: Flame sensors of module 01.

Considering the observed pattern, an analysis of the data was performed in relation to the time of each of the information. Thus, a time difference between the data was identified, as an example, the data of the *sen1*, from module 01, were sent on September 10, 2020 at midnight and the data of the *sen1*, from module 02, were sent on September 10, 2020 at midnight and forty-six seconds, a difference of 46 seconds. The figure 5.2 shows an example of the time difference between data from different modules, in this case, *sen1* from module 01 and module 02 from September 11th to 13th. In Figure 5.2 you can see times when we only have data from one module.

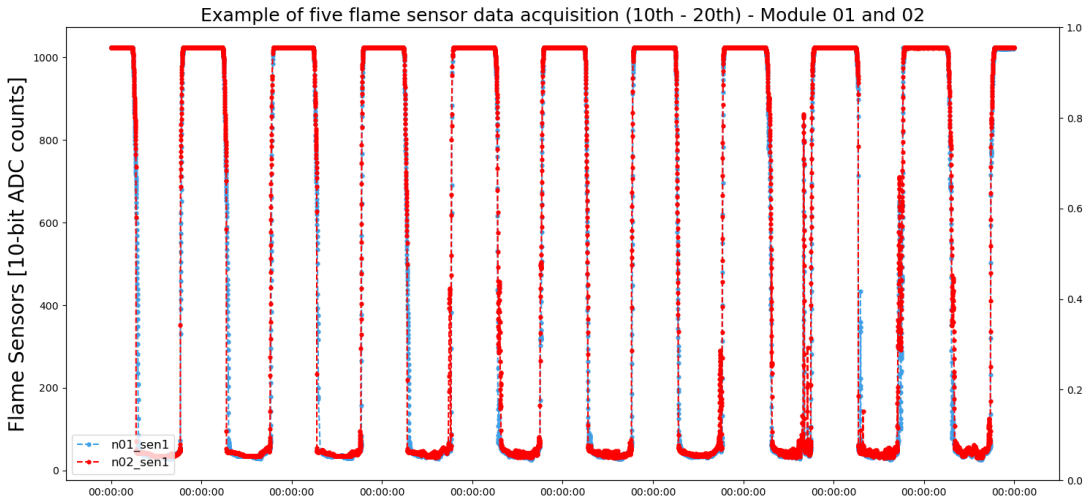


Figure 5.2: Time difference between data.

When concatenating the dataframes, there will be missing values in the resulting dataframe where the data from one sensor does not align with the data from another sensor. Regarding the handling of missing values, a possible solution could be drop the `NaN` values from the dataframe, as it will make the analysis of the data easier and more accurate. Since the sensors have different times. After combining the data, the creation of 79.92 % of `NaN` data was observed in each of the columns of the database. Therefore, it was verified that the values could be replaced by the averages of the previous values of the same column, as this small period would not influence the behavior of the sensors.

In addition to this analysis, it was thought what number of backward values would be used to calculate the average and add in the place of `null` values. In this analysis, we found that 3 backward values would be best for this data. For, the reduction of `null` values with an average of 3 values and 2 values obtained little difference in terms of percentage of reduction of `null` values. However, 3 values backwards reduced it to 1.17% and 2 values backwards reduced it to 1.32%.

Another way to check the credibility of the information after this substitution is to compare the proposed average with actual values. In Figure 5.3, we take a transition period between day and night from the *sen1* of module 1. With this data, an average of 2 to 8 units backward was performed on all the data, then we performed the difference

between the average value and the actual sensor value. We can see that the error increases as we have more values for the average.

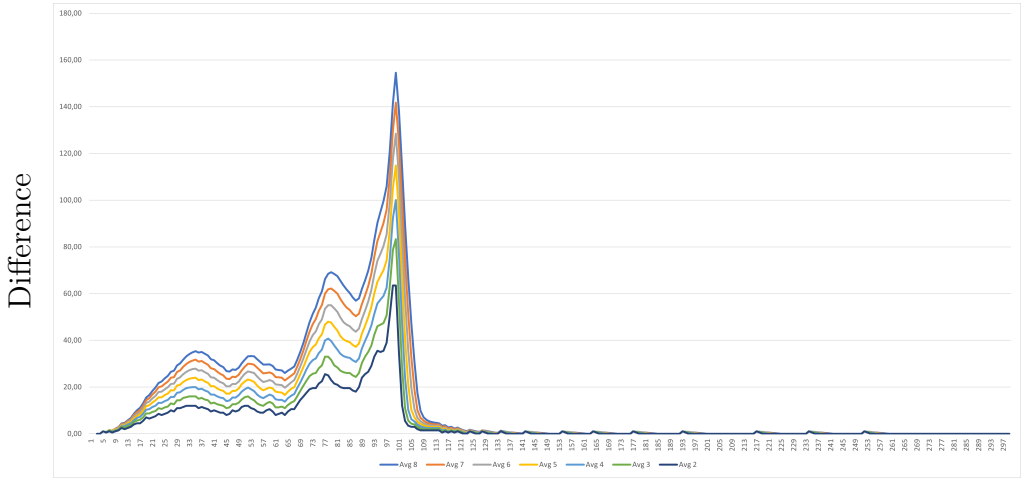


Figure 5.3: Difference Between Mean and Actual Value.

The next step was to use a while loop that iterates through all columns of the dataframe and replaces the `NaN` values of each column by the mean of the last 3 `non-NaN` values. The rolling() function was used to calculate the mean of the last 3 `non-NaN` values. It's important to note that this process of replacing missing values with the mean of the last 3 `non-NaN` values was done three times, to ensure that all missing values were replaced. Thus, there was a reduction in empty values from 79.92% to 1.17 %. These remaining values were taken from the analyzed database and the objective of this part was maintained, since the pattern and trends presented remained similar to the data of a module without performing the union with the data of other modules. Figure 5.4 shows how the pattern and trend were maintained even after the dataframes were merged.

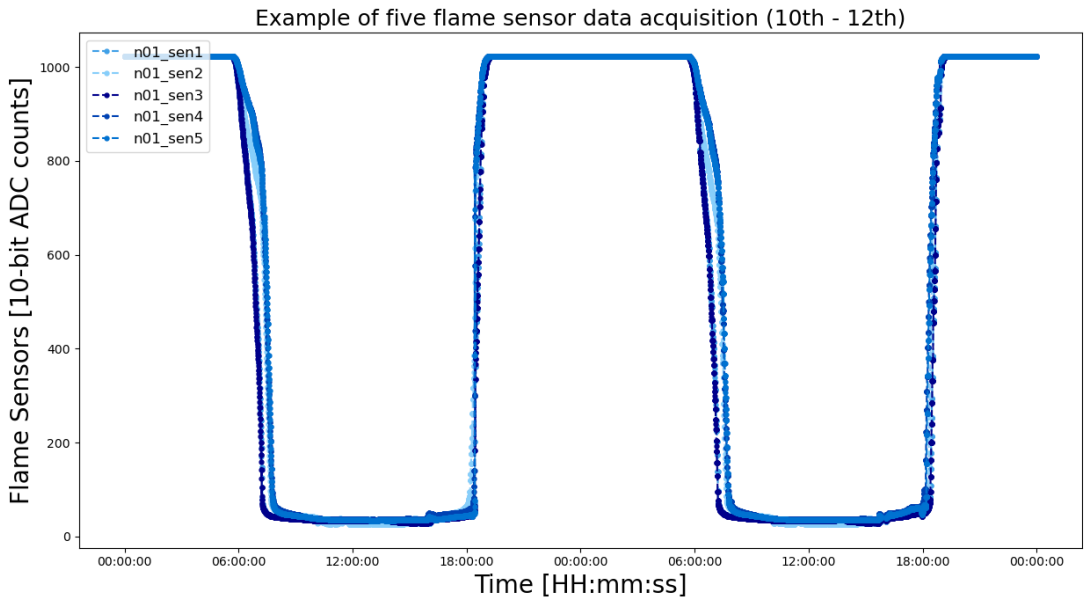


Figure 5.4: Flames sensors of module 01 after the merge.

5.2 Identification of Day and Night Transition Periods

The second aspect analyzed was the issue of separating the data into four factors: night, transition from night to day, day, and transition from day to night. For this analysis, the Python library `Astral v3.0` was used. All calculations performed through the library are based on astronomical algorithm equations by Jean Meeus [71]. In this context, the library returns the following data:

- Dawn - The time when sunlight begins to appear in the sky.
- Sunrise - The time in the morning when the top of the sun appears over the horizon, taking into account no impediments.
- Noon - The time when the sun is at its highest point over the observed point.
- Midnight - The time when the sun is at its lowest.

- Sunset - The time in the afternoon when the sun is close to disappearing below the horizon, taking into account no impediments.
- Dusk - The time in the afternoon when sunlight is fading into the sky.
- Twilight - The time between dawn and sunrise or between sunset and dusk.

To apply the library, some parameters are required, the main ones being latitude, longitude and timezone. Thus, for latitude and longitude, values of 41.8072 and -6.75919 were used, respectively, obtained with the help of Google Maps. It defines a location using the `LocationInfo()` function from the astral library. The timezone value was used Europe/Lisbon. Due to the library generating results taking into account no impediment of sunlight, it was found that transition periods between night and day, the times are not accurate, as the environment in which the modules were installed influences exposure to sunlight due to shading generated for example by trees, buildings among others.

To execute the code was created a `for` loop that iterates through all rows of the dataframe. For each row, it was used the `sun()`, `golden_hour()`, `blue_hour()`, and `twilight()` functions from the astral library to calculate the sunrise, sunset, golden hour start and end, blue hour start and end, and the twilight start, then assign the calculated values to new columns in the dataframe, such as 'dawn', 'sunrise', 'sunset', 'dusk', 'noon', 'Golden Hour Start', 'Golden Hour End', 'Blue Hour Start', 'Blue Hour End', and 'Twilight'.

It was developed a code to define the conditions to create a new column named *Status* in the dataframe. The column *Status* will contain either 'transition day to night','transition night to day', 'day' or 'night' depending on the value of the hours column in the same dataframe. The conditions used to determine the value of *Status* are based on the comparison of the hours column to other columns like 'duskless1', 'duskplus1', 'dawnless1' and 'dawnplus1'. These columns are previously defined in the code as the offsets from the dawn, sunrise, sunset, sunset columns. Then, it creates two new dataframes data_sensors_day and 'data_sensors_night by filtering the original dataframe with the condition *Status* == 'Day' or *Status* == 'Night' respectively.

It was used the `.loc` function to filter the rows in the dataframes based on the date range, from '2020-09-10' to '2020-09-12' for plot. The function then creates a plot with two y-axes and plots the sensor column of the dataframe on the first y-axis, and the *Status* column on the second y-axis.

Figure 5.5 shows a plot the data of a sensor, in this case, *sen1* of module 01, over a time period, and indicates the status of the data (day, night, or transition between day and night) on a separate y-axis, using different colors for the sensor data and the *Status* data.

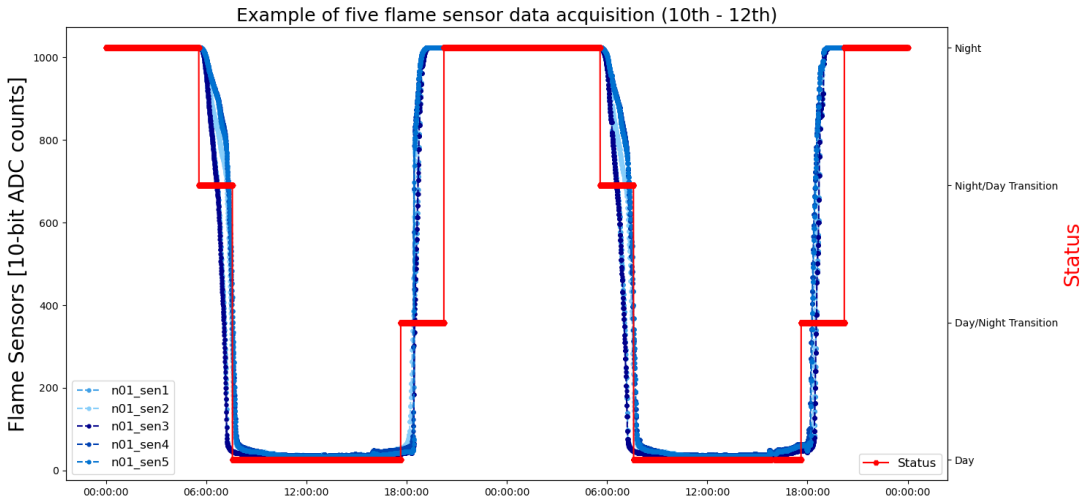


Figure 5.5: Status plot.

In this context, having all this analysis developed to identify the four periods, it was used a code to use forward and backward selection methods to select the most important features from the dataset for a given target variable, in this case *Status*. The forward selection method starts with an empty set of features, and then iteratively adds the feature with the lowest p-value to the set of best features until all remaining features have a p-value above a certain significance level. The backward elimination method starts with all features and iteratively removes the feature with the highest p-value until all remaining features have a p-value below a certain significance level. It is used to identify the most important features among all the features in the dataset to improve the performance of a predictive model.

5.3 Classification of Day/Night Transition Periods

Thus, a DT classifier was created to predict the *Status* column of the dataset. The data is split into a training set (80% of the data) and a test set (20% of the data). This ratio is commonly used as a rule of thumb in many ML application areas. The decision tree classifier is trained on the training set using two different criteria for splitting the tree: "gini" and "entropy". Then, the trained classifier is used to predict the *Status* column of the test set. The `max_depth` and `min_samples_leaf` are set to 10 and 5 respectively. The accuracy of the predictions is measured using the accuracy score metric and a classification report is generated for both "gini" and "entropy" criteria. Table 5.1 shows the result of this classification.

Table 5.1: Classification Report

Metric	**Class**	**Precision**	**Recall**	**F1-score**	**Support**
Gini	Day	0.99	0.99	0.99	31011
	Night	0.97	1.00	0.99	32993
	Transition Day/Night	0.98	0.72	0.83	3249
	Transition Night/Day	0.94	0.92	0.93	6001
	Accuracy			0.99	73254
	Macro avg	0.97	0.91	0.93	73254
	Weighted avg	0.98	0.98	0.98	73254
Entropy	Day	0.99	0.99	0.99	31011
	Night	0.97	1.00	0.99	32993
	Transition Day/Nigth	0.94	0.76	0.84	3249
	Transition Nigth/Day	0.94	0.92	0.93	6001
	Accuracy			0.98	73254
	Macro avg	0.96	0.92	0.94	73254
	Weighted avg	0.98	0.98	0.98	73254

For each class, the report provides precision, recall, and F1-score. Precision is the proportion of true positives among all predicted positives, while recall is the proportion of true positives among all actual positives. F1-score is the harmonic mean of precision and recall. The support column indicates the number of instances in each class.

The Gini and Entropy metrics produce similar results. For both metrics, the model achieves high precision and recall for the Day and Night classes, which have the largest

number of instances. The model also performs well for the Transition Night/Day class, which has a high F1-score. However, the model's performance is lower for the Transition Day/Night class, which has a lower recall and F1-score.

Overall, the model has high accuracy and achieves high precision and recall for the majority of classes, indicating good performance. Figure 5.6 shows the decision tree created for this classification problem.

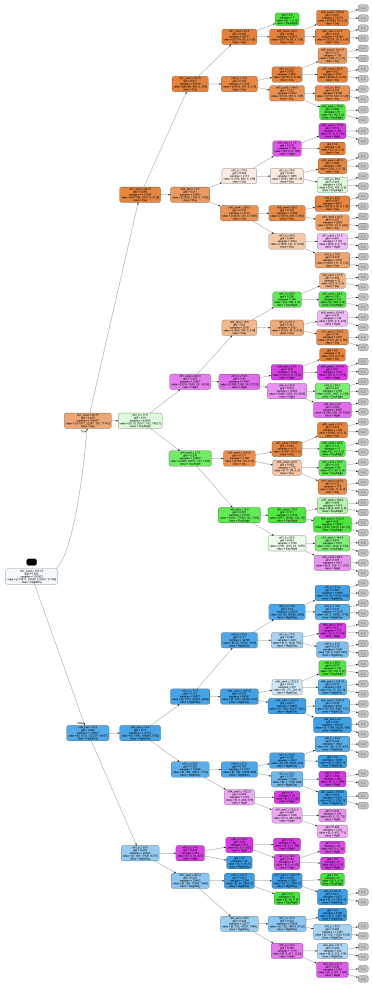


Figure 5.6: Decision Tree for Classification of Day/Night Transition Periods.

5.4 Modules Clustering

Another analysis developed was the identification of modules (dataframe) similar to each other. The code starts by importing the K-means class from the `sklearn.cluster` library. This class will be used to perform k-means clustering on the sensor data. The data for the sensor module is loaded and certain columns are removed. The next lines of code use list comprehension to create a list of columns for each sensor module (01, 02, etc.) that contain the string '01', '02' etc. in their name. This is used to select the columns for each module.

Then, for each sensor module, the data is standardized by subtracting the mean and dividing it by the standard deviation for each column. For example, the 5.1 shows how this standardization was done.

$$data = \frac{data - data.mean()}{data.std()} \tag{5.1}$$

This is done for all the sensor modules (01, 02, 03, etc.). Standardizing the data before running k-means clustering is important because the algorithm is sensitive to the scale of the variables. The k-means algorithm calculates the distance between points, and if the variables are on different scales, one variable will dominate the distance calculation. By standardizing the data, each variable is transformed to have a mean of 0 and a standard deviation of 1, so that all variables are on the same scale.

