INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Sistema Inteligente de Manutenção Preditiva

DIOGO ALEXANDRE VASCONCELOS SANTOS
Outubro de 2021

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

An Intelligent Predictive Maintenance System

Diogo Santos

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Information and Knowledge Systems

Supervisor: Goreti Marreiros
Organization Supervisor: Regina Correia

Evaluation Committee:
President:

Members:

Porto, October 14, 2021

Abstract

Maintenance Tasks in a shopfloor are one of the most critical tasks regarding the direct effect on production costs and, consequently, profit. Up until now, maintenance tasks were based on both Run-To-Failure and Reactive paradigms, fixing a machine only when it breaks or at a regular time intervals, regardless of the assets needed the maintenance or not.

However, with the Industry 4.0 Paradigm and the Smart Factories concept, machines are now equipped with sensors that monitor a large number of different and varied variables which are afterwards stored. This data can be used to predict machine failures, called Predictive Maintenance, with the aid of the manual registries of asset breakdowns.

This project, carried out in the scope of the subject TMDEI of the Master in Informatics Engineering (MEI), aims to conceive and build a system capable of doing Predictive Maintenance, by combining sensors and manual inputted data on ERP systems.

PrediMain employs different Machine Learning techniques, with a special emphasis on Ensemble Methods, making the generated machine learning models more robust and accurate, by not using a single algorithm for the predictions. For sensor predictions, before classifying them as failure or not, PrediMain uses the auto-ARIMA technique, being an auto-paremetrized method generating more accurate predictions. In the end, the system correctly classifies a set of observations with an estimated 90% of accuracy.

This system is also developed to be served as a Software-as-a-Service, allowing multiple Data Sources, and therefore shopfloors, to use the same software instance, consequently not compromising the performance of the existing systems.

Keywords: Predictive Maintenance, Industry 4.0, Machine Learning, Ensemble Methods

Resumo

As tarefas de manutenção, num contexto de chão de fábrica, são uma das tarefas mais criticas relativamente ao efeito direto nos custos de produção e consequentes lucros. Tradicionalmente, estas tarefas eram baseadas em tecnicas rudimentares, seja a manutenção quando a maquina tem uma avariar ou então manutenções regulares no tempo, independentemente de a máquina necessitar ou não.

No entanto, com o paradigma da Indústria 4.0 e *Smart Factories*, as maquinas estão cada vez mais equipadas com sensores que monitorizam um grande conjunto de variáveis e estatísticas que posteriormente são guardadas. Estes dados, em conjunto com os dados introduzidos manualmente nos sistemas ERP e MIS dos chão-de-fábrica, podem ser utilizados para prever falhas, utilizando técnicas de Machine Learning.

Este projecto, PrediMain, desenvolvido no âmbito da unidade curricular TMDEI, do Mestrado de Engenharia Informática (MEI), tem como objectivo conceber um sistema capaz de realizar Manutenção Preditiva, dando previsões ao departamento de manutenção de quando é que uma determinada máquina irá ter algum tipo de falha.

O PrediMain, tem como suporte técnicas de machine learning, com especial ênfase em técnicas de *Ensemble*, misturando diferentes algoritmos e técnicas, obtendo assim uma previsão mais fiável e precisa, contrariamente a utilizando apenas um tipo de algoritmo. Para a previsão dos valores de sensores, ainda antes de classificar uma determinada observação como possível falha, é utilizado um método auto-parametrizável e auto-ajustável, auto-ARIMA, gerando previsões mais fiáveis. No final, o sistema é capaz de classificar um conjunto de observações com uma taxa de acerto rondando os 90%.

Por fim, este sistema foi concebido para ser servido a partir da *Cloud*, com as fontes de dados configuráveis, dando assim uma maior flexibilidade aos potenciais utilizadores e prevenir falhas ou diminuições de performance nos sistemas existentes.

Acknowledgement

Firstly, I would like to thank my family, especially my parents and grandparents, who supported and motivated me even in the toughest moments. To them, a big thank you.

To both supervisors, Regina Correia and Goreti Marreiros, for their commitment and support which contributed in no small part to the project's success.

Finally, I express my sincere gratitude to who has crossed my path and helped me to become what I am today. To all, a sincere thanks.

Contents

List of Figures

List of Tables

List of Source Code

List of Acronyms

AES	Advanced Encryption Standard.
AHP	Analytical Hierarchical Process.
ARIMA	AutoRegressive Integrated Moving Average.
ERP	Enterprise Resource Planning.
HoQ	House of Quality.
IIoT	Industrial Internet of Things.
IoT	Internet of Things.
KDD	Knowledge Discovery in Databases.
MTTF	Mean Time to Failure.
NCD	New Concept Development Model.
QFD	Quality Function Deployment.
ROC	Receiver Operating Characteristic.
RUL	Remaining Useful Life.

Chapter 1

Introduction

In this chapter an analysis to the Context and Problem that originated this project is presented. Afterwards, the Approach and Methodology is described and finally the document structure is presented.

1.1 Context

Digitalization has been identified as one of the major trends changing society and business in the near and long term future [1]. According to the literature, and one of the main definitions, digitalization (or digital transformation) refers to "the changes associated with the application of digital technology in all aspects of human society" [1].
With the increasing market competition and the need to be distinctive and innovative, the idea of the digitalization is now omnipresent in most industries, seeming to be one of the main forces behind the company's strategic decisions [2]. The nowadays most prominent application of digitalization in industry is the one of smart manufacturing – commonly referred to as Industry 4.0. This paradigm has brought a large overhaul in the plants' operational processes and the technology and connectivity presence to the shopfloors. Among those, are the increasing machines' sensorization, mainly to the IoT (Internet of Things) concept (concept further developed in section 2.1.3). This further usage of sensors and the continuous monitoring of the shopfloor, down to the individual machines attributes, such as temperature, vibration and energy consumption, has allowed to evolve and develop new maintenance techniques and processes.

1.2 Problem

The vast generation of data within the shopfloors, and within the Industry 4.0 paradigm, is causing the organizations and industries to focus on a more datadriven perspective. However, and as of today, a large number of data acquired, digitally, in the shopfloor is still mostly ignored. Regarding maintenance operations, the majority of the production industry, especially small to medium sized companies, still relies on outdated maintenance policies and focuses on an inefficient approaches [3]. In the particular case of sensorization, while this data is used, for instance, in dashboards or stored, it never comes to an effective and efficient use of it in order to improve the machine's life, as it is the case with most industrial organizations. In fact, many of these organizations, especially the ones that are only recently investing in the Industry 4.0 area and evolving digitally, do not have the knowledge or the understanding of what advantages these, seemingly not useful, data can bring. Nevertheless,

with a wider flow of data being generated, this knowledge is gradually becoming a priority for maintenance-related decision-making processes.

Maintenance Tasks have attained critical importance for industries, being fundamental for a shopfloor's profit and the continuously running, without interruptions, of the factories. Since some of the assets can cost several thousands, and, in some cases, millions of euros, it is essential to ensure the continuous monitoring of the asset's health, in order to extend their lifetime. To do this, analysing and calculating a different number of maintenance-related variables, such as the time-to-failure and the RUL (Remaining Useful Life) metrics, of the machine is critical.

1.3 Objectives

Taking into consideration both the analysis on the concepts and existing solutions regarding Predictive Maintenance, the primary goal of this project is to study, conceive, implement and test, a Predictive Maintenance system. PrediMain has its its core three main objectives that should be the key for the project uniqueness and to produce more accuracte results.

The first main objective is to test, and use if it is proven that it helps creating better predictions, an ensembled based model. Instead of relying on classic Machine Learning approaches and algorithms, in order to improve the final models' accuracy, the system should combine multiple algorithms. A theoretical analysis and context is presented throughout this document.

The second main objective has to do with the system's supportability and easness of implementation and usage. It should allow the configuration of the datasources, typically the datasets that hold the historical data of sensor's observations, therefore making it easier to be used and configured by an inexperienced end-user, not being required technical knowledge of machine learning and data processing.

The third, and final, objective is related to the improvement of the generated models. Instead of generating a model, with a given data at a given time, the system should continuously check if the model should be updated having in consideration new data that has been produced and sent to the system. Consequently, it is ensured that the model is constantly updated, creating accuratemore accurate results.

1.4 Approach

In order to understand the concepts, challenges and the current state of the art for conducting this project, the first step was to understand the base concept that underlines PrediMain, the "Maintenance" concept. The different strategies of maintenance were analysed, including the predictive one.

The next step, was to analyse the machine learning concepts and algorithms in general, and specifically the ensemble techniques, by also searching existent approaches and papers in this area. Finally, a study was conducted on existing commercial applications for predictive maintenance.

Finally, and regarding the used dataset, while the proposed system is intended to have configurable data sources, for connections with multiple shopfloors, for the system development and evaluation a literature data set was used, and explained in section 4.1.3.

1.5 Document Structure

In the first chapter, Introduction, it is explained the overall problem and objectives of the current project and also the document structure.

Afterwards, in the State of the Art chapter, it is firstly presented an overview on the Predictive Maintenance concept, along with information on its background and existent systems in this area. In a second part, it is presented an overview of the Machine Learning concept, with a deeper study and explanation on the concepts that are most important for this context, namely the Supervised Learning and Ensemble Methods.

In Chapter 3, the Value Analysis, the value analysis process conducted during the project is described.

As for the Chapter 4, the Requirements Analysis as Design are presented. Regarding the Design it is also presented the methodology used for the technology selection regarding the model building and processing module.

In Chapter 5 the System's Implementation process is described, together will all the design and implementations decisions that were taken.

Chapter 6 presents a practical scenario where the dataset is used to evaluate the system's performance and accuracy, describing all the steps and results in each of the stages.

Finally, in Chapter 7, conclusions are withdrawn from the project, together with an analysis on the project's future work.

Chapter 2

State of the Art

In this chapter, a state of the art analysis is presented on the main concepts that are approached in this thesis. It is started by presenting an overview of Maintenance Strategies, ending with the Predictive Maintenance, and Machine Learning techniques that can be applied in the system.

2.1 Maintenance Strategies

Within the manufacturing industry, the efficiency, correctness and timeliness of maintenance decisions is often beyond the skill set of a human operator to perform to a satisfactory standard. This leads to the requirement of maintenance knowledge as guidelines on how to complete more informed maintenance tasks [4]. Industrial and process plants typically employ two types of maintenance management: reactive maintenance (sometimes also referred in the literature as *run-to-failure*) or preventive maintenance. However, a somehow recent approach has emerged with the Industry 4.0 and Smart Manufacturing concepts, called Predictive Maintenance.

2.1.1 Reactive Maintenance

With the Reactive Maintenance approach, the premise is simple: a plant using run-to-failure management does not spend any money on maintenance until a machine or system fails to operate. Therefore, the machine is used to its limit, and repairs are only performed when a machine actually stops working. This method is also the most expensive method within maintenance management techniques. This is due to the fact that there is a high risk of making serious damage to expensive parts of a machine, for instance. However, few plants use exclusively this approach, performing basic maintenance operations such as lubrication and machine cleaning, even though the run-to-failure is the premise in the plant environment [5].
The major expenses associated with this type of maintenance management are, amongst others, high spare parts, high machine downtime, and low production availability, since no attempt is made to anticipate maintenance requirements. Figure 2.1 represents this maintenance approach in a graphical and simplified form.

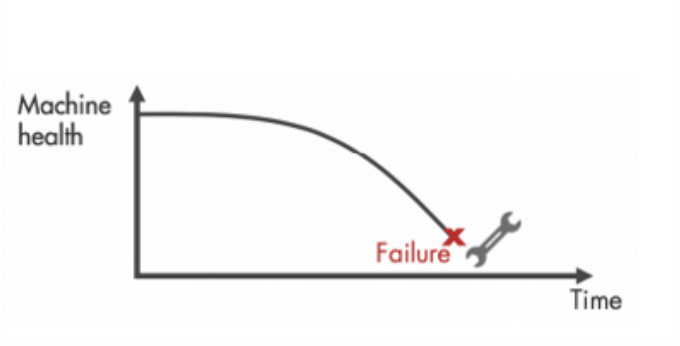


Figure 2.1: Simplified view of the Reactive Maintenance Management approaches. Retrieved from [6]

.

2.1.2 Preventive Maintenance

Preventive Maintenance, however, has a different premise: maintenance tasks are based on elapsed time or hours of operation, therefore being time-based approaches. In preventive maintenance management, machine repairs or rebuilds are scheduled based on the MTTF (Mean Time to Failure) statistic, where it is tried to prevent a machine failure by performing regular checks on their equipment. This type of approach has a few advantages, starting by a controlled forecast and needs on spare parts, since maintenance is done on a regular basis. However, when the maintenance operations is scheduled earlier than needed, this wastes part of the machine lifespan, increasing costs in the long-term. Another downside is that preventive maintenance actually prevents learning: since the equipment maintenance tasks are carried out with certain periodicity, they do not allow the depreciation or wear of the pieces of the equipment to be exactly determined. Figure 2.2 represents this maintenance approach in a graphical and simplified form.

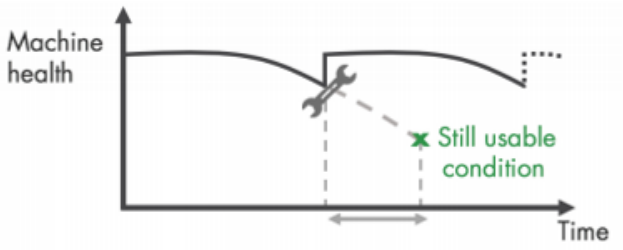


Figure 2.2: Simplified view of the Preventive Maintenance Management approaches. Retrieved from [6]

.

2.1.3 Predictive Maintenance

The maintenance has attained critical importance for industries, due to the growth in complexity of the interactions between different production activities in increasingly extended

manufacturing ecosystems. Two concepts that add value to the improvement of maintenance methods are both the Internet of Things (IoT), more specifically the Industrial Internet of Things (IIOT), and the Smart Factories [7].
In these scenarios, automated systems and equipment, internal logistics systems as as the ERP (Enterprise Resource Planning) systems, and operating supplies are consistently inter meshed with help of cyber technology, such as wireless and wireline communication services, smart actuators, and, most importantly for the improvement of Maintenance Processes, sensors. Smart Factories, amongst mass customization, Flexibility, and Optimized Decision-Making, also comprises the creation of values from the collected data, which can be used to understand the machine's behavior through different periods [7, 8]. Data is the key to this generation of information that is the base for making predictive decisions.

Predictive Maintenance can be seen as an continuous monitoring to avoid system breakdown, which will lead to maximize the time interval between consecutive maintenance tasks and reduce the overall production costs [9]. Another possible definition is a set of activities that detect changes in the physical condition of equipment (signs of failure) in order to carry out the appropriate maintenance work for maximizing the service life of equipment without increasing the risk of failure [10]. This can be seen in Figure 2.3.

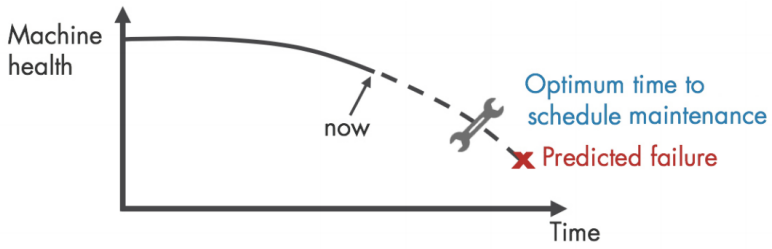


Figure 2.3: Simplified view of the Predictive Maintenance Management approach. Retrieved from [6]

.

The main benefit from using predictive maintenance systems to continuously monitor the shopfloor's machines is the decrease of maintenance costs. This is a key benefit for any industry, given that many of the equipments used in a modern shopfloor can cost several thousand euros, in some cases millions, just the purchase alone. Therefore, it is vital to ensure that these equipment's have their lifetime extended as much as it is possible. A study conducted by the consultant Deloitte, concluded that up that maintenance costs can be reduced by 5 to 10%, and, at the same time, the equipment up-time increases by 10 to 20%, reducing their breakdowns up to 70%. These values are significant, especially in cases where the equipments are, as noted, expensive [11].

While typically predictive maintenance is intended to be used solely for a maintenance management tool, as previously described to prevent downtime or failures, there are two other fields where this concept can be applied: as a plant optimization tool and as a reliability tool [5]. The Plant Optimization can be achieved by analysing cause-and-effect relationship of various modes of operation in producing several different items, establishing insights on the needs to use different assets for certain operations or even change the internal production overall procedures. As for the reliability analysis, is intrinsically connected to the benefits

mentioned above in the document: by analysing the deviations of certain variables that are causing failures, the machine's reliability can be improved, for instance, by switching unreliable components instead of just performing the scheduled maintenance task by the system.

Two main challenges are identified when analysing the Predictive Maintenance concept, one of them already mentioned in section 1.2. For any prediction problem, and therefore for also Predictive Maintenance systems, the data quality and quantity is critical for the system's capability of providing meaningful and useful outputs for the organization's management and key-members in the production and maintenance areas. Two types of data exist when referring to input data for predictive maintenance: required data and additional/optional data [11]. Required data, refers to the process failures (with the exact date and time), and the components variables, such as the temperature and vibration history. Additional data includes additional sensors, like cameras or laser sensors. The second challenge is directly related to the organizations itself, where the application of a predictive maintenance system also requires investment, not only in the system, but also in installing additional sensors and monitoring tools [11].