Standardizing the data also helps to avoid the bias that might happen if one variable has a large scale compared to others. This way, the algorithm will consider all features to be on the same footing, and it would make the results more reliable.

A K-means model is then created with 2 clusters and a random state of 0. This value was set with the help of a code to identify the value of k using the elbow method. The random state is used to initialize the algorithm in a reproducible way. The data for each sensor module is then fit to the K-means model using the `kmeans.fit()` and similar lines of code. This means that the K-means algorithm will cluster the data for each sensor module separately.

The cluster labels for each module's data are then obtained by accessing the labels attribute of the K-means model. The Adjusted Rand Index (ARI) is a measure of similarity between two clusterings and ranges from -1 to 1, where 1 indicates perfect similarity and -1 indicates no similarity. The ARI was converted to a percentage for ease of interpretation. This methodology allows us to compare the degree of similarity between the cluster labels produced by the K-means algorithm for each pair of sensor modules.

That is, the goal of this code is to compare the similarity between the cluster labels of the current module and all the other modules. This helps to understand how the different modules are related to each other.

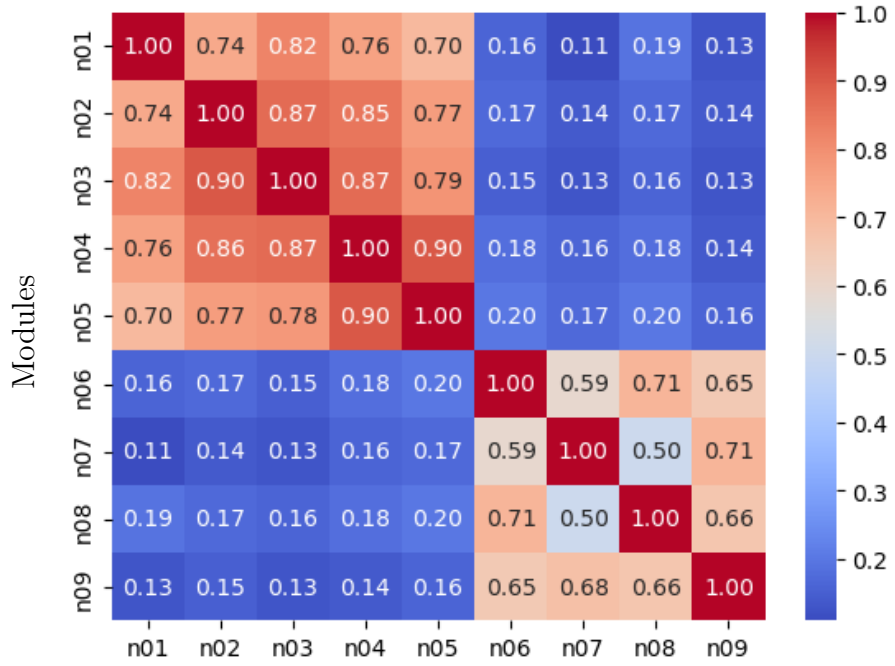


Figure 5.7: Heat Map Clusters.

With the similarity results between the clusters, a similarity matrix was created and through it a heat map was developed. Figure 5.7 shows the results of `adjusted_random_index()`. The heatmap displays the similarity values between each of the modules. From the results, it seems that the sensor modules are mostly similar to each other, but the similarity is lower when compared to modules 06, 07, 08, and 09. Thus, it is possible to see that there are two larger groups: modules 01 to 05 and modules 06 to 09.

5.5 Regression Models per Module

In this section, the analysis developed is focused on train and evaluate regression models on each module separately. The script uses training and testing sets to evaluate the models on unknown data and calculates evaluation metrics such as R-Square, MAE, MSE and RMSE. The code uses K-fold cross-validation to evaluate the performance of the three models (Random Forest Regressor (RFR), Support Vector Regressor (SVR), and Gradient Boosting Regressor (GBR)). The analysis also explains that the R-Square and MAE are the primary metrics to check the performance and that models with the highest R-Square and lowest MAE are preferred. The analysis also mentions the importance of RMSE and MSE as measures of the accuracy of the predictions.

In these first analyses, regression models were created taking into account each sensor of the module as a target and the other information from the same module as features of the model. The script was run nine times, for each module separately. Initially, a function code was created to analyze all modules in the same way. The function for this accepts two arguments: a dataframe (named df) and a column (named target).

The code is using K-fold cross-validation to evaluate the performance of the three models (RFR, SVR and GBR) on the dataset. K-fold cross-validation is a technique used to assess the performance of a ML model on unseen data. It involves splitting the dataset into k subsets (folds) of roughly equal size, where k is a user-specified parameter, in this case k equal to 3. This process is repeated k times, where each time one of the k subsets is used as the test set, and the remaining k-1 subsets are used as the training set. This allows the model to be trained and evaluated k times, providing an estimate of the model's performance on unseen data.

The function then creates three different regression models: a RFR, a SVR, and a GBR. The models are fit to the training data and used to make predictions on the test data. The function then calculates the MAE, R-Square score, RMSE and MSE for each of the models using the "mean_absolute_error" and "R-Square_score" python functions. Then a code was created to run the function on each column of the dataframe and store

the results of all metrics, along with the sensor name and file name of the dataframe. After that, it concatenates all the dataframes returned by the function into one.

After that, the code is grouping dataframe rows by the 'sensor' column, and then takes the mean of the values in each group. The result is a new dataframe where each row corresponds to a unique sensor and the columns are the mean of MAE, R-Square score, RMSE and MSE across all the different modules that the sensor appeared in. The purpose of this code is to aggregate the performance results of the different models for each target analyzed. This can help to identify which sensors are the most accurate or have the least error across all the different modules and can help make decisions about which sensors to use for further analysis or which sensors need more attention or improvement.

Table 5.2 shows an example of the result of the performance of the three models (RFR, SVR and GBR) on a specific sensor (*sen1* module 01) using k-fold cross-validation. The performance of the models is evaluated using the MAE, R-Square score, RMSE and MSE.

The RFR model has the smallest MAE and the highest R-Square score, indicating that it has the best performance among the three models on this sensor. The SVR model has a larger MAE and lower R-Square score value, indicating that it has worse performance than the RFR model. The GBR model has a larger MAE and similar R-Square score value as the RFR model, indicating that it has a performance similar to the RFR model. The R-Square score value is close to 1 which is considered a good fit.

Table 5.2: Results of regression for sensor 1 module 1.

RFR				**SVM**				**GBR**			
MAE	R-Square	MSE	RMSE	MAE	R-Square	MSE	RMSE	MAE	R-Square	MSE	RMSE
1.11	1.00	15.94	3.99	14.99	0.99	1564.80	39.56	1.13	1.00	14.18	3.77
1.09	1.00	25.74	5.07	14.07	0.99	2036.17	45.12	1.09	1.00	26.48	5.15
1.02	1.00	25.39	5.04	15.51	0.99	2092.63	45.75	1.08	1.00	26.56	5,15

Another way to analyze the results, can be seen in the Table 5.3, which displays the results of the averages of the 3 folders executed for each sensor, in this case, the average values of module 1.

Table 5.3: Results of regression for module 1.

	RFR				SVM				GBR			
sensor	**MAE**	**R-Square**	**MSE**	**RMSE**	**MAE**	**R-Square**	**MSE**	**RMSE**	**MAE**	**R-Square**	**MSE**	**RMSE**
h	9.70	0.53	206.49	14.15	16.53	-0.07	488.32	21.32	9.84	0.57	195.73	13.56
sen1	1.07	1.00	22.35	4.70	14.86	0.99	1897.87	43.48	1.10	1.00	22.40	4.69
sen2	5.11	1.00	341.35	18.37	12.04	1.00	1067.37	32.66	6.57	1.00	398.13	19.92
sen3	0.98	1.00	19.89	4.39	15.74	0.99	2197.69	46.81	1.37	1.00	21.52	4.60
sen4	3.60	1.00	180.28	13.32	19.48	0.99	3120.56	55.86	4.47	1.00	197.03	13.81
sen5	3.39	1.00	223.68	14.71	15.82	0.99	2040.85	45.17	4.68	1.00	280.22	16.47
tem	2.62	0.56	13.13	3.58	3.88	0.17	26.89	4.95	2.18	0.71	9.19	2.95

When analyzing this result, R-Square and MAE are the two primary metrics on which good performance should be checked. R-Square is a measure of how well the model predicts, with 1 indicating a perfect fit and 0 indicating that the model is not the best.

In general, models with the highest R-Square scores and the lowest MAE, MSE and RMSE scores were checked, as these indicate the best overall performance. The Table 5.4 shows the best models according to the values in the previous table.

Table 5.4: Analysis of Models for Module 1.

Sensor	**MAE**	**R-Square**	**MSE**	**RMSE**
h	RFR	GBR	GBR	GBR
sen1	RFR	RFR	RFR	GBR
sen2	RFR	RFR	RFR	RFR
sen3	RFR	RFR	RFR	RFR
sen4	RFR	GBR	RFR	RFR
sen5	RFR	GBR	RFR	RFR
tem	GBR	GBR	GBR	GBR

Analyzing the table we can see that for flame sensors, in this case, the RFR models are more accurate, for the rest of the sensors the GBR models are more accurate. However, if we look at the whole picture, and if we had to choose one model for the module itself, the GBR model would be the most accurate. This is because the values are close for flame sensors, that is, both GBR and RFR work for these sensors. It is worth mentioning that the results are based on a specific dataset and the performance of these models may vary when applied to different datasets or different sensors.

Figure 5.8 shows a comparison of the R-Square values of all targets analyzed. With these regression models for each individual module, we can analyze that the GBR model behaves better with this metric analyzed, especially when we analyze the drops in R-Square values, we realize that the lowest drops in performance are in GBR models.

Figure 5.8: Results R-Square per Module Regression.

The `Scikit-learn` python library allows negative R-Square values, this indicates that the model is worse than a model that simply uses the average of the target variable values as the prediction for each observation. This can happen when the model is too complex for the data set in question, or when the explanatory variables do not have a linear effect on the target variable. Figure 5.9 shows a comparison of the MAE values of all targets analyzed. If we analyze that the MAE values must be less to be better. At flame sensor points we can see that RFR shows better values than GBR.

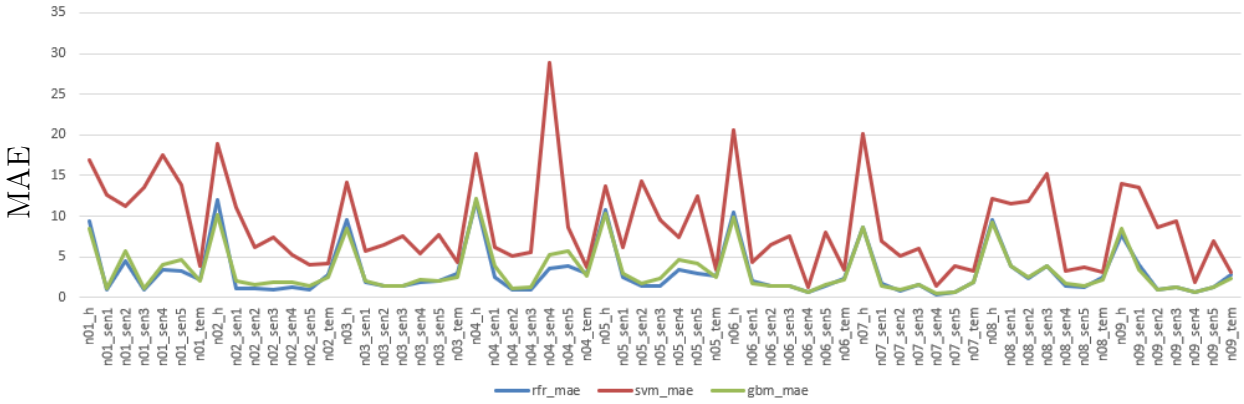


Figure 5.9: Results MAE per Module Regression.

In the analysis of the remaining metrics, the objective is to obtain the smallest value similar to the MAE. Thus, if we analyze Figure 5.10, we can see that GBR worked better for the MSE.

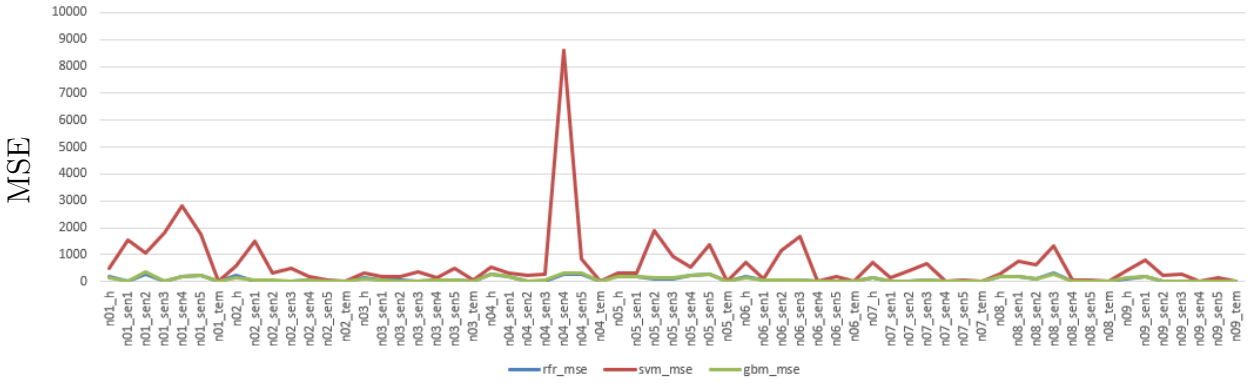


Figure 5.10: Results MSE per Module Regression.

If we analyze Figure 5.11, we can see the behavior similar to the metric MSE. Therefore, we can consider GBR as the best model.

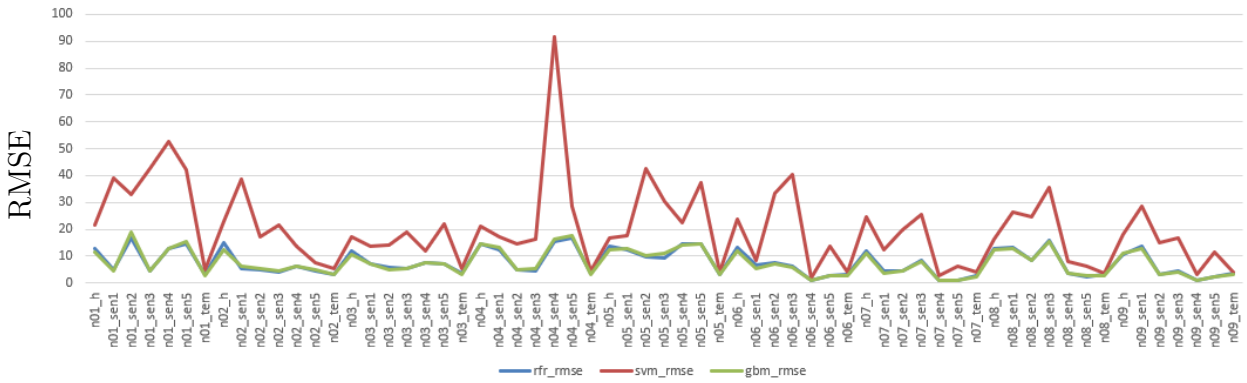


Figure 5.11: Results RMSE per Module Regression.

Based on these results we can see which GBR models work best for these sensors. However, we cannot rule out RFR models. Nevertheless, the SVM models performed much worse relative to the others.

After all these analyses, another analysis of the models behavior was performed only considering data during the transition period between day and night. For this purpose, it uses the prediction methods of the three models already developed to make predictions on a new data set. Then the number of predictions that have an absolute difference greater than a specified margin of error is calculated. The idea is to compare the actual values versus the values predicted by each model.

To check the errors, some error margin values were selected. The analysis was from 1 unit up to 8 units of margin of error between the actual value and the predicted value. In Figure 5.12 we can analyze the error margin of the RFR models. The error percentages tend to decrease as the error margins become larger. This is expected, since the larger the margin, the more likely it is that a prediction will have an error greater than that margin. Some sensors have higher error percentages than others, suggesting that the model may perform better on some sensors than others.

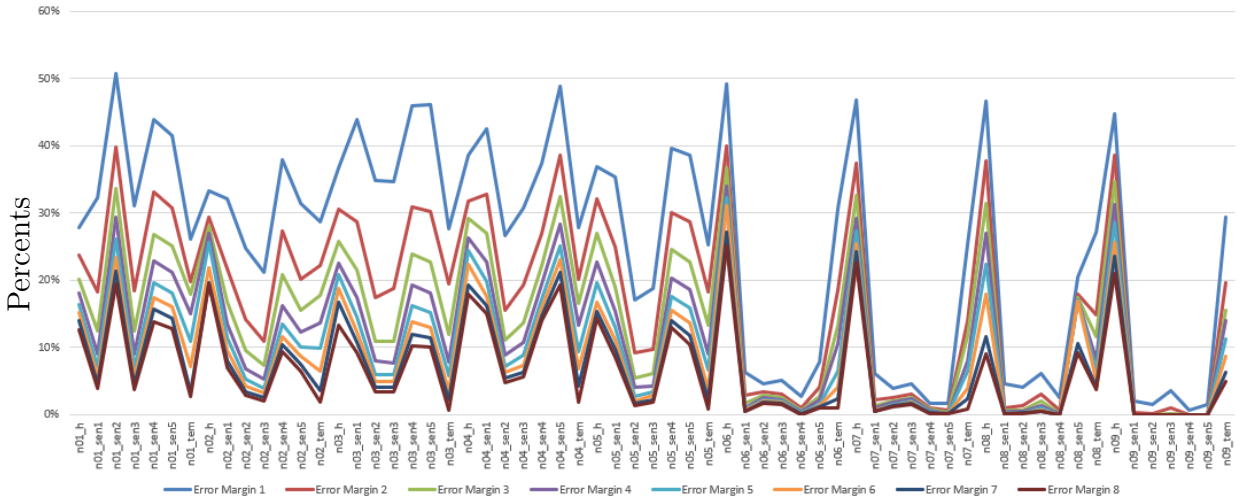


Figure 5.12: Percents of Margin of Error by RFR per module.