2.1.4 Summary and Comparison

The three mentioned approaches have all their space and usage in the industry, as shown. This also means that the three approaches have their disadvantages and advantages, including the Predictive approach which while being the seemingly better option it does have a few downsides. Table 2.1 makes a summary and comparison of these approaches, according to a set of common criteria that are common key issues regarding maintenance tasks.

Table 2.1: Comparison between the maintenance approaches

	Run-to-Fail	**Interval Based**	**Forecast**
Spare Management	Ad-Hoc	Planned	Semi-Planned
Vision Approach	Optimistic	Pessimistic	Balanced
Machine Breakdown Probability	High	Low	Low

2.2 Machine Learning

Regardless of the task at hand, any machine learning algorithm can be deployed by generally following a set of steps [12]. The KDD (Knowledge Discovery in Databases) process is one of the most used and generally accepted practices for applying machine learning to data [13].
Firstly, there is the data collection and selection phase in which, as the name suggests, involves the gathering of the data that will be used by the algorithms.
Secondly, there is the data pre-processing and transformation. Independently of the algorithm to be used, there is the need to evaluate and test the quality of the gathered data. Consequently, the second sub-step of this process is to clean the gathered data, including the removal of noise or outliers, corrupted information or other adjustments. Strategies for handling missing data fields, for instance, are also used in this step. The next step, there is what it is usually called "Model Training" (after the algorithm is selected) which involves providing the algorithm with training data to learn from. The training data must contain the correct answer to the problem, known as target. The algorithm shall then find patterns

in the data that maps the input data to the target, and provides a model that captures and contains those discovered patterns.
Sub sequentially, the Model Evaluation step takes place. Every future instance will have unknown target values, therefore there should be a test to the accuracy of the Model for which the target answer is already known. With this, there can be an estimation and prediction on the accuracy of the algorithm's performance on future data.
The final step is an optional but often applied which is the Model Improvement step. Based on the Model Evaluation results, there may be the need to use more advance strategies and methods to augment the model's performance, ranging from supplying more data to the training process or even switching to a different model completely.
This steps, and specifically the KDD process, are represented in figure 2.4.

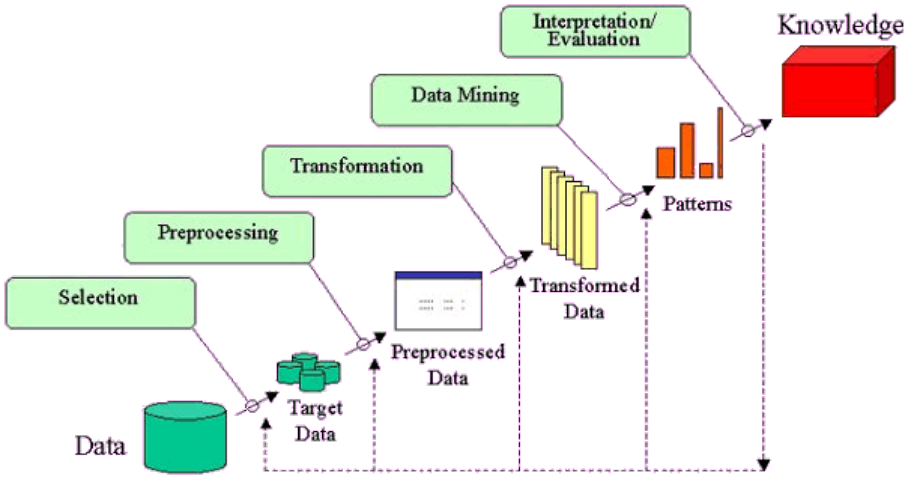


Figure 2.4: An outline of the steps of the KDD Process. Retrieved from [14]

.

Since learning involves an interaction between the learner and the environment, the learning tasks are often divided according to this interaction [15]:Descriptive Tasks (Unsupervised Learning) and Predictive Tasks (Supervised Learning).

The Unsupervised Learning is briefly introduced and explained in section 2.2.1 while the Supervised counterpart is explained with more detail in section 2.2.2, since this work is related with Predictive Tasks instead of Descriptive ones.

2.2.1 Unsupervised Learning

In unsupervised algorithms, all the items in the dataset are equally important having no distinction between the training and test set. This happens since the target values are unknown from the beginning. Consequently, the primary function of these type of algorithms is to discover hidden patterns or data groupings by themselves [12].

Clustering is a typical example of unsupervised learning in which subsets of similar objects are inferred from the set. The premise of a determined Cluster is that items inside a cluster should be similar to each other, but significantly different from those outside. The usual output of a clustering algorithm comprises a scatter plot, identifying the clusters/groups that are formed, as depicted in Figure 2.5.

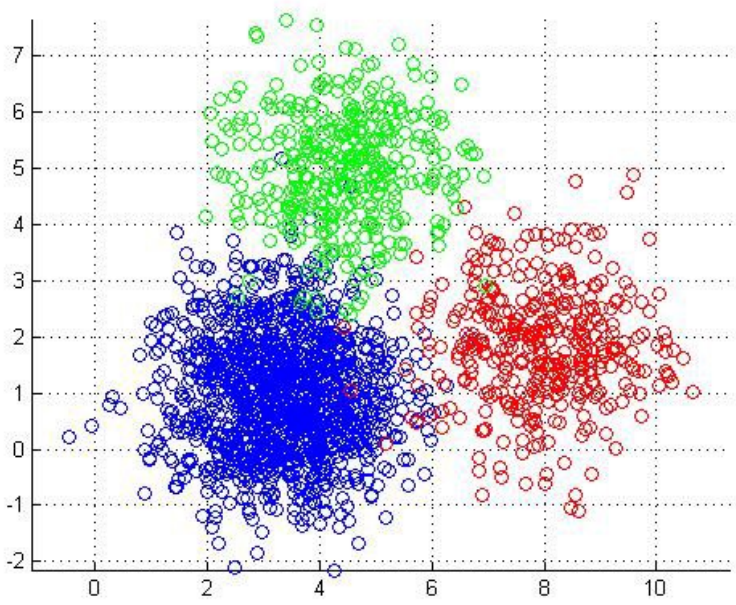


Figure 2.5: Clustering output example. Retrieved from [16]

.

The other well-known task in the field on unsupervised learning is pattern discovery/detection being employed to identify associations within data, using Association Rules, usually written in the form of:

$$\{A, B, C\} \rightarrow \{X\}$$

This states that "if A, B and C is present, then likely X will be also present". The typical example in the literature is the market basket analysis, on retailer's transactions. Here, the goal is to determine items that are often purchased simultaneously [12].

Another example of unsupervised learning has to do with Anomaly Detection techniques. One of the approaches is using a one-class classification algorithms. These algorithms attempt to model and identify the normal observations in order to classify new examples as either normal or abnormal (outliers). Given this, the training data set needs to contain "normal" observations only, in order to identify if a set of values are outliers or not. One of the approaches uses SVM (Support Vector Machines) algorithms. In this case, it models one class, so that the SVM captures the density of the majority class and classifies examples on the extremes of the density function as outliers.

2.2.2 Supervised Learning

A predictive model is used for tasks that involve, as the name implies, the prediction of one value using other values in the dataset. Because predictive models are given clear instruction on what they need to learn and how they are intended to learn it, the process of training a predictive model is known as supervised learning [12, 17]. Therefore, the training set contains a full set of the data, including the correct target values, which the test set does not have. Figure 2.6 represents a generic Supervised Learning approach.

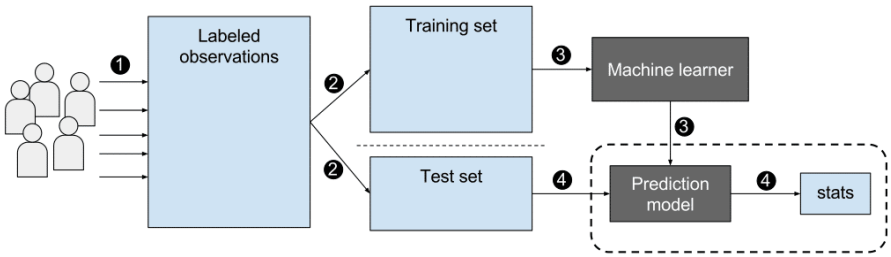


Figure 2.6: A generic approach with Supervised Learning. Retrieved from [17]

.

Using a Supervised Learning approach, there are two main areas that can be solved by using this approach: Classification Problems and Regression Problems.

Classification

Cambridge Dictionary defines "Classification" as "the act or process of dividing things into groups according to their type" [18]. Classification in Machine Learning is not different: it is the task of predicting which category an example belongs to. The usual example for classification is to predict whether an email is spam or not. There are many different Classification algorithms, such as Decision Trees, Naive Bayes and KNN (k-Nearest Neighbor).

The **Decision Trees** method "comprises a series of logical decisions, similar to a flowchart, with decision nodes that indicate a decision to be made on an attribute" [12]. These split into branches that indicate the decision's choices, eventually terminating with leaf nodes [12]. Figure 2.7 is an abstract representation of a generated decision tree for a particular problem.

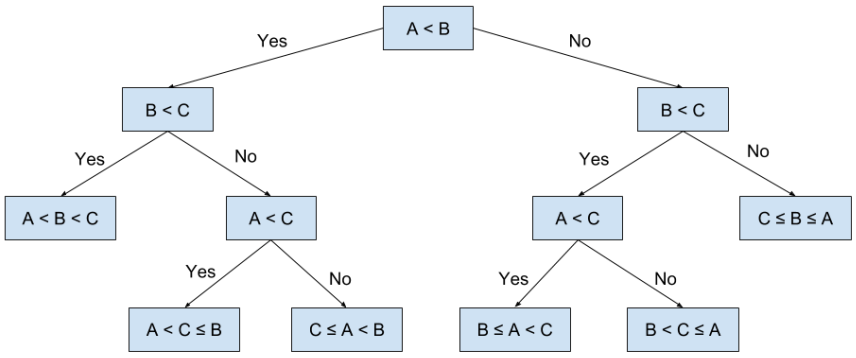


Figure 2.7: A generic representation of a Decision Tree. Retrieved from [19]

.

The tree is constructed using a divide-and-conquer method, top to bottom. The identification of which feature to split in a given branch, and the values to be split on, is a key challenge of decision tree's algorithms. The C5.0 algorithm, for instance, uses the "Entropy" concept, which "indicates how mixed the class values are; the minimum value of 0 indicates that the sample is completely homogeneous, while 1 indicates the maximum amount of disorder" [12].

Besides the already mentioned C5.0 algorithm, C4.5 (the C5.0 predecessor), and CART are amongst the most used algorithms for implementing decision trees based classification [12, 20].

Naive Bayes method is, as the name implies, based on the Bayes Theorem. Bayes Theorem describes the probability of an event, taking into account previous knowledge of conditions that might affect this event. Mathematically is stated as a conditional probability:

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B)} = \frac{\Pr(A \cap B)}{\Pr(B)} \tag{2.1}$$

Finally, **KNN** uses the concepts of distance and closeness to classify new instances, placing them in a *n-axis* plot. Since KNN uses distances, and therefore numerical values, a normalization is almost mandatory to make the algorithm's outputs reliable. Each of the axes corresponds to each of the dataset's features. To predict a new instance, the algorithm, typically calculates the euclidean distance between that instance and every instance already in the model, choosing the K (a parameter) nearest data point. This can be seen in Figure 2.8.

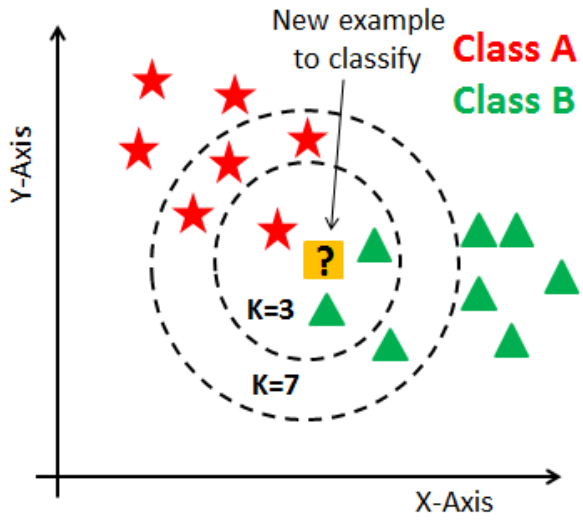


Figure 2.8: A generic representation of the KNN Algorithm. Retrieved from [21]

.

Regression

Unlike Classification, Regression predicts a continuous value. The most simple form of Regression is the **Linear Regression**, known as multiple regression when more than two independent variables are used [12]. The simplest form of Regression, using only one predictor, uses the following equation, representing a straight line:

$$y = ax + b \tag{2.2}$$

Figure 2.9 represents a linear regression (in red) for a set of data points.

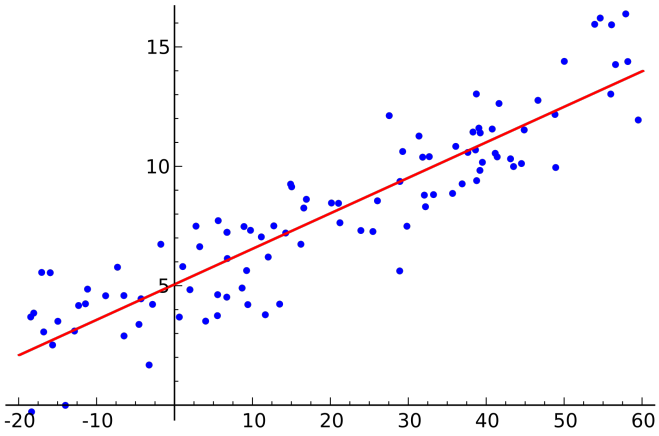


Figure 2.9: Linear Regression. Retrieved from [22]

.

In order to estimate the equation to be used, a different estimators and functions are available, being the most simple the Linear Least Squares method, using the following equation:

$$\Sigma(y_i - \dot{y}_i)^2 = \Sigma e_i^2 \tag{2.3}$$

This method, as the name implies, calculates the best approximation, minimizing the error e that minimizes the sum of squared differences between the data values and their corresponding modeled values, as represented in Figure 2.10.

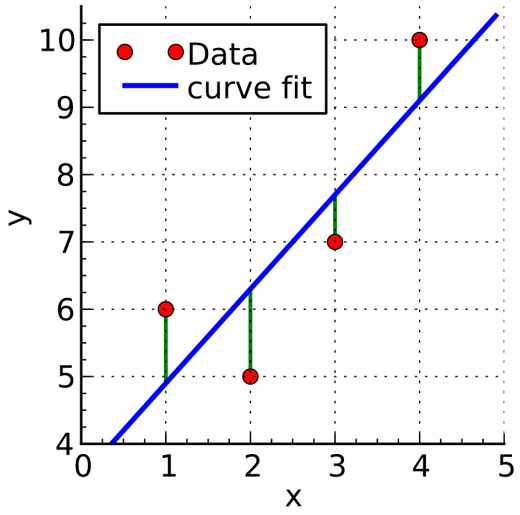


Figure 2.10: Linear Least Squares method. Retrieved from [22]

.

2.2.3 Ensemble Techniques

Ensemble Techniques (also called Ensemble Methods, or Ensemble Learning) represents the combination of multiple predictions from several base estimators "built with a given learning algorithm in order to improve generalizability/robustness over a single estimator". This way, the prediction made should be of a higher accuracy than the prediction of any of the constituent algorithms by itself. Ensemble Methods are usually split in three large families: Bagging Methods, Boosting Methods, and Stacking Methods.

In ensemble algorithms, **Bagging** Methods form a class of algorithms which build several instances of an estimator, on random subsets of the original training set, and then aggregate their individual predictions to form a final prediction. The objective and premise, is that the combined estimator performs better than any of the single base estimator because its variance is reduced. An example of this, is the generation of multiple decision trees, each with a subset of the training data. After each Decision Tree computes its prediction, all results are aggregated, forming the theoretically most precise prediction. This is represented in Figure 2.11.

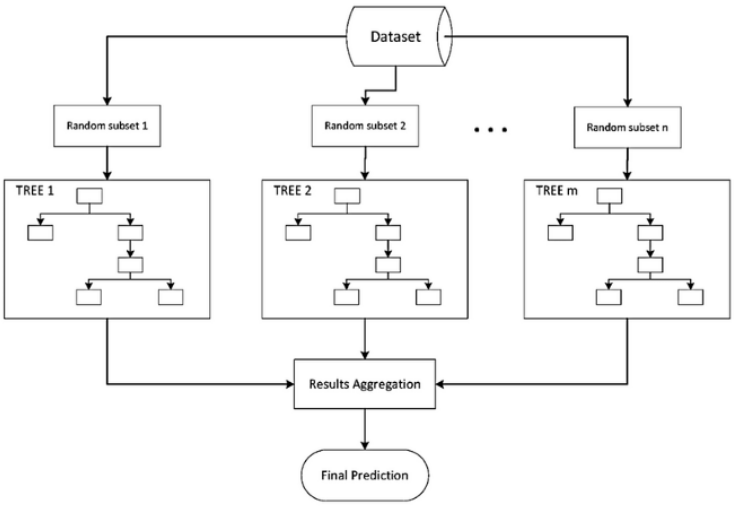


Figure 2.11: Bagging method using Decision Trees. Retrieved from [23].