In general, the error percentages appear to be relatively high, even for bigger error margins. This may indicate that the model is not very accurate or that the problem is intrinsically difficult. This will be dealt with in the section Based on this, we can see that the flame sensors in module 06 and all temperature sensors, in general, have very low error margins, suggesting that these sensors are more accurate than the others. The pattern presented by analyzing the errors of the RFR model can be seen in the results of the SVM model. However, the error values are much larger than those presented in the previous model. Figure 5.13 shows this pattern for errors margin using the SVM model

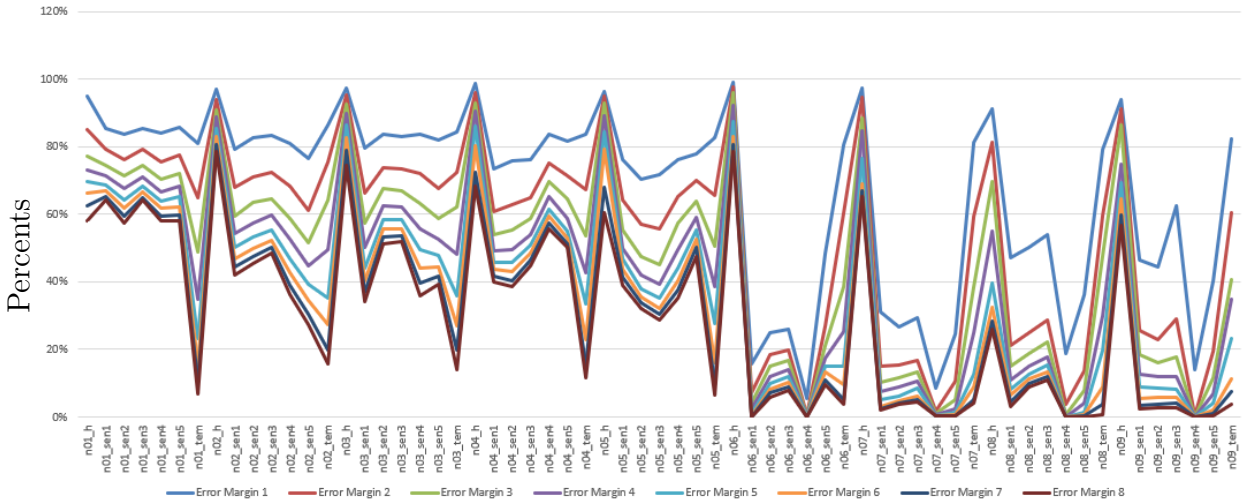


Figure 5.13: Percents of Margin of Error by SVM per module.

In contrast to the SVM model, Figure 5.13, the GBR model, Figure 5.14, presents better results, and it can be seen that in temperature sensors, where the errors are much better than the RFR model.

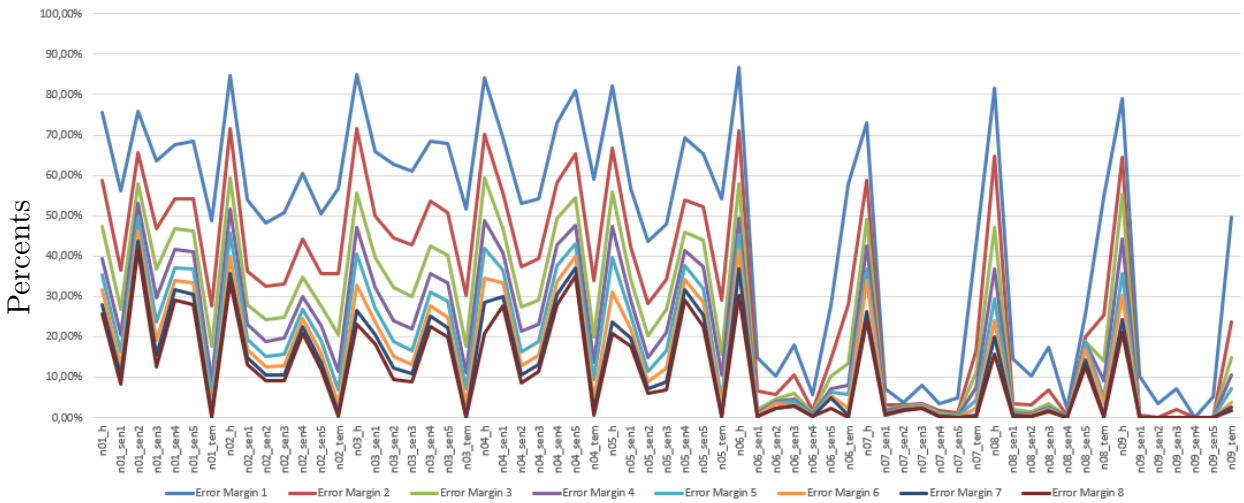


Figure 5.14: Percents of Margin of Error by GBR per module.

Therefore, if we average the 8 error margins for each model and compare them, as seen in Figure 5.15, we can see that the RFR model behaves better than the GBR model, this can be justified by the behavior of the model for lower error margins, where this model has better results.

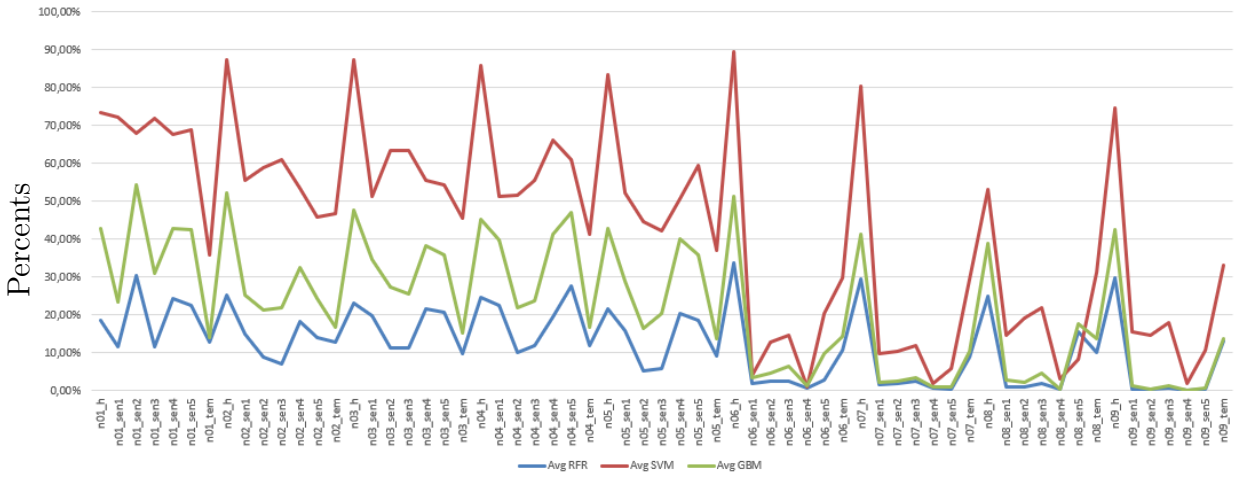
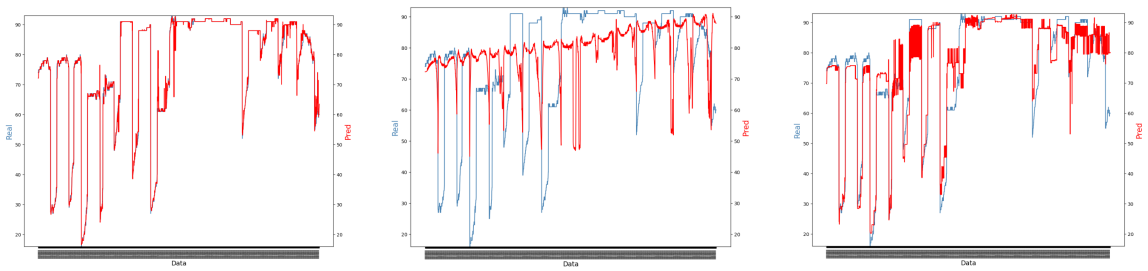


Figure 5.15: Percents of Error by target per module.

Based on Figure 5.15, it can be seen that the performance of the three models varies on different datasets. For example, the RFR model produces a relatively low error rate (less than 25%) on module 01 and module 02 datasets, but a high error rate on the flame *sen2* and *sen4* of module 01 datasets. Meanwhile, the SVR model generally has a high error rate across all datasets, with the exception of the flame *sen1* and *sen5* of module 06 datasets, where it produces a relatively low error rate. The GBR model performs relatively well on some datasets (such as *h* sensors and *tem* sensors from sensors of 1,2,3,4 and 5 modules), but has a higher error rate on others (such as *h* from sensors of 6,7,8 and 9 modules).

In general, it can be seen that the RFR model performs moderately well on all data sets. The SVR model, on the other hand, generally performs poorly. Finally, The GBR model performs better than the other two models, but still has a relatively high error rate on some datasets.

If we analyze Figure 5.16 that has an example of the predicted and actual values of the humidity sensor 01, we can really prove the better adherence of the RFR model. The visual analysis allowed us to prove that the errors and behavior of the RFR model were really better than the others.



(a) RFR Real x Pred (b) SVM Real x Pred (c) GBR Real x Pred

Figure 5.16: Visual analysis of the Actual x Pred values in transition periods for one module training

These analyses are of major importance since it was reported at the beginning of the work that the largest false alarm warnings were at periods of transitions between day and night.

5.6 Regression Models for One Module Using Transitions

In this part of the analysis, the same analysis created previously for each module was performed, however, in this new step of the analysis, it was performed tests of models for day data only and for night data only. In addition to these tests, the same analysis was also performed for each of the complete modules, taking into account the time of day as feature. These period features were developed using the analysis from Section 5.2 and added in a new column called *Status*. Thus, the basic idea of this section is to run the same algorithm that was run previously and in the end compare the metrics obtained for each model with the complete data and with separate data.

Figure 5.17 shows four scatter plots on it. In this example, the model comparison of RFR is shown for the regression models using all data per module (All), using day data only (Day), using night data only (Night) and *Status* column as another feature (Status). Then as we have 3 results for each sensor in each of the models. We used the groupby() function per module and sensor using mean() to get just the average of each metrics for

all targets by modules.

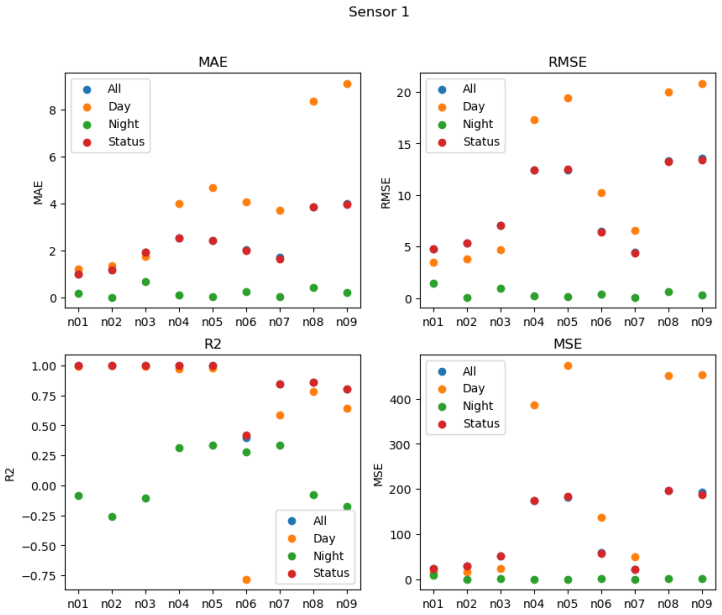


Figure 5.17: Model Comparison for Sensor 1 per Different Datasets.

Figure 5.18 shows the comparison of R-Square metrics for *sen1* with the model trained with all data without and with *Status* as a feature. In this example, it is possible to compare the values of this metric and verify possible improvements in the regression models. But it is still necessary to check the other metrics.

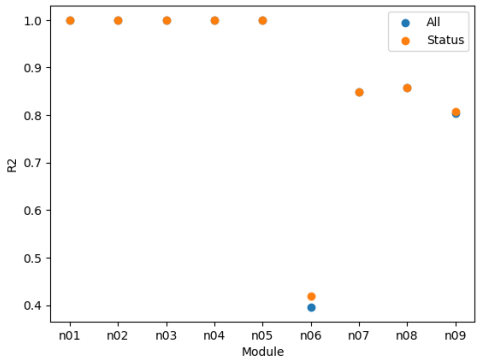


Figure 5.18: Model Comparison for Sensor 1 All and With Status.

Another analysis of results was with regression models trained with Day, Night, and Status columns as another feature, Figure 5.19 shows the R-Square results for day data.

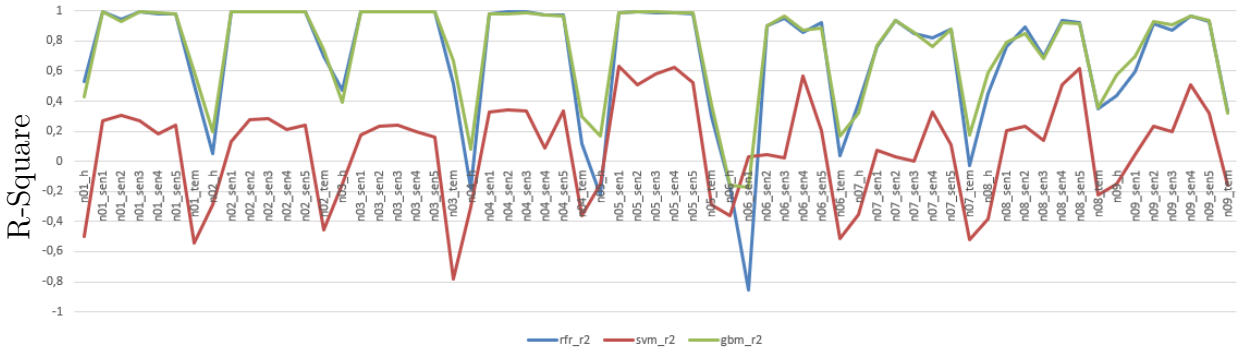


Figure 5.19: R-Square Results for Day Data Only

The GBR values performed better for models using Day data and the *Status* column as features. But for the *sen5* of module 6 and the *sen2* of module 07, it is noticeable that RFR was better for the R-Square metric. Figure 5.20 shows the result for the *Status* columns as feature.

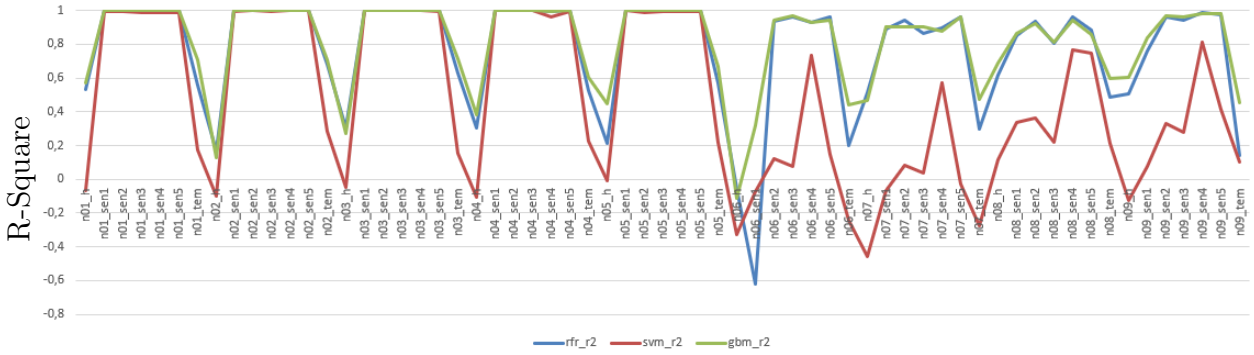


Figure 5.20: R-Square Results for Status as Feature

For night data, RFR models performed better, but with little difference from GBR models. But the outcomes are not good results.