A small, but important, difference to this method is used in Random Forests methods. While in the previous case, all of the Decision Trees had the full set of features for the given sample, Random Forest models decide where to split based on a random selection of features. Rather than splitting at similar features at each node throughout, Random Forest models implement a level of differentiation because each tree will split based on different features [23], that are assigned to each tree.There is also another variation of this method, called Extremely Randomized Trees, where the random factor is also present on the splitting rules in the nodes. In this case, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule.

As for the **Boosting** algorithms, "base estimators are built sequentially and one tries to reduce the bias of the combined estimator". In this case, the idea is to combine several weak models to produce a powerful ensemble. Since the models are built sequentially, a given model depends on the previous one. Once the first model is built, the falsely classified points are taken in addition to the second bootstrapped sample to train the second model. Afterwards, the ensemble model is used against the testing dataset, and the process continues [24]. This is represented in figure 2.12.

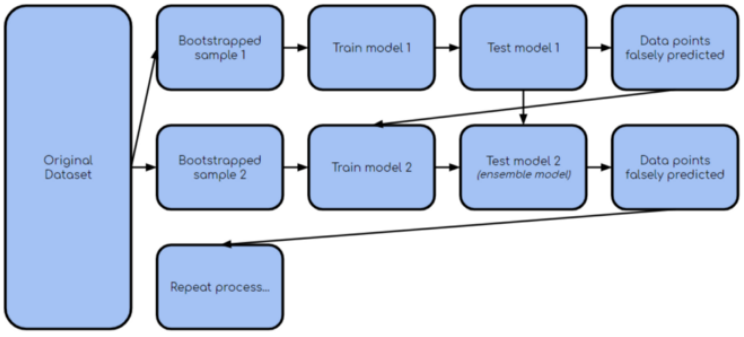


Figure 2.12: Boosting method example. Retrieved from [24].

Adaboost, named after "adaptative boosting", implementations follows this methods, in with the objective is to "fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data".
There is another type of boosting implementations, called Gradient Boosting, where the main difference is in the definition of the sequential optimisation process, in which gradient boosting casts the problem into a gradient descent one. While the AdaBoost model identifies the shortcomings by using high weight data points, gradient boosting performs the same by using gradients in the loss function [24, 25]. This means that "at each iteration we fit a weak learner to the opposite of the gradient of the current fitting error with respect to the current ensemble model" [24].

Table 2.2 summarizes the key differences between these two types of Ensemble Learning.

Table 2.2: Comparison between Bagging and Boosting methods

	Bagging	**Boosting**
Models	Independent Models	Dependent Models
Method for making the final decision	Equally Weighted Average	Weighted Average
Deals with over-fitting issues	Yes	No (can actually increase it)
Reduces Bias	No	Yes

Finally, **Stacking** is drastically different from Boosting and Bagging. Unlike bagging, in stacking, the models are typically different, and unlike boosting, a single model is used to learn how to best combine the predictions from the contributing models (instead of a sequence of models that correct the predictions of prior models) [26, 27]. Figure 2.13 represents a typical stacking-based approach model architecture.

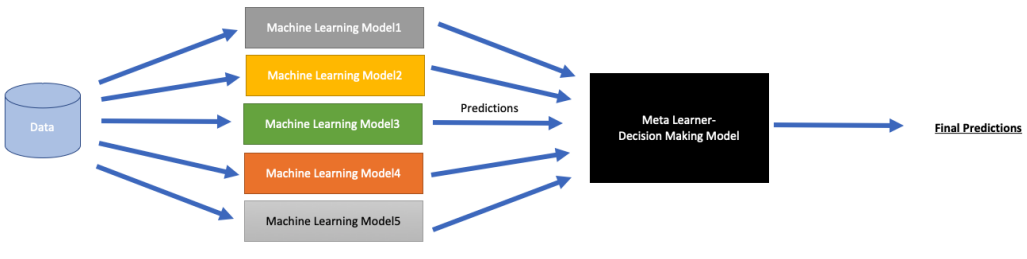


Figure 2.13: Stacking method architecture. Retrieved from [26].

The base level, usually called level-0, contains all the base-models that will compose the stacking model, who will individually classify the training dataset. A second level, called level-1, will hold the meta-estimator model, which is the model that learns how to best combine the predictions of the base models.

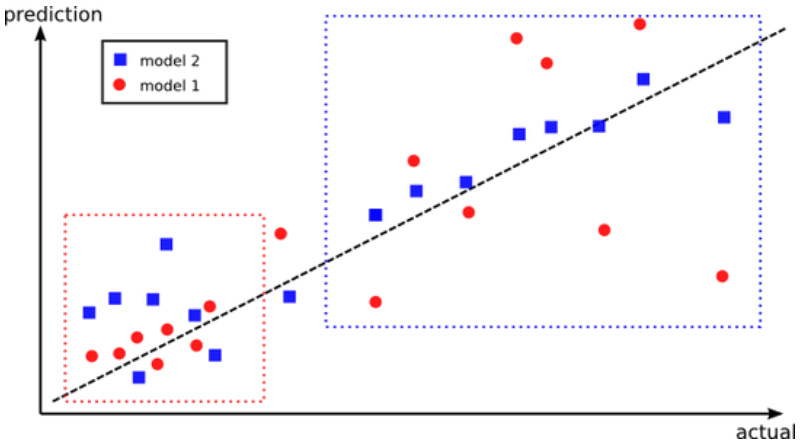


Figure 2.14: Stacking method advantage example. Retrieved from [27].

Figure 2.14 is an example that demonstrates the advantages on using a stacking approach [27]. In this case, for instance, model 1 has a lower training accuracy than model 2 however in some data points model 1 performs better. Using a Regression meta-model, the best out of the two algorithms is combined into a single one. Usually Regression problems have an Linear Regression as its meta-model and Classification problems have a Logistic Regression meta-model.

2.2.4 Algorithms Performance Evaluation

The Model Evaluation is perhaps one of the most important steps of any Machine Learning process. If a given model is not evaluated properly, it will likely result in poor predictions made in the future. An evaluation process, usually comprises of using the Holdout Method: two thirds of the dataset is used for the Model Training, and the rest is used to validate the model's predictions against the known values.

The base evaluation metric, for supervised algorithms, is called a confusion matrix, where the rows represents the real values and the columns represents the predicted values. This matrix is represented in Figure 2.15.

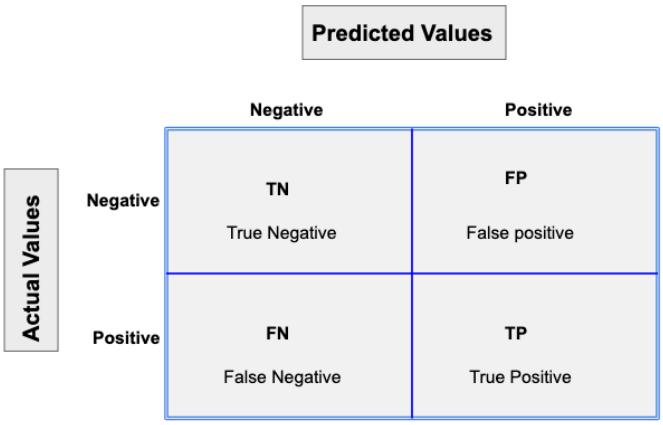


Figure 2.15: A Confusion Matrix. Retrieved from [28].

Each predictions is then classified based on four different outputs:

1. True Positive (TP): Predicted True and True in reality.
2. True Negative (TN): Predicted False and False in reality.
3. False Positive (FP): Predicted True and False in reality.
4. False Negative (FN): Predicted False and True in reality.

With this is mind, two types of miss classifications can therefore occur, False Positives and False Negatives. Along with these four outputs, a large number of rates and statistics can be inferred from a confusion matrix.

The **Accuracy** rate, measures the " how often is the classifier correct". This is given by the following formula:

$$Accuracy = \frac{(TP + TN)}{total} \tag{2.4}$$

The **Precision** rate is slightly different, measuring how often the classifier predicts correctly when it predicts as being positive, being given by the following formula:

$$Precision = \frac{(TP)}{FP + TP} \tag{2.5}$$

Another rate that can be inferred from the confusion matrix is the **Recall** statistic, which indicates the True Positive Rate (TPR):

$$Recall = \frac{(TP)}{FN + TP} \tag{2.6}$$

Finally, it is also common to relate the Precision and Recall metrics using the **F1-Score**, also called F-Measure:

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{2.7}$$

This metric uses the harmonic mean to combine these two metrics. The harmonic mean is used rather than the more common arithmetic mean since both precision and recall are expressed as proportions between zero and one [12]. The main issue with this metric is that is assumes that "equal weight should be assigned to precision and recall, an assumption that is not always valid" [12].

A different method, that examine the tradeoff between the detection of true positives, while avoiding the false positives, is called **ROC Curves** analysis. Figure 2.16 represents a sample ROC Curve diagram, where the dashed line represents "random guessing".

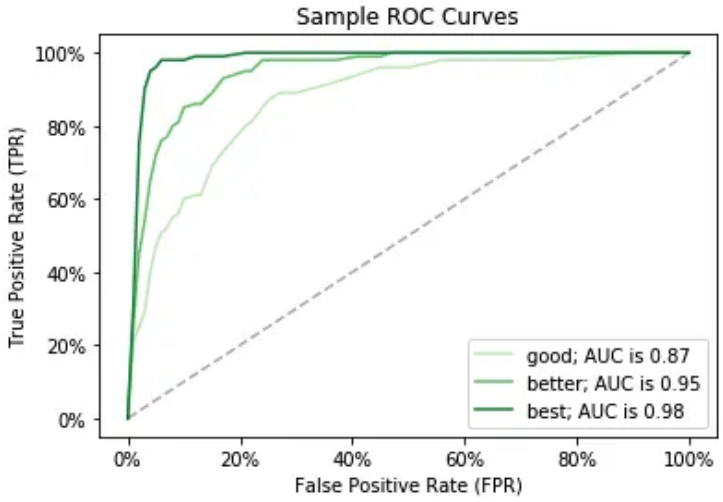


Figure 2.16: Sample ROC Curve. Retrieved from [12].

The closer the curve is to the chart axis, the better the prediction value of the model is. This can be measured using a statistic known as the area under the ROC curve (AUC), where it ranges from 0.5 (for a classifier with no predictive value, representing the curve overlaying the dashedline), to 1.0 (for a perfect classifier) [12].

2.3 Existing Predictive Maintenance Approaches and Systems

Currently, there are a significant number of existing machine learning approaches to tackle the predictive maintenance problem.

Firstly, an approach by IEEE members was taken, using Multiple Classifiers in a semiconductor manufacturing industry, specifically replacing tungsten filaments used in ion implantation [29]. In this scenario, approaches using KNN and SVM (Support Vector Machines) were tested and compared. Instead of only labeling the last iteration of a maintenance cycle as F (Failure) it labels as F the last m iterations, allowing to provide more conservative maintenance recommendations by choosing larger values for the failure horizon m. Figure 2.17 represents the different analysed techniques and Figure 2.18 represents the gotten results.

Type	Based on:	Acronym
PvM	Mean μ	PvM-μ
	Median η	PvM-η
PdM	Linear SVM	PdM-lin
	Gaussian Kernel SVM	PdM-rbf
MC PdM	k-NN	MC PdM-knn
	SVM	MC PdM-svm

Figure 2.17: Tested Techniques. Retrieved from [29].

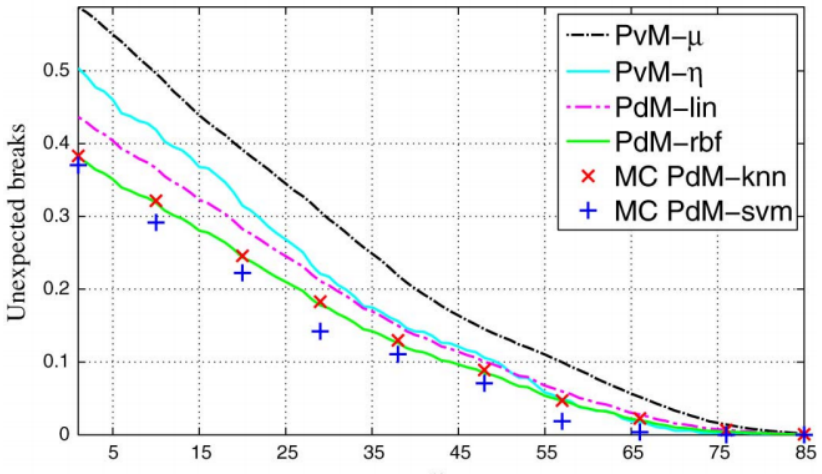


Figure 2.18: Results given for each of the applied techniques. Retrieved from [29].

A second approach that was found, was conducted by a group of investigators of Sweden and Italy, mainly by using the Azure Cloud systems, applying predictive maintenance on CNC machines [30], using features such as the spindle rotation speeds and the speed on all three (X, Y and Z) axis. In this case, a Random Forest classifier was used, using the Bagging method. The input data was comprised of drive data (sampled in real time by the CNC, data related to I/O signals and vibrational data. All the collected data, using the proposed architecture were evaluated using the 30% of the data set as a training set and the rest for the results evaluation. The final system had an high accuracy (95%) on a data set of 530731 data readings on 15 different machine features collected in real time from the tested cutting machine. The results table is presented in Table 2.3.

Table 2.3: Classification Results of the System. Retrived from [30]

Metric	**Results**
Overall Accuracy	0.95
Average Accuracy	0.92
Micro-Averaged Precision	0.94
Macro-Averaged Precision	0.93
Micro-Averaged Recall	0.95
Macro-Averaged Recall	0.94

A third approach was conducted by the GECAD research department, of the Instituto Superior de Engenharia do Porto (ISEP), and the BISITE Research Centre. Data collected from monitoring four CNC machines in a mechanical metallurgy factory was used, with the objective of the prediction of faults. In this study, since data concerning problems in the machines was not available, and therefore the prediction of faults was framed as an anomaly detection problem (unsupervised learning), using classical time series methods to build forecasting models [3]. Specially, autoregressive integrated moving average models (ARIMA) were used, in order to model the autocorrelations in the data.

Given this, for each feature, it was compared the new data with the respective 95% prediction interval, calculated by the ARIMA algorithm. If a previously unseen value falls outside the prediction interval it is considered an anomaly, as represented in figure 2.19. However, since an anomaly doesn't have much significance by itself, it's the accumulation of anomalies over a given time period that indicated the possibility of a malfunction [3].

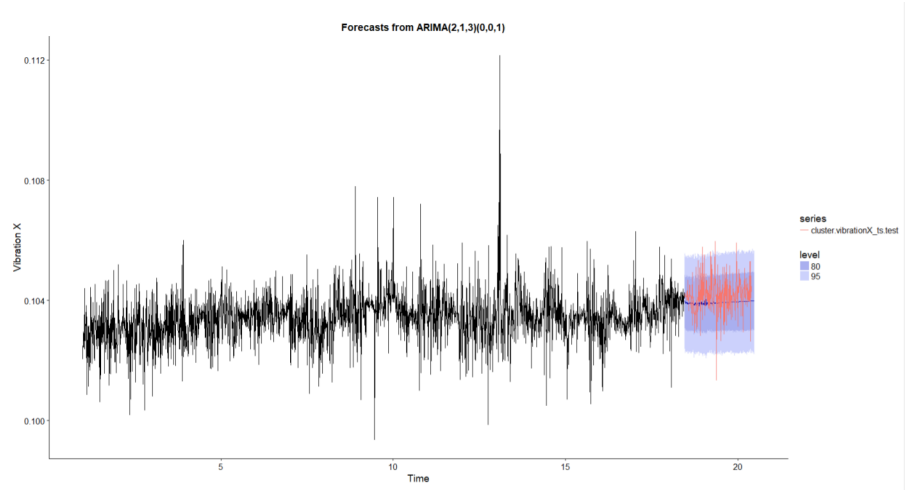


Figure 2.19: Generated ARIMA model, and prediction interval, for the vibration feature. Retrieved from [3]

Regarding the existent predictive maintenance systems in the market, there are still no developed and established solutions for Predictive Maintenance. Two solutions were found that are the most complete in the field, amongst the analysed solutions, capable of actively doing predictive maintenance as the concept is stated in the literature review.

The first potential solution that was found is MindShpere, by Siemens. MindSphere is stated to be an IIOT as a Service. The MindSphere stores and collects data provided by the "MindSphere Applications", including data collected from the assets [31]. This collected can be done independently from the asset's manufacturer since it includes support for the OPC Foundation's OPC Unified Architecture (OPCUA). This solution also contains a root cause analysis protects asset against permanent damage arising from failures [32].