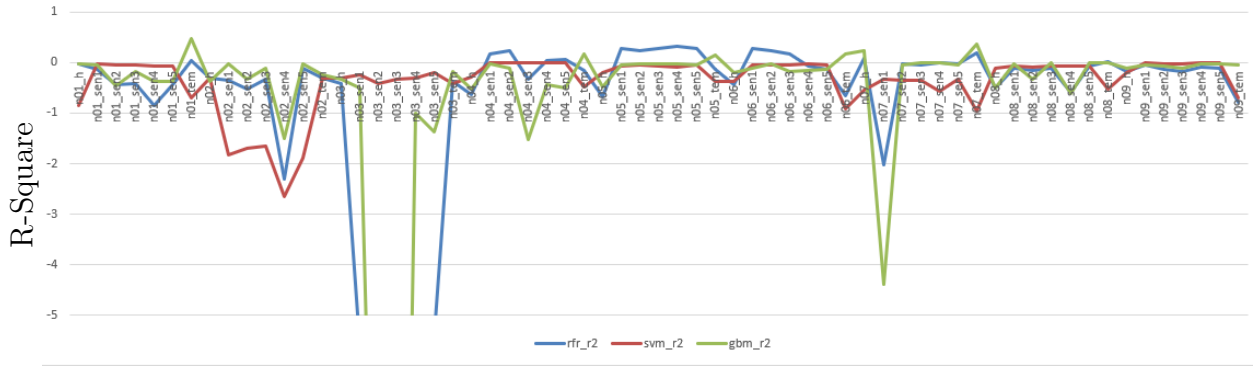


Figure 5.21: R-Square Results for Night Data Only

We can analyze that for these types of training the GBR models were better than RFR and SVR. This is because in general when RFR is better than GBR the difference is not significant. Now when we analyze the MAE values for the same types of training data.

Figure 5.22 shows the MAE values for day data, where we can see again that SVR models have very different and worse metrics than the other two models. Another point that calls attention is that in some moments the RFR model has better metrics than the GBR model, such as *sen2* of module 04 and module 08, in the remaining moments the GBR model is better, but with little difference.

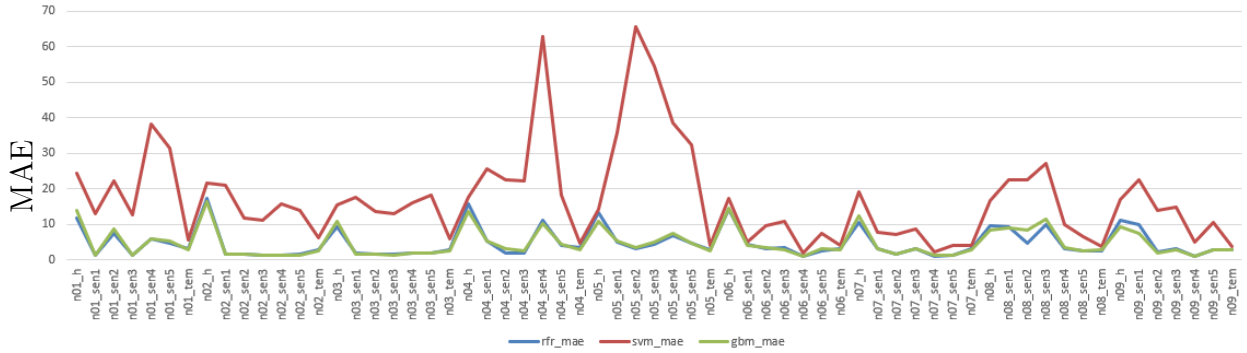


Figure 5.22: MAE Results for Day Data Only

Analyzing data that used the *Status* column as a feature, Figure 5.23, we can see the same behavior as the day data. Although, there are more points that the RFR model was better than the GBR model, such as *sen2* and *sen4* of module 01, *sen4* of module

03, *sen4* of module 04 and *sen2* of module 07.

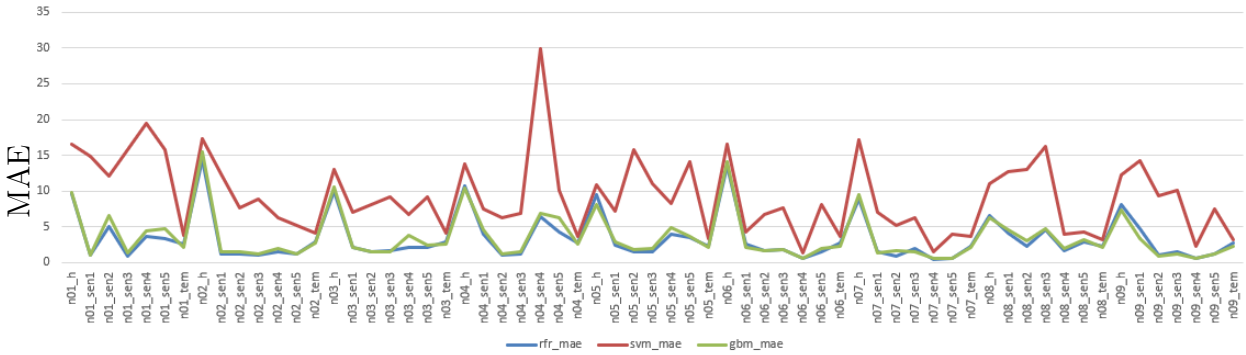


Figure 5.23: MAE Results for Status as Feature

Figure 5.24 displays the results of MAE for night data, we can see low value in the metric, the errors for SVR model showed to be better in several points. this suggests that the model is not explaining well the variation in the data and is also performing poorly.

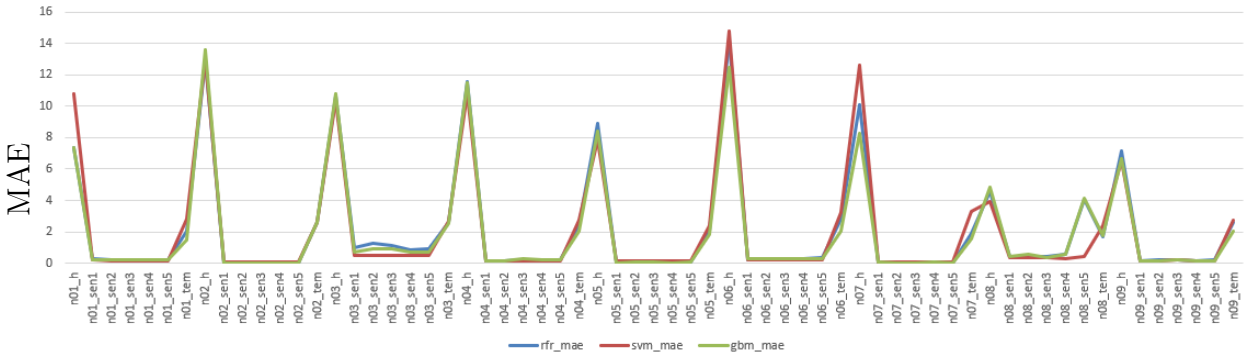


Figure 5.24: MAE Results for Night Data Only

In Figure 5.25 we can see the MSE results for day data, allowing us to once again verify the discrepancy of the SVR results with the other models. For day data, the GBR model was better, but for module 4 flame sensors, RFR behaved better.

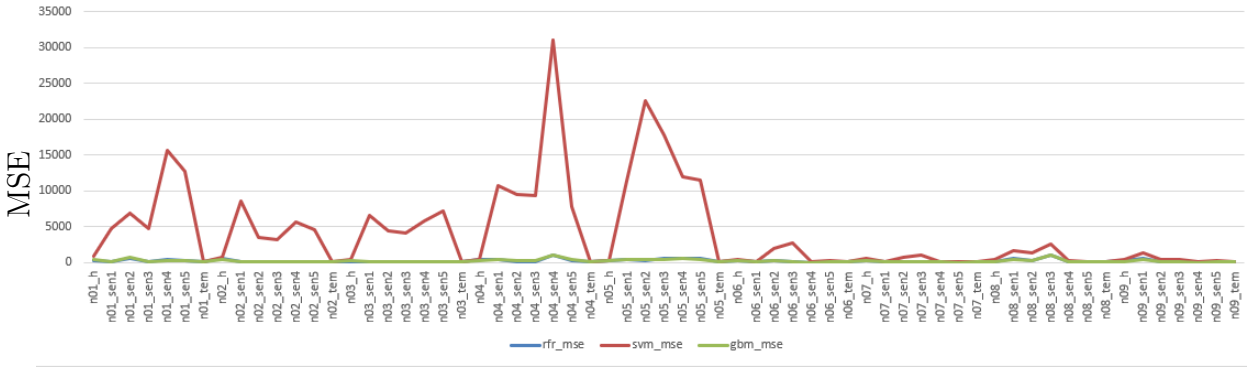


Figure 5.25: MSE Results for Day Data Only

In Figure 5.26, we can see similar behavior again for the day data. This figure displays the MSE results for data using the *Status* column as a feature. In this analysis GBR, was better at all the points analyzed.

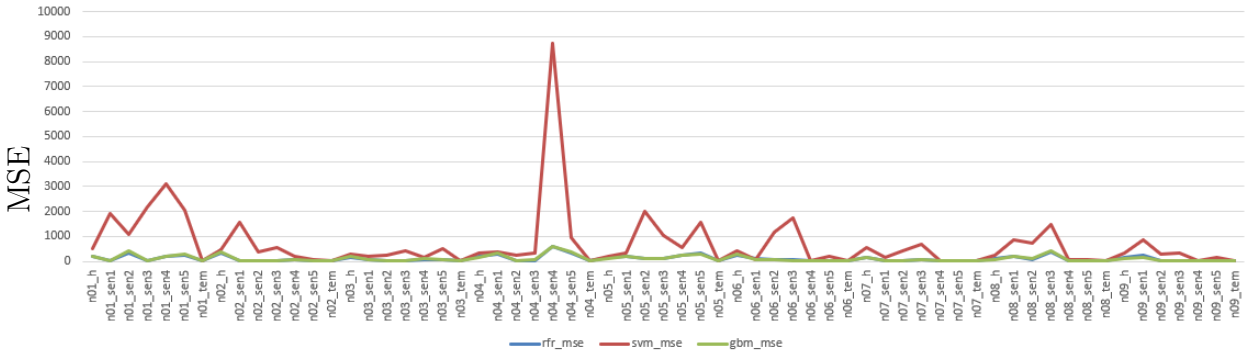


Figure 5.26: MSE Results for Status as Feature

In Figure 5.27, we can see the same behavior as for the night data, the training of these models for this data set proved to be not appropriate for the data in question.

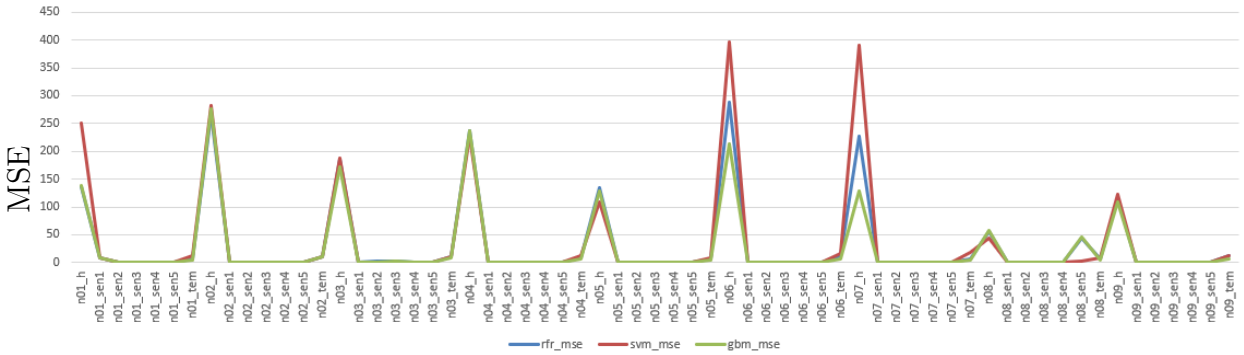


Figure 5.27: MSE Results for Night Data Only

In Figure 5.28 we can see the RMSE results for day data. For day data, the GBR model was performing better, but for module 4 flame sensors again, RFR performed better.

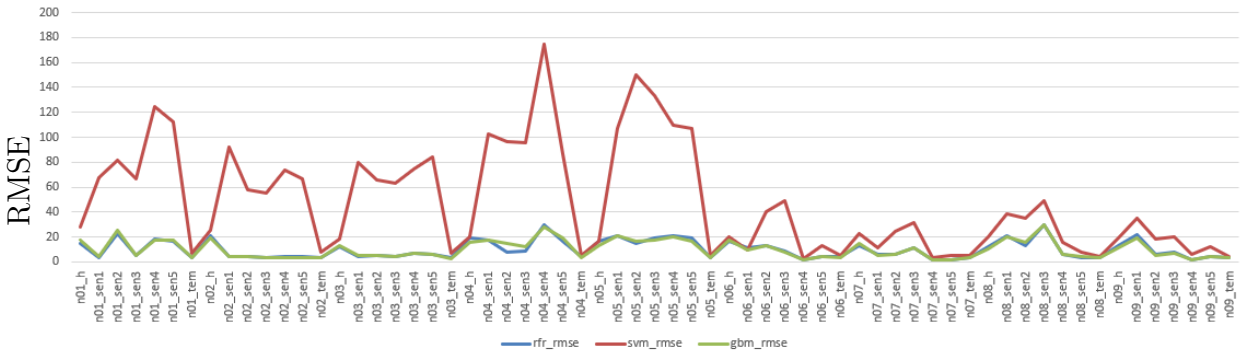


Figure 5.28: RMSE Results for Day Data Only

In Figure 5.29, we can see similar behavior again for the day data. This figure displays the MSE results for data using the *Status* column as a feature. In this analysis GBR, was better in all analyzed points, except *sen4* of module 3.

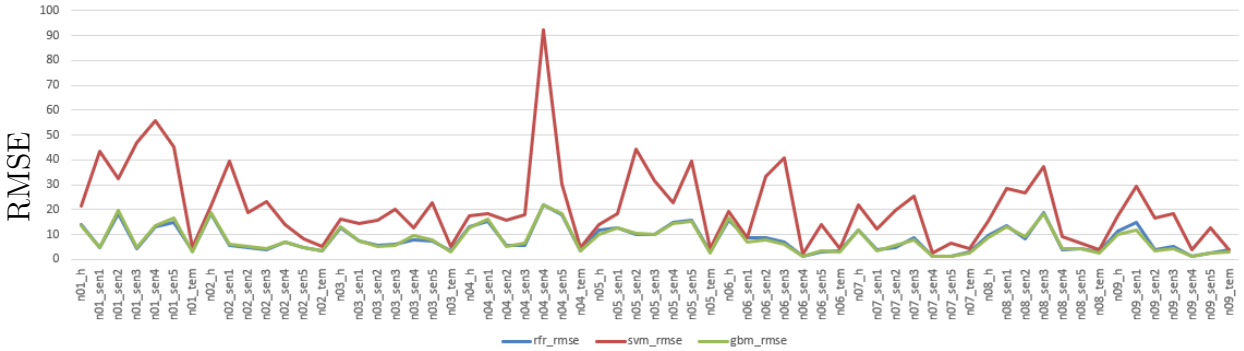


Figure 5.29: RMSE Results for Status as Feature

With the help of Figure 5.30, we can dismiss the results of the models for night data, as in all metrics the performance was poor.

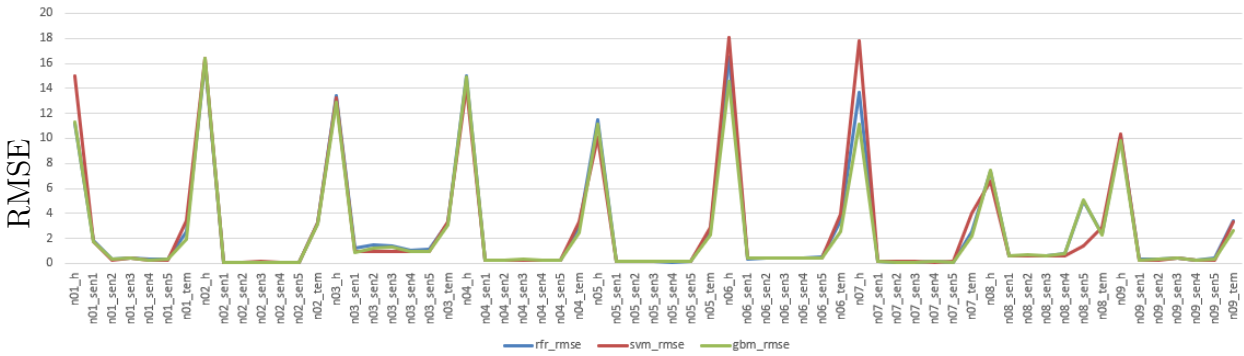


Figure 5.30: RMSE Results for Night Data Only

Another analysis that was performed was the averaging of metrics for comparison between models using data from a module, day data, night data, and data from a module with a *Status* column as an additional feature. In the first comparison of the metric R-Square we can verify that for night data we did not get a good result analyzing the average. Figure 5.31 shows this comparison. However, for the rest of the tests, GBR models presented themselves as better results for R-Square metric.

Figure 5.31: Average Results of R-Square.

The second comparison performed was the MAE metric, Figure 5.32. In this metric, we further verified that models using GBR and RFR presented very close with lower values. Thus, it was further verified that SVM did not generate adequate results. For modulo trained data with and without column *Status*, on average, the RFR models were the fittest.

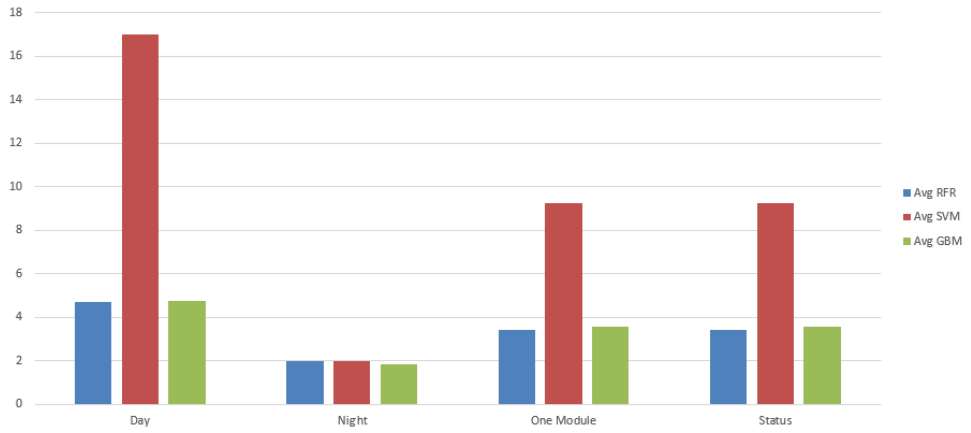


Figure 5.32: Average Results of MAE.

Comparing the values of MSE, Figure 5.33, we can see the discrepancy between the results of SVR with the other two models. In this context, we can still see that GBR models were better on data trained with module with and without column *Status*, but using the average, RFR models for day data were better.

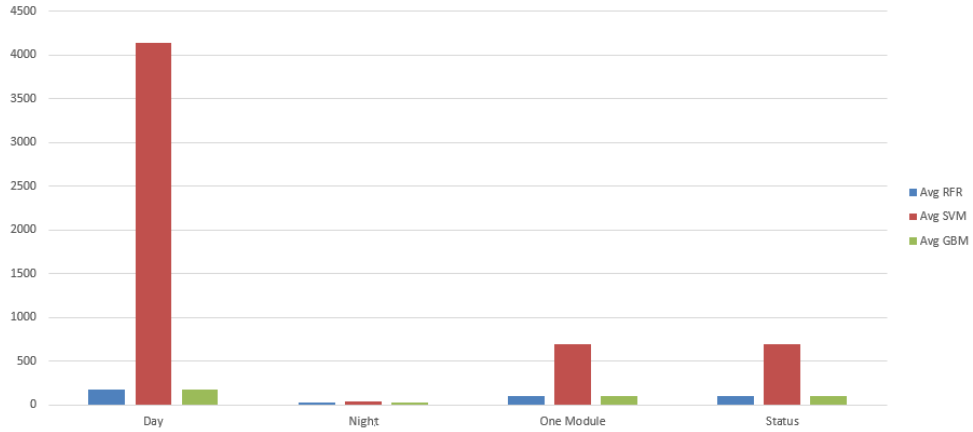


Figure 5.33: Average Results of MSE.

The behavior shown in Figure 5.33 can be compared with those in Figure 5.34, they are similar behaviors. However, for RMSE trained on day data, in this case, the GBR models showed better results.

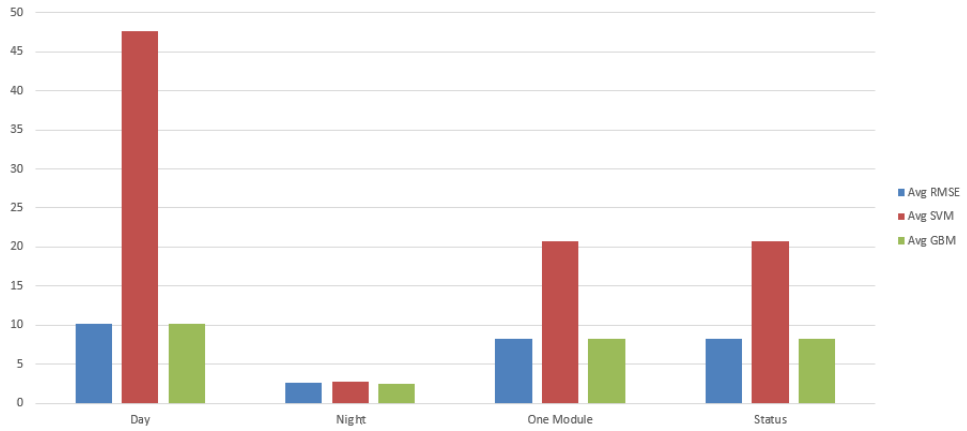


Figure 5.34: Average Results of RMSE.

In order to have all the results of the metrics, another analysis performed was the same as the one previously performed for the full data of the modules, which was the analysis of the behavior of the models in periods of transitions. Figure 5.35 displays the average results of error margins between 1 and 8 units for each of the trained models, Day, Night, *Status* column as a feature, and complete data (One Module). In these results we can see better results for models RFR with complete data and data using the *Status* column as features.

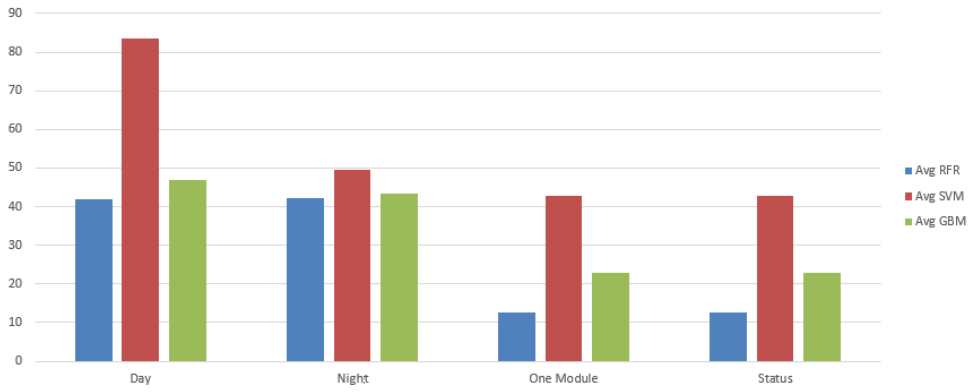


Figure 5.35: Average Errors in Transitions Data.

In addition to the previous analysis, we can compare the errors by the average errors per sensor type. In this way, the RFR still has the best performance and together with that, the 5.36 figure still makes it possible to verify that the error is greater for humidity sensors.

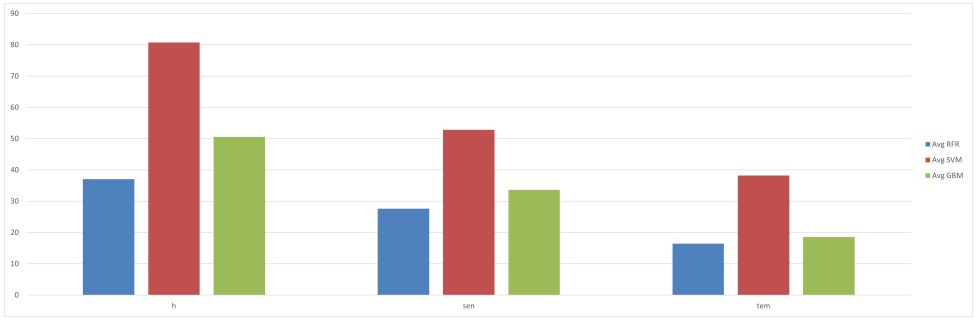


Figure 5.36: Average Errors in Transitions Data per Type of Sensor.

5.7 Regression Models for Clusters

In this step of the analysis, a code was developed that reads the sensor data all together and maps each module to a corresponding cluster number. In this first part as mentioned in the clusters part, 5.4, it was divided into two groups, cluster 01 for n01 to n05 and cluster 02 for n06 to n09.

The code then maps each column in the dataframe to the corresponding cluster. It

does this by iterating over all columns in the dataframe and for each column if the name starts with the module string, it maps the column to the corresponding cluster. As seen in the previous sections, the SVM model was not developed due to the low adherence to the previous tests

The function then performs k-fold cross-validation by splitting the data into training and test sets, fitting a RFR and GBR model to the training data, making predictions on the test data, and recording the MAE, MSE, RMSE, and R-Square values. The k-fold used was 3.

Using the results of the models we can verify that for the metric R-Square the modules of cluster 01 behave similarly for both models, RFR and GBR. In this sense, the behavior for cluster 02 is similar in some points, but for *sen1* of module 06 and has of module 07, the results of GBR are better, but the *sen5* of module 08, the result of the model RFR was perfomed better. Figure 5.37 shows the results of R-Square. It is important to note that the R-Square is close to 1 for all sensors, indicating that the models fit the data well.

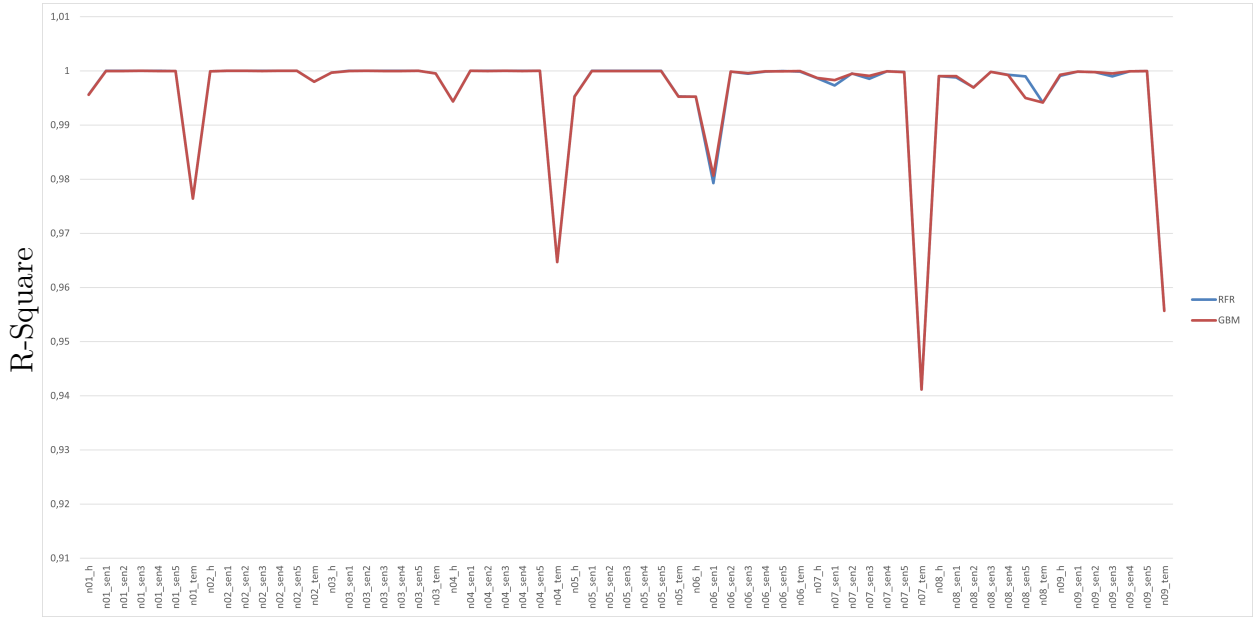


Figure 5.37: R-Square for cluster analysis.

Figure 5.38 shows the results for MAE. We can verify better performance for RFR models. Another point observed is the largest difference between the models for group

01.

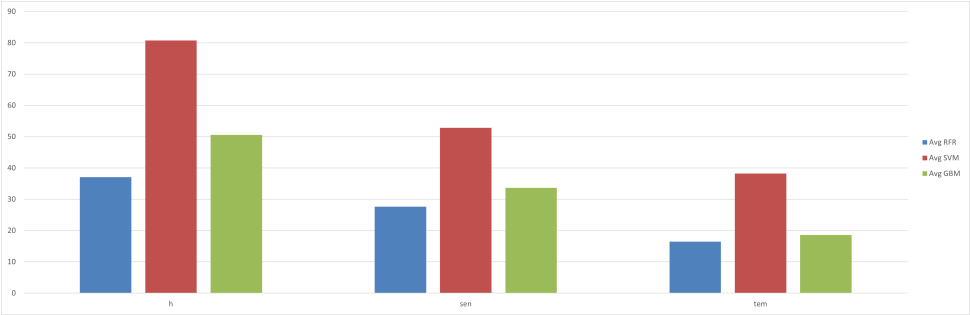


Figure 5.38: MAE for cluster analysis.

The values of MSE can be seen in Figure 5.39 , in it model RFR was better in almost all targets, being surpassed in the *sen1* of module 07 in the *sen3* of module 08.

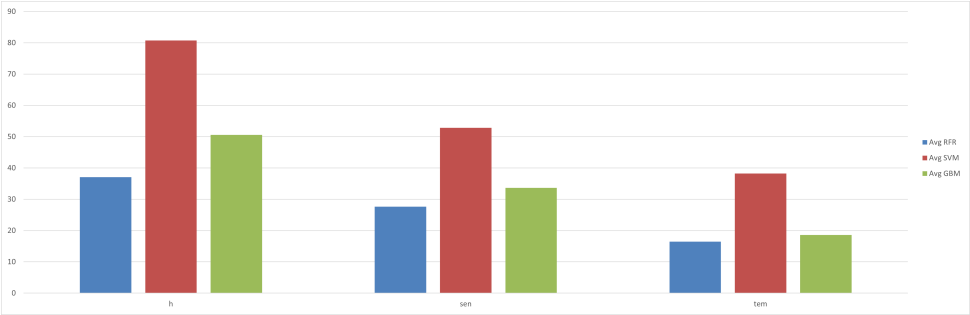


Figure 5.39: MSE for cluster analysis.

Similar to the previous models, for metric RMSE, the RFR models performed better, being closest to the GBR models in group 02. Figure 5.40 displays the results for this metric.

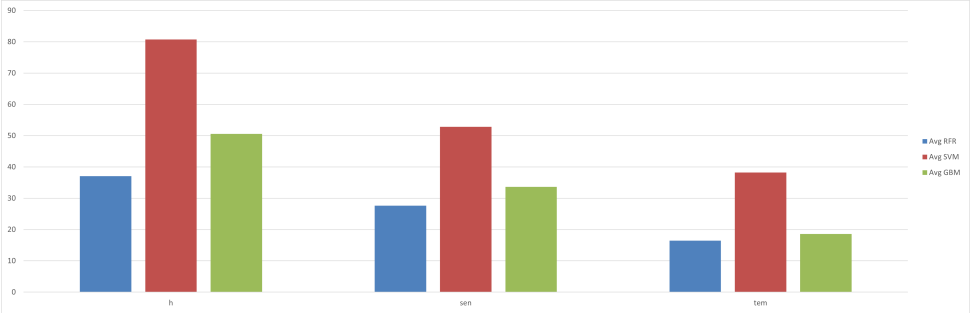


Figure 5.40: RMSE for cluster analysis.

Once all the results are in, we can compare groups 01 and 02. Figure 5.41 displays the average of each of the metrics per model for each of the groups. It is possible to verify that for the metric MAE group 02 presents better results than group 01. For the metric MSE the group 01 presents better results for the model RFR and group 02 presents better results for the model GBR.

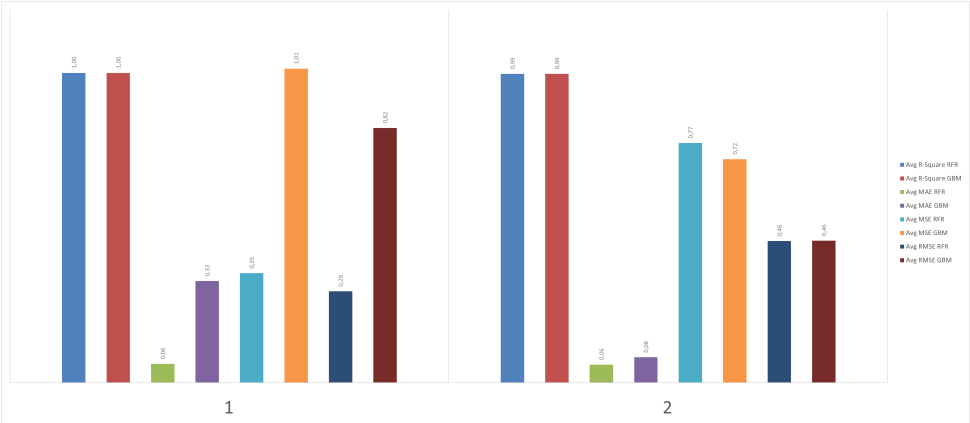


Figure 5.41: Cluster Metrics for the Models

Finally, if we analyze the results of the models on transition data as was done in the previous tests, Figure 5.42 displays the average error margins from 1 to 8 units. We can see a higher error percentage for group 01 with trained model GBR. Overall, the RFR model performed better for this data.