The second, is SAP Predictive Asset Insights. This solution uses Digital Twins and Machine Learning to maintain full visibility into current asset health and predict future needs. Amongst others, the main features of this product are:

- Enrich asset management to remotely monitor assets anywhere
- Detect system malfunctions using real-time fault management
- Improve quality and service by predicting malfunctions before they cause unscheduled downtime and higher costs
- Offer performance-based service and dispatch proper technical assistance based on near-real-time monitoring of operations

While this solution is a cloud-based system, it runs integrated with a base SAP environment only, not being available as a completely stand-alone application.

In conclusion, although the concept has been extensively studied and developed, there are still no common and publicly disclosed commercial solutions, apart from the mentioned systems, that employs predictive maintenance throughout its full concept, at an affordable and with seamless installation for a standard shopfloor that is just entering in the Industry 4.0 concept.

Chapter 3

Value Analysis

3.1 New Concept Development Model

"The New Concept Development Model (NCD) provides a common language and definition of the key components of the Front End of Innovation" [33]. In this Model, the engine, the central component, represents senior and executive-level management support, and powers the five other elements of the model. The model is represented in figure 3.1.

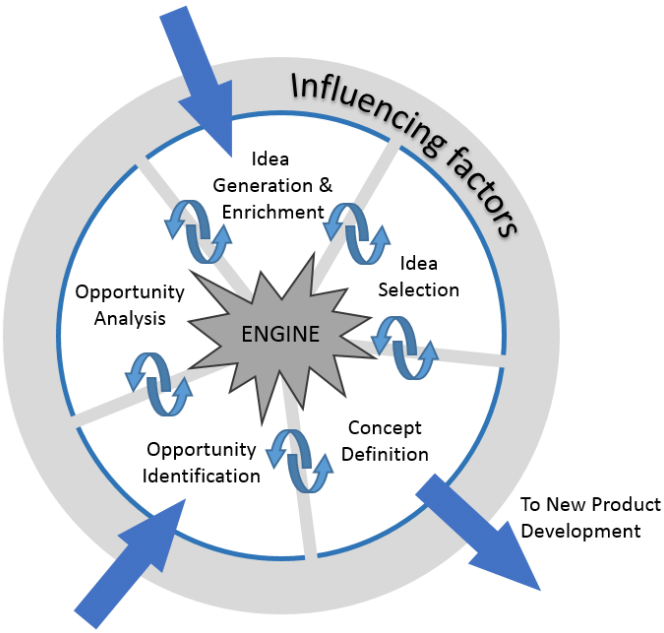


Figure 3.1: NCD Model. Retrieved from [34]

.

The following is a characterization of each of the five elements that are present in the NCD model [33, 35]:

Opportunity Identification - Large or incremental business and technological chances and opportunities are identified, by design or default, in a more or less structured way.

Opportunity Analysis - Gathering together the additional information required in order to translate the identified opportunities into specific business and technology opportunities for the company.

Idea Generation & Enrichment - Birth, development and maturation of the opportunity into a concrete idea.

Idea Selection - Choose which ideas to pursue in order to achieve the most business and consumer value. The activity of prioritizing and selecting ideas may be based on an individual's choice or a comprehensive portfolio planning approach.

Concept Definition - Business case is developed, based on estimates of the other activities; market potential, customer needs, investment requirements, competition analysis and project uncertainty.

Given the referred NCD Model, and its components, the following elements were identified in the context of the this document:

- Opportunity Identification - Maintenance Management within shopfloors is evolving towards a data and technology oriented paradigms, diverging from the "run-to-failure" classic approaches. However, the majority of the production industry still relies on outdated maintenance policies and focuses on an inefficient run to failure approach or statistical trend driven maintenance intervals [36]. There is now a large flow of data being generated from the machines and also from the ERP/MIS systems that are now widely used in Manufacturing Industries. This identification resulted from the aim to personally explore not only this area, but also the Machine Learning processes, specifically the combination of different algorithms, making the predictions more accurate.
- Opportunity Analysis - Digitalization is now omnipresent in most industries, seeming to be one of the main forces behind the company's strategic decisions [2].Some studies report that up to 30% of the costs within a shopfloor are from Maintenance operations [37]. Reducing costs is a large priority in any organization, and therefore making the maintenance operations more efficient and cost-effective is also a priority.
- Idea Generation & Enrichment - Given the Opportunity Identification and Analysis described above, the main concepts regarding both Predictive Maintenance and Machine Learning Techniques (and specifically Ensemble Methods) were studied and analysed, in which the outcomes of this study are represented in sections 2.1 and 2.2, respectively. Secondly, a research on existing solutions that partially (or are related to) solve this issue was also conducted.

3.2 Value

3.2.1 Value for the Customer

"Value for the Customer" is usually used in marketing to represent the delivered value, enclosed in the delivered system, for the client.

In this context, the value is mainly the opportunity to optimize and predict machine failures, reducing the costs related to maintenance operations.

3.2.2 Perceived Value

Perceived value is often defined as "the customers own perception of a given product, or service, merit or desirability to them, and its ability to meet their needs and expectations" [38].

It is expectable that any predictive maintenance system is used by two main user profiles/-types: Maintenance Directors/Supervisors and the organization's high-level management. For the first, the perceive value will be highly affected by how the system presents the results. The system should present the results as simple as possible, given concise outputs. At the same time, it should present the reasons for the generated predictions, without using any technical terms (such as the underlying machine learning processing). As for the latter, the perceived value will be affected by the costs reduction regarding maintenance activities.

3.3 Value Proposition

Value propositions should not only express why your products, services, and/or solutions are better than the competition's, but also should be simple, clear, easy to absorb, adaptable to specific clients or segments, and credible [39].

In this context, this project aims to aid the top level management and/or maintenance management, of industries who are now entering, or in, the Industry 4.0 processes, to save up on maintenance costs and increasing the machine's uptime by predicting machine failures. The fact that the system is generic, providing a seamless integration with existing shopfloor management systems, also contributes for an increase in the Perceived Value.

Alexander Osterwalder proposed a Value Proposition framework [40]. This framework revolves around two larger entities – customer profile and the value map, which are visually presented in a canvas. The customer profile aims to identify the proposed system gain's (benefits which the customer expects and needs), pains (risks that the customer may experience) and the customer jobs, which represent the tasks that are trying to be done. As for the value map, it represents the Gain Creators (how the system creates and satisfies the customer's gains), pain relievers (how the product or service alleviates customer pains), and the products and services, representing what functionalities and operations the system presents [40]. The canvas for this project is represented in Figure 3.2.

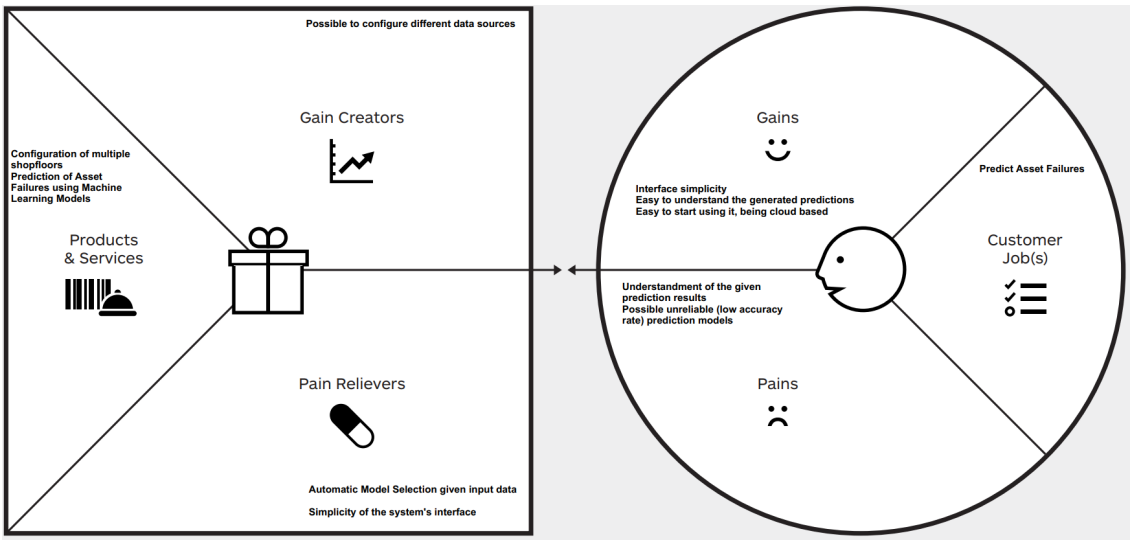


Figure 3.2: Value Proposition Canvas

3.4 Quality Function Deployment

QFD is a structured method that uses the seven management and planning tools to identify and prioritize customers' expectations quickly and effectively [41]. This method usually contains an analysis using an House of Quality (HoQ) template and methodology. This diagram answers the customer's *wows*, *wants*, and *musts*. The HoQ diagram for this project is represented in Figure 3.3.