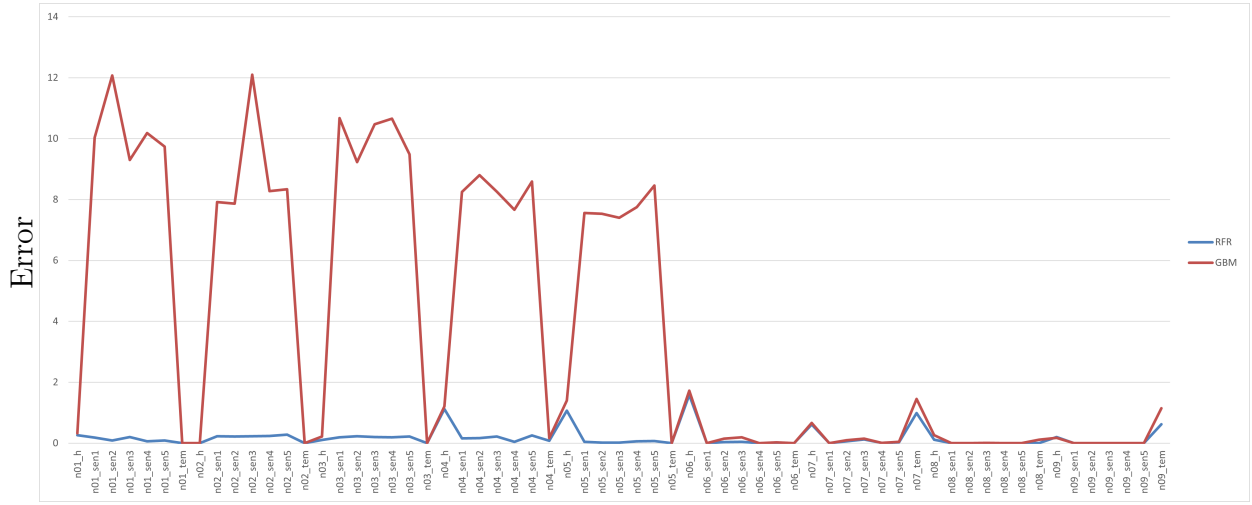
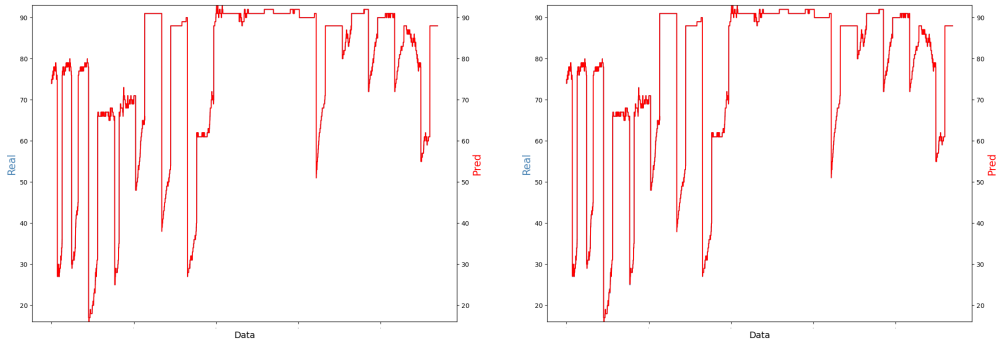


Figure 5.42: Average Error Percentage Between 1 and 8 units for Cluster Trained Data.

Another way to visualize this error is displayed in Figure 5.43, where we can see the high adherence of the trained models to the selected transition data. Both images are from the h of module 01.



(a) RFR Real x Pred (b) GBR Real x Pred

Figure 5.43: Visual analysis of the Actual x Pred values in transition periods for clusters models.

5.8 Regression Models for All Modules

In this step of the analysis, a function was developed that performs a feature selection technique called forward selection. It creates an empty dictionary 'selected_features' and

a list 'remaining_features', which contains all the features. The forward feature selection algorithm is a search algorithm that starts with an empty set of features and iteratively adds features to the set. It starts by choosing the feature that results in the highest performance in the model using a fixed evaluation metric, such as precision or in this case R-Square value, and then adds the feature that results in the highest performance when combined with the previously selected features, and so on. The function uses a RFR to fit and evaluate the model, and adds the feature that results in the best score to the list of selected features The code then stores the 5 best selected features for the current target feature in the 'selected_features' dictionary.

To start, the function uses a threshold which is a parameter used to control the stopping criteria of the forward selection algorithm. The threshold is set to 0.70 by default, which means that the algorithm will stop when the best score obtained by a feature is greater than or equal to 0.70. The idea of using this threshold is to determine the minimum level of accuracy that the model must achieve to consider a feature for selection. Features with a score below the threshold will not be added to the set of selected features, since they are considered not useful for improving model performance.

By adjusting the threshold, it can control the trade-off between model complexity (number of features) and accuracy. A lower threshold means that the algorithm will select more features and result in a more complex model, while a higher threshold will result in a simpler model with fewer features. The function created has the input parameters :

- Target: the target variable for which the regression model is built.
- X_train: a pandas dataframe containing the training data for the independent variables.
- y_train: a pandas Series containing the training data for the dependent variable.
- Remaining_features: a list of features from X_train that have not been selected yet.

- Threshold (optional, default is 0.70): the minimum p-value for a feature to be considered for selection.

The base code iterates through each feature and assigns it as the target feature. For each target feature, it performs the forward selection function of the remaining features. Because of the processing delay due to the amount of data in the data frame, 366268 rows and 76 columns, two data splits were used for the feature surveys. The first split of test and training splits the data into 80% training data and 20% test data. The second split of test and training divides the training data again, this time into 50% of the training data and 50% of the test data.

Using two train splits is a way to handle large data. If the data is too large to fit into memory all at once, it can be split into smaller pieces and trained on each piece. The two train splits can be used to divide the large dataset into a smaller training set and a smaller validation set, allowing the model to be trained on the smaller training.

As a result of the previous feature selection, if we analyze the percentage of the number of features of the same type as the target we can see that humidity sensors feature other humidity sensors as features, on average 51% of the features selected for humidity sensors are other humidity sensors. Figure 5.44 displays this previously described analysis, in which you can see the concentration of the largest percentage values in the targets on the left.

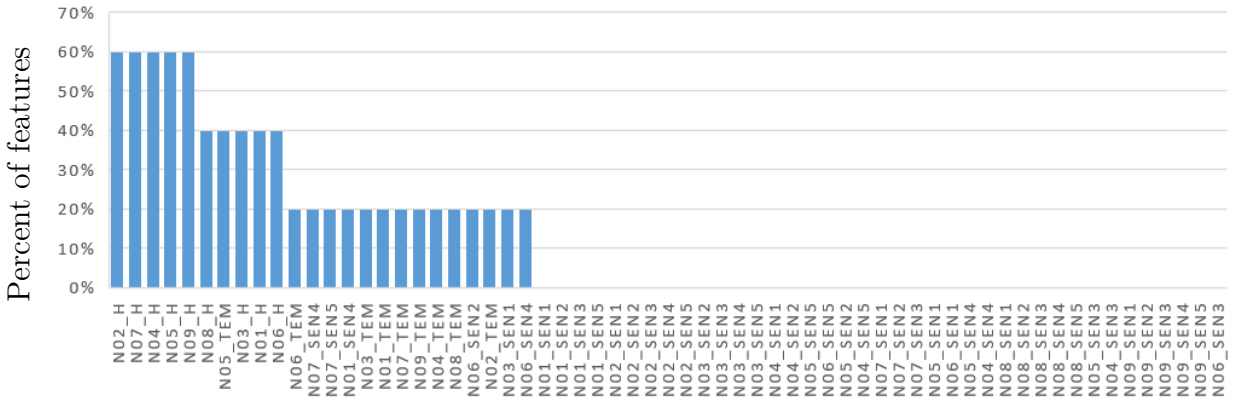


Figure 5.44: Percent of the number of features of the same type as the target

If we consider the flame sensors as all equal, that is, not dividing them between sensor 1,2,3,4 and 5, we can see that there is an influence of sensors of the same type for each of the targets. Calculating the average features of sensors of the same type with this analysis we get a 53% average. This value can be seen in Figure 5.45.

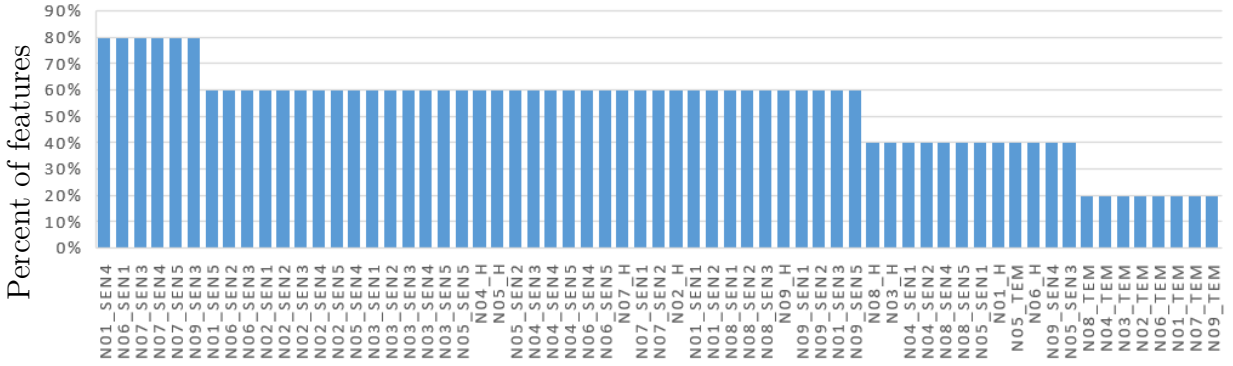


Figure 5.45: Percent of the number of features of the same type as the target considering the same flame sensors

Figure 5.46 shows another analysis that was performed. In this analysis, we can see that on average 55% of the features are from the same module as the target. However, we can verify three targets that do not have features of the same module, all humidity sensor targets. However, all the other 60 targets have features from the same module.

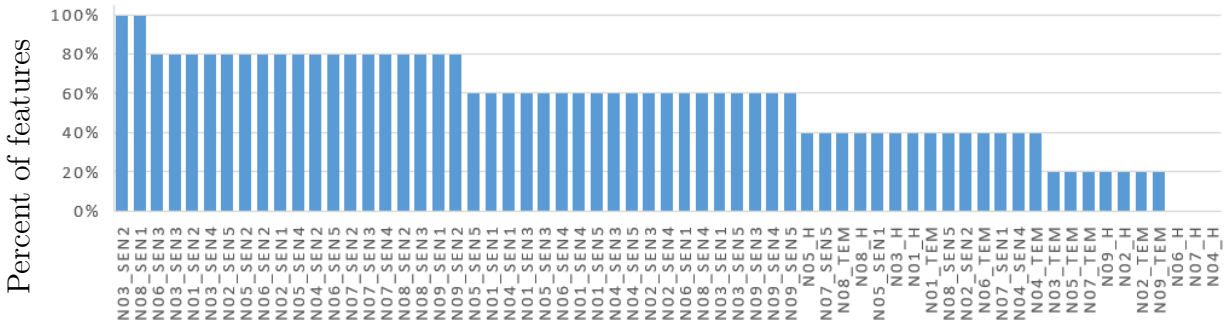


Figure 5.46: Percent of the number of features of the same module as the target

Figure 5.47 shows a final analysis of the selected features. In this analysis we can see that 34% of the selected features are of the same type and model as the target, considering the flame sensors as the same type. That is, we can see a low influence of sensors of the same type from the same module. However, it is not excluded that sensors that are not in the analysis are not influential in the analysis of the targets.

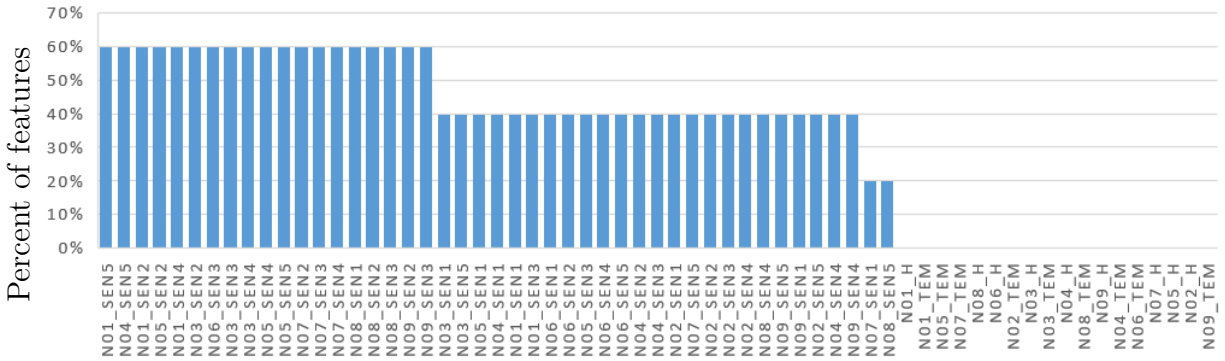


Figure 5.47: Percent of the number of features of the same type and module considering the flame sensors as the same

Thus, with all these 5 features for each target, we got the best score. Figure 5.48 displays the best scores for each of figures, and show that there was a good score for all those features. In this code, the score is the R-Square of the regression model.

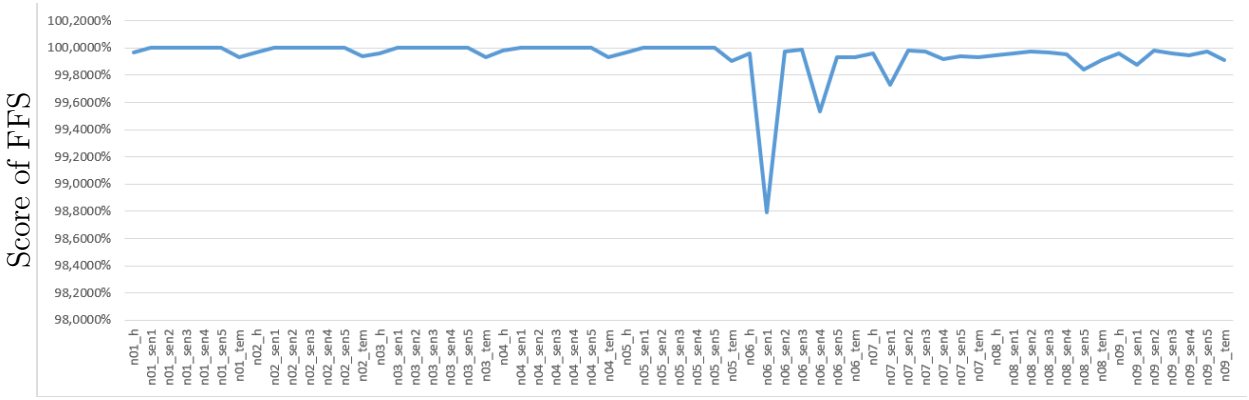


Figure 5.48: Best score for forward feature selection

Based on the results of the forward feature selection, a code for regression model analysis was developed. As seen in the previous sections, the SVM model was not developed due to the low adherence to the previous tests. Another point that deserves to be highlighted is that the best result presented in Figure 5.48 may not be the same after the next tests. The reason for this difference is that for the forward feature selection, the data was split twice to improve processing time and 50 estimators were used, while for the training with all data it was split into 80 % and 20 % of test and kept 100 estimators for regression models.

After training the data, in Figure 5.49 it can be seen that the R-Square scores for the RFR and GBR models are generally high, with values close to 1. For example, for *sen1* of module 01, the R-Square score is 0.9998 for RFR and 0.9998 for GBR. This suggests that the models have a good fit to the data.

However, there are some exceptions, such as *sen1* of module 06, where the R-Square score is 0.1708 for RFR and 0.569 for GBR. This suggests that the fit of the models to the data is not good for this particular sensor. Overall, the data suggest that the RFR and GBR models have a good fit to the data for most objects and sensors, with a few exceptions.

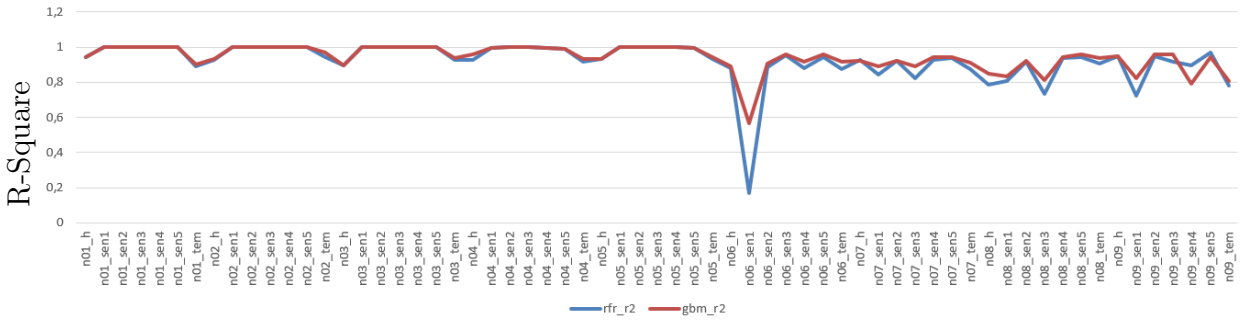


Figure 5.49: R-Square Results of Full Data Regression.

Using MAE, Figure 5.50 it is difficult to determine an overall winner based on the results alone. Both models seem to have some strengths and weaknesses between the different sensors. But if we compare on particular sensors it is possible to find the better model, for example, for humidity of module 01, the RFR model has an MAE of 2,90 and the GBR model has an MAE of 3,48. This suggests that the RFR model performs better on this particular sensor compared to the GBR model.

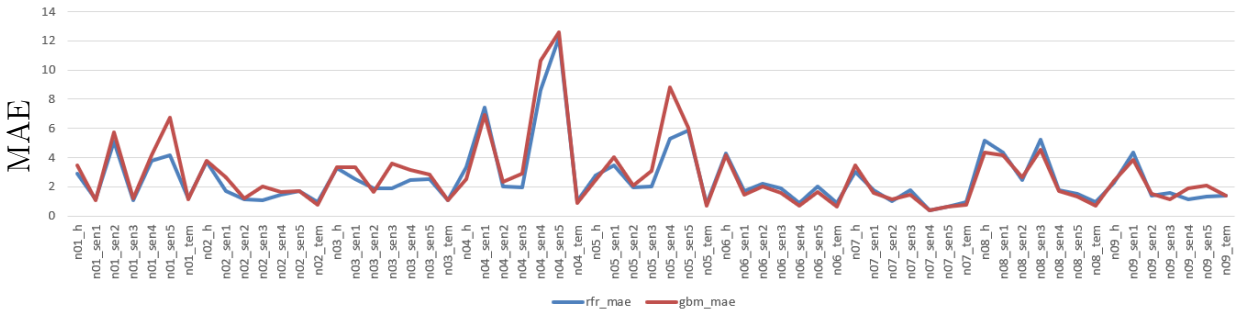


Figure 5.50: MAE Results of Full Data Regression.

Based on the data, it seems that for most sensors, the GBR model has a lower MSE compared to the RFR model, which indicates that the predictions made by the GBR model are more accurate. However, there are some exceptions where the RFR model performs better than the GBR model (e.g., *h* of module 07,*tem* of module 08). Figure 5.51 shows the results of the MSE.

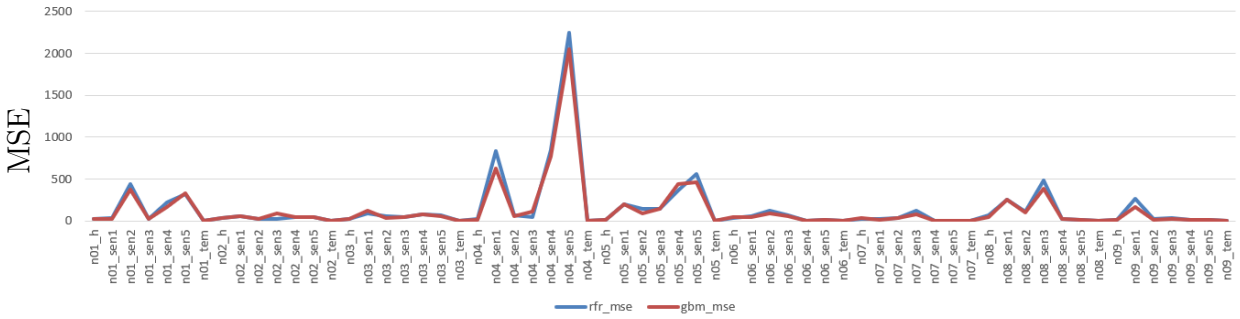


Figure 5.51: MSE of Results Full Data Regression.

Based on the values in Figure 5.52 , it seems that the GBR is outperforming the RFR in most cases, as the RMSE values are lower. This suggests that the GBR model is better at fitting the data and has higher accuracy in its predictions.

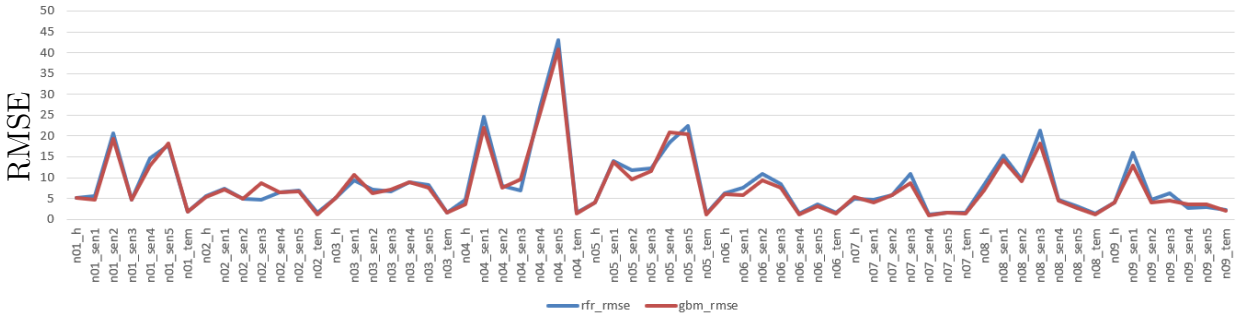


Figure 5.52: RMSE Results Full Data Regression

Another analysis was to verify the behavior of the models in periods of transitions between day and night. For this, the transition data were separated into another dataframe and divided in two to facilitate plotting. A comparison was then performed between the predicted values of transition data with the actual values for that separate dataframe, this comparison uses a margin of error of 2 units. Then the number of cases where the absolute error between the predicted and actual values is greater than the error margin was counted.

One way to interpret these results would be to compare the number of errors to the total number of instances, to get an error rate. Another way would be to plot the

predicted vs actual values for y_new and visually inspect the differences between the two. The larger the differences, the lower the accuracy of the model.

The percentage of errors for each model is calculated by dividing the number of instances where the predicted value deviates from the actual value by more than the error margin, with the total number of instances.

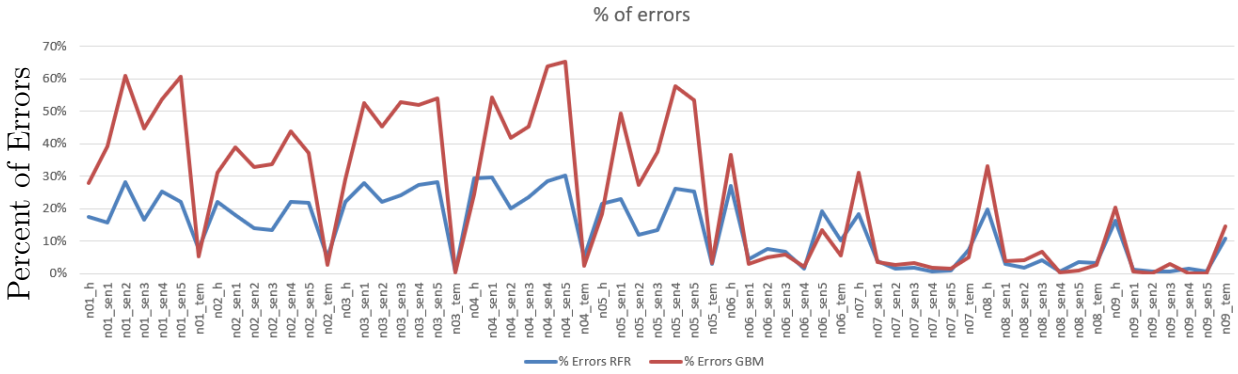
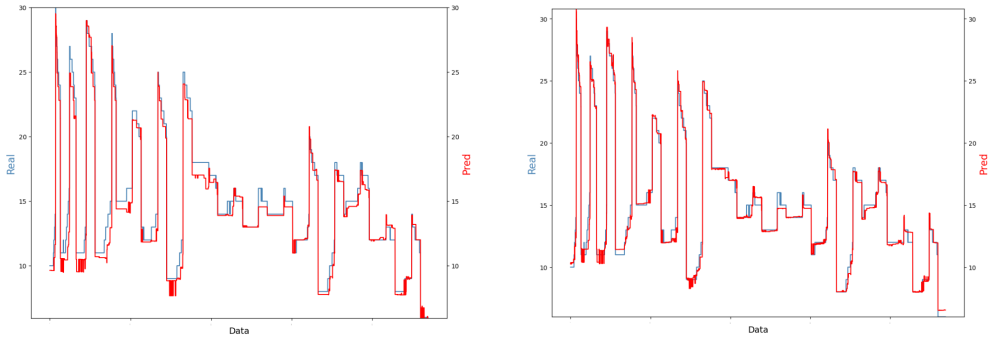


Figure 5.53: Errors in Transition Periods.

From Figure 5.53, it appears that the performance of the GBR model is worse than the RFR model, as the percentage of errors is generally higher for the GBR model. However, it is important to note that the performance of both models can vary greatly depending on the target category. For example, in the "n06_sen1" target, the RFR model has a higher percentage of errors (4%) compared to the GBR model (3%). On the other hand, in the "n01_sen2" target category, the RFR model has a lower percentage of errors (28%) compared to the GBR model (61%).

It is also important to consider the error margin in evaluating the performance of these models. A smaller error margin means that the models are expected to have a smaller deviation from the actual values, making the prediction more accurate.

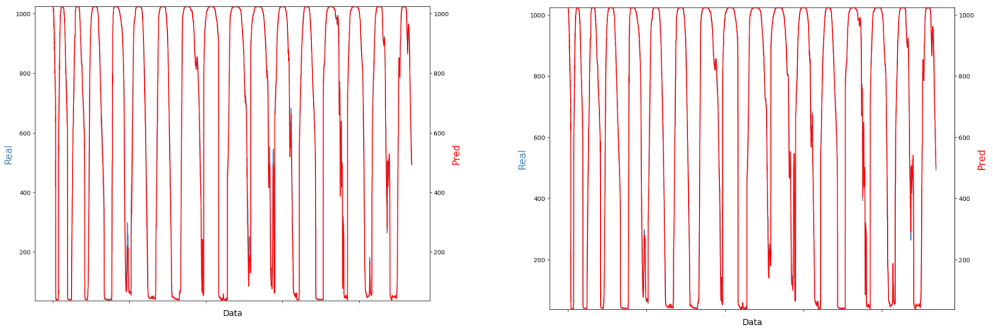


(a) RFR Real x Pred (b) GBR Real x Pred

Figure 5.54: Visual analysis of the Actual x Pred values in transition periods.

Figure 5.54 shows the visual analysis of the actual versus predicted values for the *tem* of module 02. All in all, visual analysis is of great importance for temperature and humidity sensors.

For the flame sensors, the visual analysis is not very effective, the predicted data show a good fit to the trend and range of the actual values. It is interesting to do this analysis because the values of flame sensors present typical visual patterns, which are not easily visualized in other types of sensors. This pattern can be seen in Figure 5.55, where we see the standardized variation of the data of flame *sen5* of module 02.



(a) RFR Real x Pred (b) GBR Real x Pred

Figure 5.55: Visual analysis of the Actual x Pred values in transition periods 2.

Overall, the results suggest that the RFR model is a better choice for predicting the transitions target values in this case.

5.9 Obtained Results

As discussed in the previous sections, the models have similar behavior for data trained in different ways. In this context, as one of the problems is the sending of false alerts, we can see that although in a general view the data present GBR as the best model. However, for the transition periods, where data varies more, the RFR model performs better across all training types and datasets.

Table 5.5 displays the results of the averages of all metrics for all trained datasets. There is no difference between the data trained per module with the *Status* column and without it. However, models trained using only night data performed worse than others. Models using cluster data performed better, but took longer to run than models trained by modules.

Table 5.5: Result for all Metrics per Datasets

Datasets	**Avg MAE**		**Avg R-Square**		**Avg MSE**		**Avg RMSE**	
	RFR	**GBM**	**RFR**	**GBM**	**RFR**	**GBM**	**RFR**	**GBM**
Cluster	0.06	0.22	1.00	1.00	0.54	0.88	0.37	0.66
Day	4.71	4.73	0.71	0.75	173.69	170.54	10.21	10.17
Full	2.61	2.82	0.92	0.94	142.56	128.18	8.36	7.91
Nigth	1.97	1.85	-1.79	-0.85	28.47	25.15	2.63	2.46
Per Module	3.39	3.57	0.78	0.82	103.17	103.34	8.28	8.20
Status	3.39	3.57	0.78	0.82	103.17	103.34	8.28	8.20

Table 5.6 displays the most detailed results by sensors, across almost all datasets and sensors, the GBR model consistently outperforms the RFR model in terms of R-Square, MSE, and RMSE. The only metric where RFR has been experienced to perform is MAE.

The performance of both models is highly dependent on the specific dataset and sensor being used. For example, in Cluster dataset, the RFR model performs significantly worse on *tem* targets compared to the other two sensors, while in *Day* dataset, GBR model performs significantly better on temperature data compared to the other two sensors.

For the Cluster dataset, the performance of both models is relatively consistent across

all sensors, with no sensor performing significantly better or worse than the others. However, for the Day dataset, the *tem* consistently outperforms the other two sensors in all metrics and in both models. In general, both models perform better on *h* targets than *tem* or *sen* data, as evidenced by the lower error metrics for the *h* sensor compared to the *sen* and *tem* sensors.

Finally, it's worth noting that the overall performance of both models is very good. This suggests that both models are effective in predicting environmental parameters based on the provided sensor data.

Table 5.6: Result for all Metrics per Datasets and Sensors

Datasets	**Sensors**	**Avg MAE**		**Avg R-Square**		**Avg MSE**		**Avg RMSE**	
		RFR	**GBM**	**RFR**	**GBM**	**RFR**	**GBM**	**RFR**	**GBM**
Cluster	h	0,17	0,17	1,00	1,00	1,37	1,37	0,59	0,60
	sen	0,03	0,25	1,00	1,00	0,37	0,85	0,32	0,73
	tem	0,12	0,12	0,98	0,98	0,54	0,54	0,38	0,38
Day	h	12,54	12,16	0,20	0,29	264,99	243,44	15,47	14,86
	sen	3,50	3,65	0,89	0,91	187,35	187,68	10,46	10,59
	tem	2,91	2,70	0,31	0,41	14,11	11,95	3,67	3,39
Full	h	3,42	3,33	0,91	0,92	30,82	28,25	5,29	5,04
	sen	2,76	3,10	0,93	0,95	192,83	173,34	10,32	9,78
	tem	1,02	0,90	0,89	0,92	2,90	2,32	1,61	1,40
Night	h	9,72	9,33	-0,34	-0,26	182,81	162,13	12,78	12,17
	sen	0,36	0,33	-2,39	-1,16	1,48	1,47	0,54	0,52
	tem	2,31	1,98	-0,24	0,10	9,05	6,52	2,93	2,49
Per Module	h	10,18	10,21	0,34	0,38	184,69	179,08	13,24	12,86
	sen	2,19	2,49	0,93	0,95	105,03	106,96	8,25	8,30
	tem	2,63	2,35	0,46	0,60	12,37	9,48	3,46	3,03
Status	h	10,18	10,21	0,34	0,38	184,69	179,08	13,24	12,86
	sen	2,19	2,49	0,93	0,95	105,03	106,96	8,25	8,30
	tem	2,63	2,35	0,46	0,60	12,37	9,48	3,46	3,03

With all these analyses, forecasting the daily behavior of the sensors on normal days, in case of fire, it would be possible to identify deviations from the expected behavior. Thus, with the identification of the difference between actual and predicted values, it is possible to detect fire risks in the analyzed areas.

Chapter 6

Conclusions

This chapter will present some conclusions of this work. For the data available for the study, it is possible to verify the standardization of the flame sensor data during the course of time. The data from these sensors vary in a standardized way between daytime values and nighttime values.

The strategy of using the algorithm to identify the periods of the day can be very useful, since it was reported at the beginning of the work the difficulty with false alerts in these periods.

Therefore, with the use of this identification, it is possible to use RFR for periods when there is a transition and GBR for normal times. In other words, if it is identified that the received data comes from day or night data, it can be used with regression models trained with data by modules. If it is a transition time, the work showed that the random forest regression model was the best performing, that is, the RFR model trained with data by modules.

The study also allowed to verify that of the three proposed models, the SVR model was the one that performed worse than all other two. Both GBR and RFR performed well for all data types trained.

In terms of the type of data trained, it was found that data trained on clustered basis performed better than the others. However, both per module and with status also performed well.

Given that data from each module is sent at different times, the best strategy for the models is to train for each individual module, avoiding dependence on data not received from other modules.

Therefore, it is concluded that for the data that was made available at the beginning, the regression models using RFR and GBR proved to be better for this data. In order to reduce false positives and to facilitate abnormality detection, some improvements are necessary.

6.1 Suggestions for Future Works

In the future, it is suggested to train the data for other months. Since the seasonality in the region's climate is quite strong, i.e. if we use data from September and October to predict behavior in other months, low results are expected. Therefore, it is necessary to train data for the remaining months.

Another suggestion is to reanalyze the data at night, evaluate other models and understand why the models in this work had a not so good result, since the data does not oscillate much.

It is suggested to maintain a standardized code for training data by module in a uniform way. In this context, it is still necessary to implement the algorithms that will be trained in the future on real time data for testing and validation of the models suggested in this work.

Once the algorithm is implemented, it is expected that the alarms generated will be monitored through deviations in the behavior of the data.

Bibliography

[1] U. Dampage, L. Bandaranayake, R. Wanasinghe, K. Kottahachchi, and B. Jayasanka, "Forest fire detection system using wireless sensor networks and machine learning", en, *Sci. Rep.*, vol. 12, no. 1, p. 46, Jan. 2022.

[2] J. G. Pausas and J. E. Keeley, "Abrupt climate-independent fire regime changes", *Ecosystems*, vol. 22, pp. 18–32, 2019. DOI: 10.1007/s10021-018-0279-6. [Online]. Available: https://doi.org/10.1007/s10021-018-0279-6.

[3] M. G. Pereira and et al., "Recent changes in the fire regime across europe highlighting the importance of weather-fire interactions", *Forests*, vol. 9, no. 2, p. 89, 2018. DOI: 10.3390/f9020089. [Online]. Available: https://doi.org/10.3390/f9020089.

[4] European Commission, Joint Research Centre and San-Miguel-Ayanz, J. and Durrant, T. and Boca, R., *Forest Fires in Europe, Middle East and North Africa 2021*. Publications Office of the European Union, 2022. DOI: 10.2760/34094. [Online]. Available: https://data.europa.eu/doi/10.2760/34094.

[5] T. Brito, B. F. Azevedo, J. Mendes, M. Zorawski, F. P. Fernandes, A. I. Pereira, J. Rufino, J. Lima, and P. Costa, "Data acquisition filtering focused on optimizing transmission in a lorawan network applied to the wsn forest monitoring system", *Sensors*, vol. 23, no. 3, 2023, ISSN: 1424-8220. DOI: 10.3390/s23031282. [Online]. Available: https://www.mdpi.com/1424-8220/23/3/1282.

[6] T. Freitas, J. Santos, A. Silva, J. Martins, and H. Fraga, "Climate change projections for bioclimatic distribution of castanea sativa in portugal", *Agronomy*, vol. 12,

p. 1137, 2022. DOI: 10.3390/agronomy12071137. [Online]. Available: https://www.mdpi.com/2073-4395/12/7/1137.

[7] POSEUR - Programa Operacional Sustentabilidade e Eficiência no Uso de Recursos. (2017). Sistemas de Videovigilância de Prevenção de Incêndios a Partir de 2017, [Online]. Available: https://poseur.portugal2020.pt/media/4140/plano_nacional_defesa_floresta_contra_incendios.pdf (visited on 12/31/2022).

[8] M. Pollet and F. Jambon, "Forest fire detection from space: A review", *Remote Sensing*, vol. 11, p. 1014, 2019.

[9] G. Fazakas and C. Benedek, "Detection of forest fires using remote sensing techniques: A review", *ISPRS International Journal of Geo-Information*, vol. 9, p. 199, 2020.

[10] V. Zope, T. Dadlani, A. Matai, P. Tembhurnikar, and R. Kalani, "Iot sensor and deep neural network based wildfire prediction system", in *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2020, pp. 205–208. DOI: 10.1109/ICICCS48265.2020.9120949.

[11] C. Inal and I. F. Akyildiz, "Real-time detection and tracking of forest fires using iot, uavs, and machine learning", *IEEE Communications Magazine*, vol. 58, pp. 66–73, 2020.

[12] J. Chen, Y. Chen, and Y. Chen, "A review of machine learning applications in wildfire management", *Fire*, vol. 3, p. 47, 2020.

[13] J. C. S. M. C. Mendes, " deep learning na identificação de incêndios florestais", in *Engenharia Industrial*, Bragança, Portugal: IPB - Instituto Politécnico de Bragança, 2020, p. 72. [Online]. Available: http://hdl.handle.net/10198/23054.

[14] T. Brito, B. F. Azevedo, A. Valente, A. I. Pereira, J. Lima, and P. Costa, "Environment monitoring modules with fire detection capability based on iot methodology", in *Science and Technologies for Smart Cities*, S. Paiva, S. I. Lopes, R. Zitouni, N.

Gupta, S. F. Lopes, and T. Yonezawa, Eds., Cham: Springer International Publishing, 2021, pp. 211–227.

[15] T. Brito, M. Zorawski, J. Mendes, B. F. Azevedo, A. I. Pereira, J. Lima, and P. Costa, "Optimizing data transmission in a wireless sensor network based on lorawan protocol", in *Optimization, Learning Algorithms and Applications*, A. I. Pereira, F. P. Fernandes, J. P. Coelho, J. P. Teixeira, M. F. Pacheco, P. Alves, and R. P. Lopes, Eds., Cham: Springer International Publishing, 2021, pp. 281–293, ISBN: 978-3-030-91885-9.

[16] B. F. Azevedo, T. Brito, J. Lima, and A. I. Pereira, "Optimum sensors allocation for a forest fires monitoring system", *Forests*, vol. 12, no. 4, 2021, ISSN: 1999-4907. DOI: 10.3390/f12040453. [Online]. Available: https://www.mdpi.com/1999-4907/12/4/453.

[17] M. Arif, K. Alghamdi, S. Sahel, S. Alosaimi, M. Alsahaft, M. Alharthi, and M. Arif, "Role of machine learning algorithms in forest fire management: A literature review", *J Robotics Autom*, vol. 5, no. 1, pp. 212–226, 2021.

[18] L. Yu, S. Wang, K. K. Lai, and F. Wen, "A multiscale neural network learning paradigm for financial crisis forecasting", *Neurocomputing*, vol. 73, no. 4, pp. 716–725, 2010, Bayesian Networks / Design and Application of Neural Networks and Intelligent Learning Systems (KES 2008 / Bio-inspired Computing: Theories and Applications (BIC-TA 2007), ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2008.11.035. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231209004342.

[19] S. P. Chatzis, V. Siakoulis, A. Petropoulos, E. Stavroulakis, and N. Vlachogiannakis, "Forecasting stock market crisis events using deep and statistical machine learning techniques", *Expert Systems with Applications*, vol. 112, pp. 353–371, 2018, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2018.06.032. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417418303798.

[20] M. Sangiorgio, S. Barindelli, R. Biondi, E. Solazzo, E. Realini, G. Venuti, and G. Guariso, "Improved extreme rainfall events forecasting using neural networks and water vapor measures", Sep. 2019.

[21] K. Whan and M. Schmeits, "Comparing area probability forecasts of (extreme) local precipitation using parametric and machine learning statistical postprocessing methods", *Monthly Weather Review*, vol. 146, no. 11, pp. 3651–3673, 2018. DOI: 10.1175/MWR-D-17-0290.1. [Online]. Available: https://journals.ametsoc.org/view/journals/mwre/146/11/mwr-d-17-0290.1.xml.

[22] S. Yang, M. Lupascu, and K. S. Meel, *Predicting forest fire using remote sensing data and machine learning*, 2021. DOI: 10.48550/ARXIV.2101.01975. [Online]. Available: https://arxiv.org/abs/2101.01975.

[23] R. Sharma, S. Rani, and I. Memon, "A smart approach for fire prediction under uncertain conditions using machine learning", *MULTIMEDIA TOOLS AND APPLICATIONS*, vol. 79, no. 37-38, pp. 28 155–28 168, Oct. 2020, ISSN: 1380-7501. DOI: 10.1007/s11042-020-09347-x.

[24] M. R. Nosouhi, K. Sood, N. Kumar, T. Wevill, and C. Thapa, "Bushfire risk detection using internet of things: An application scenario", *IEEE INTERNET OF THINGS JOURNAL*, vol. 9, no. 7, pp. 5266–5274, Apr. 2022, ISSN: 2327-4662. DOI: 10.1109/JIOT.2021.3110256.

[25] A. A. A. Alkhatib, "A review on forest fire detection techniques", *International Journal of Distributed Sensor Networks*, vol. 10, no. 3, p. 597 368, 2014. DOI: 10.1155/2014/597368. eprint: https://doi.org/10.1155/2014/597368. [Online]. Available: https://doi.org/10.1155/2014/597368.

[26] A. Ferrari, S. Lombardi, and A. Signoroni, "Bacterial colony counting by convolutional neural networks", vol. 2015, Aug. 2015, pp. 7458–7461. DOI: 10.1109/EMBC.2015.7320116.

[27] Y. Lai, "A comparison of traditional machine learning and deep learning in image recognition", *Journal of Physics: Conference Series*, vol. 1314, no. 1, p. 012 148, Oct. 2019. DOI: 10.1088/1742-6596/1314/1/012148. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1314/1/012148.

[28] R. W. Pettit, R. Fullem, C. Cheng, and C. I. Amos, "Artificial intelligence, machine learning, and deep learning for clinical outcome prediction", *Emerging topics in life sciences*, vol. 5, no. 6, pp. 729–745, 2021. DOI: 10.1042/ETLS20210246. [Online]. Available: https://doi.org/10.1042/ETLS20210246.

[29] F. Pereira de Figueiredo, "Tp555 - introdução à inteligência artificial e machine learning", Tech. Rep., Mar. 2020. DOI: 10.13140/RG.2.2.10108.90241.

[30] E. R. Okoroafor, C. M. Smith, K. I. Ochie, C. J. Nwosu, H. Gudmundsdottir, and M. (Jabs) Aljubran, "Machine learning in subsurface geothermal energy: Two decades in review", *Geothermics*, vol. 102, p. 102 401, 2022, ISSN: 0375-6505. DOI: https://doi.org/10.1016/j.geothermics.2022.102401. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0375650522000530.

[31] I. C. Education, "Machine learning", Tech. Rep., Jul. 2020. [Online]. Available: https://www.ibm.com/cloud/learn/machine-learning.

[32] L. Xi, Z. Yun, H. Liu, R. Wang, X. Huang, and H. Fan, "Semi-supervised time series classification model with self-supervised learning", *Engineering Applications of Artificial Intelligence*, vol. 116, p. 105 331, 2022, ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2022.105331. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197622003633.

[33] A. Guha, R. Lei, J. Zhu, X. Nguyen, and D. Zhao, "Robust unsupervised learning of temporal dynamic vehicle-to-vehicle interactions", *Transportation Research Part C: Emerging Technologies*, vol. 142, p. 103 768, 2022, ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2022.103768. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X22002005.

[34] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural Networks*, vol. 61, pp. 85–117, 2015, ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2014.09.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608014002135.

[35] M. Kamber, L. Winstone, W. Gong, S. Cheng, and J. Han, "Generalization and decision tree induction: Efficient classification in data mining", in *Proceedings Seventh International Workshop on Research Issues in Data Engineering. High Performance Database Management for Large-Scale Applications*, 1997, pp. 111–120. DOI: 10.1109/RIDE.1997.583715.

[36] I. Batool and T. A. Khan, "Software fault prediction using data mining, machine learning and deep learning techniques: A systematic literature review", *Computers and Electrical Engineering*, vol. 100, p. 107 886, 2022, ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2022.107886. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790622001744.

[37] E. Hüllermeier, "Fuzzy methods in machine learning and data mining: Status and prospects", *Fuzzy Sets and Systems*, vol. 156, no. 3, pp. 387–406, 2005, 40th Anniversary of Fuzzy Sets, ISSN: 0165-0114. DOI: https://doi.org/10.1016/j.fss.2005.05.036. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165011405002861.

[38] P. Kavitha and V. Subramaniyaswamy, "A study on the detection and prevention of forest fires using machine learning techniques", *International Journal of Advanced Science and Technology*, vol. 29, no. 9s, pp. 2932–2937, 2020.

[39] T. T. Nguyen, T. H. Nguyen, T. T. Nguyen, and T. Pham, "An intelligent forest fire detection and monitoring system based on machine learning and iot", *Sensors*, vol. 21, no. 3, p. 920, 2021.

[40] G. Sarailidis, T. Wagener, and F. Pianosi, "Integrating scientific knowledge into machine learning using interactive decision trees", *Computers & Geosciences*, p. 105 248, 2022, ISSN: 0098-3004. DOI: https://doi.org/10.1016/j.cageo.2022.105248.

[Online]. Available: https://www.sciencedirect.com/science/article/pii/S0098300422001972.

[41] F. L. d. M. Lucas Batista Fontes, "Análise de aprendizado de máquina aplicada a focos de incêndio nos estados brasileiros", *UFRJ*, 2020. [Online]. Available: http://www.repositorio.poli.ufrj.br/monografias/monopoli10031735.pdf.

[42] T. A. Team, *Decision trees explained with a practical example*, Feb. 2021. [Online]. Available: https://towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53.

[43] R. Silipo, *From a single decision tree to a random forest*, Oct. 2019. [Online]. Available: https://towardsdatascience.com/from-a-single-decision-tree-to-a-random-forest-b9523be65147.

[44] L. M. O. D. SILVA, "Uma aplicação de árvores de decisão, redes neurais e knn para a identificação de modelos arma não-sazonais e sazonais", *PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO - PUC-RIO*, vol. 7587, 2005. DOI: https://doi.org/10.17771/PUCRio.acad.7587. [Online]. Available: https://www.maxwell.vrac.puc-rio.br/colecao.php?strSecao=resultado&nrSeq=7587@1.

[45] E. Laber and L. Murtinho, "Minimization of gini impurity: Np-completeness and approximation algorithm via connections with the k-means problem", *Electronic Notes in Theoretical Computer Science*, vol. 346, pp. 567–576, 2019, The proceedings of Lagos 2019, the tenth Latin and American Algorithms, Graphs and Optimization Symposium (LAGOS 2019), ISSN: 1571-0661. DOI: https://doi.org/10.1016/j.entcs.2019.08.050. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S157106611930101X.

[46] H. Cao, "A systematic study for learning-based software defect prediction", *Journal of Physics: Conference Series*, vol. 1487, no. 1, p. 012 017, Mar. 2020. DOI: 10.1088/1742-6596/1487/1/012017. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1487/1/012017.

[47] L. Breiman, "Random forests", en, *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001, ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

[48] G. Biau and E. Scornet, "A random forest guided tour", *TEST*, vol. 25, no. 2, pp. 197–227, Jun. 2016, ISSN: 1863-8260. DOI: 10.1007/s11749-016-0481-7. [Online]. Available: https://doi.org/10.1007/s11749-016-0481-7.

[49] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees.", *Neural Computation*, vol. 9, pp. 1545–1588, Oct. 1997. DOI: 10.1162/neco.1997.9.7.1545.

[50] M. Ay, L. Özbakır, S. Kulluk, B. Gülmez, G. Öztürk, and S. Özer, "Fc-kmeans: Fixed-centered k-means algorithm", *Expert Systems with Applications*, vol. 211, p. 118 656, 2023, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.118656. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422016979.

[51] R. M. Adnan, P. Khosravinia, B. Karimi, and O. Kisi, "Prediction of hydraulics performance in drain envelopes using kmeans based multivariate adaptive regression spline", *Applied Soft Computing*, vol. 100, p. 107 008, 2021, ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2020.107008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494620309479.

[52] A. Nowak - Brzezińska and C. Horyń, "Outliers in rules - the comparision of lof, cof and kmeans algorithms.", *Procedia Computer Science*, vol. 176, pp. 1420–1429, 2020, Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 24th International Conference KES2020, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.09.152. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920320524.

[53] *K-means clustering algorithm*. [Online]. Available: https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning.

[54] A. Falini, "A review on the selection criteria for the truncated svd in data science applications", *Journal of Computational Mathematics and Data Science*, p. 100 064, 2022, ISSN: 2772-4158. DOI: https://doi.org/10.1016/j.jcmds.2022.100064. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772415822000244.

[55] R. Bisht, D. Kumar, and M. Paswan, "Accuracy prediction using data-driven algorithm for carbon containing compounds", *Materials Today: Proceedings*, 2022, ISSN: 2214-7853. DOI: https://doi.org/10.1016/j.matpr.2022.08.119. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214785322052750.

[56] T. Kodinariya and P. Makwana, "Review on determining of cluster in k-means clustering", *International Journal of Advance Research in Computer Science and Management Studies*, vol. 1, pp. 90–95, Jan. 2013.

[57] B. Saji, *In-depth intuition of k-means clustering algorithm in machine learning*. [Online]. Available: https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learninghttps://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/.

[58] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction To Cluster Analysis*. Jan. 1990, ISBN: 0-471-87876-6. DOI: 10.2307/2532178.

[59] C. Cortes and V. Vapnik, "Support-vector networks", *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[60] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines", *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[61] J. H. Friedman, "Greedy function approximation: A gradient boosting machine", *Annals of statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[62] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system", in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 785–794.

[63] P. Cortez and A. Morais, "A data mining approach to predict forest fires using meteorological data", Jan. 2007.

[64] M. A. Morais, N. G. Pinto, and J. J. Pereira, "Linear regression models for predicting the burned area of forest fires", *Climatic change*, vol. 126, no. 4, pp. 507–519, 2014.

[65] S. Makridakis and S. C. Wheelwright, *Forecasting: methods and applications*. John Wiley & Sons, 1989.

[66] S. J. Hwang and C. J. Lin, "Evaluation of forecast accuracy of electricity load: A comparative study of various error measures", *Energy*, vol. 82, pp. 118–127, 2015.

[67] C. Chatfield, "Modeling volatility and forecasting", *Journal of Business & Economic Statistics*, vol. 13, no. 3, pp. 369–375, 1995.

[68] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy", *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[69] Pandas Development Team, *Pandas dataframe.describe() method*, `https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html`, Accessed on: 07/03/2023, 2021.

[70] J. W. Tukey, *Exploratory Data Analysis*. Addison-Wesley, 1977.

[71] N. US Department of Commerce, *Esrl global monitoring laboratory - global radiation and aerosols*, Oct. 2005. [Online]. Available: `https://gml.noaa.gov/grad/solcalc/calcdetails.html`.

Appendix A

Original Project Proposal

Code used for analyzing by module:

```
def regression_onemodule(df, target, k, dftransition, df_time, col):
    kf = KFold(n_splits=k)
    X = df.drop(target, axis=1)
    y = df[target]
    X_new = dftransition.drop(target, axis=1).iloc[:len(dftransition)//2]
    y_new = dftransition[target].iloc[:len(dftransition)//2]
    df_time = df_time.iloc[:len(df_time)//2]

    margins = [1, 2, 3, 4, 5, 6, 7, 8]
    models = [RandomForestRegressor(n_estimators=100, random_state=42),
              SVR(),
              GradientBoostingRegressor(n_estimators=100, random_state=42)]
    model_names = ['Random Forest', 'SVM', 'Gradient Boosting']

    results = []

    for i, (train_index, test_index) in enumerate(kf.split(X)):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        result = {'fold': i + 1}
```

```
22
23          for model, name in zip(models, model_names):
24              preds, mae, mse, rmse, r2 = evaluate_model(model, X_train,
                    X_test, y_train, y_test)
25              result[name] = {'preds': preds, 'mae': mae, 'mse': mse, 'rmse':
                     rmse, 'r2': r2}
26
27              for margin in margins:
28                  error_count = (abs(preds - y_test) > margin).sum()
29                  result[name][f'errors_{margin}'] = error_count
30
31          results.append(result)
32
33      return results
34
35
36  def evaluate_model(model, X_train, X_test, y_train, y_test):
37      model.fit(X_train, y_train)
38      preds = model.predict(X_test)
39      mae = mean_absolute_error(y_test, preds)
40      mse = mean_squared_error(y_test, preds)
41      rmse = sqrt(mse)
42      r2 = r2_score(y_test, preds)
43      return preds, mae, mse, rmse, r2
```