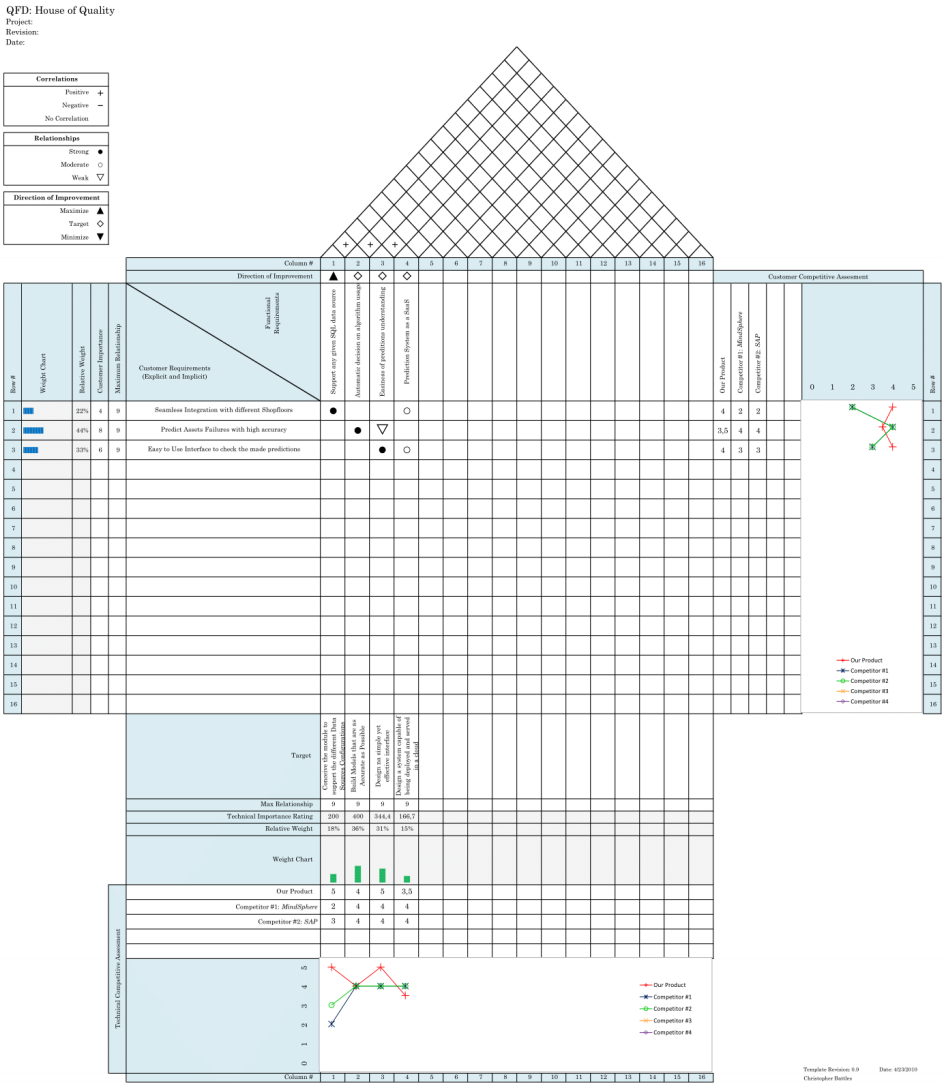


Figure 3.3: Quality Function Deployment - House of Quality of the project

Chapter 4

Analysis and Design

In this chapter the Requirements Analysis and Design Decisions are presented. The selection process of the technology for applying the machine learning models is also presented.

4.1 Requirements Analysis

The first critical issue, and requirement, for the project is the system's integrability with other systems. The solution is intended to work and integrate with different data sources, of different types with different connection properties. This is critical for the project's maintainability and usage in the long term, since usually each asset manufacturer, and sometimes even each asset, has its own mechanisms to access its collected digital data. For instance, some machines provide a connection to an internal Relational Database, while others send the sensors data to an external repository.

Secondly, the system should also work with its own database, to hold the trained model in order to only train a new one when a given time has passed since the last training (or if it is manually triggered). Therefore, the database should be prepared to store and manage non-structured data.

Finally, the system should provide an interface to check the outputted predictions. This interface, however, should be built with the premise that the users that will use it, have potentially little technical expertise, and especially regarding machine learning concepts. Therefore, the interface should be simple, concise yet showing all the information for understanding and to justify the outputted prediction.

4.1.1 Functional Requirements

After the analysis on the requirements that were gathered above, four use cases were idealized:

UC1: Manage the available Data Sources
UC2: Train the Machine Learning Model
UC3: Tunning of the Algorithm Input Parameters
UC4: Visualize the Model's Predictions

The Use Case diagram is represented in Figure 4.1.

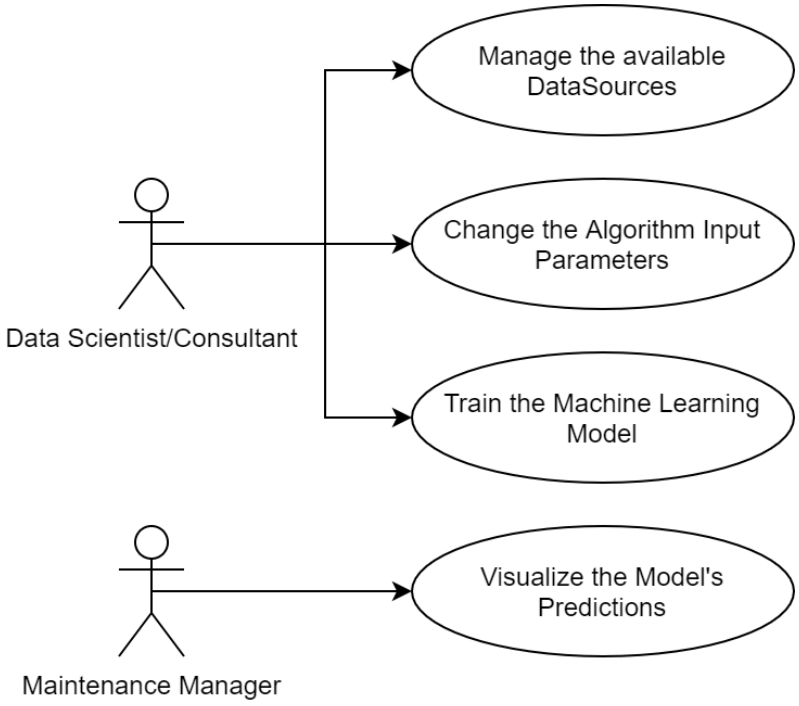


Figure 4.1: Use Case Diagram

.

UC2 represents the manual trigger of the model training process (further described in section 4.2.2). However, this training can also be started automatically. This happens when a prediction request is made, and the generated model that is stored in the database is significantly dated, regarding the data that was used for its training. In this case, the system firstly trains a new model, and only then the requested prediction is processes and shown to the user (UC4).

4.1.2 Non Functional Requirements

In order to evaluate and assess the non-functional requirements, the FURPS+ model was used. This model categorizes the non-functional requirements into 5+ categories: Functionality, Usability, Reliability, Performance and Supportability. The "plus" refers to Design Constraints, Implementation requirements, Interface requirements and Physical requirements. Table 4.1 contains the identified Non Functional Requirements, organized by their category.

Table 4.1: Non Functional Requirements

Code	Requirement	Type
NFF01	HTTPS protocol should be used in the communications between components	Functionality
NFU01	The User Interface should be simple, yet effective, and adapted to the User's needs	Usability
NFU02	The User Interface for showing the predictions that were made should not show technical data that could not possible be understood by the maintenance managers	Usability
NFP01	The Data Retrieval from the existent systems shouldn't affect their performance	Performance
NFS01	The system should support any SQL database, needing the specification of all data needed for the connection (such as IP Address, username, password and port number)	Supportability
NFS02	The system should be prepared to, in the future, accept new implementations of other types of Databases/sources besides SQL. Therefore, the code should be highly maintainable and provide an easy way to implement the new Database handlers	Supportability

4.1.3 Dataset

Usually, predictive maintenance datasets contains a standard set of features that are commonly retrieved and measured in the industrial assets, such as vibration and temperature.

As per the requirement NFS01 and NFS02, the system should handle multiple data sources, and therefore multiple datasets.

Ideally, a real dataset would be used, however it was decided that, during the system's development and evaluation stages, a single dataset would be used, retrieved from an online repository, given that no complete and accurate real datasets were available at the time. For this decision, considerations regarding security and data protection issues were also taken into account.

The chosen dataset, found in [42], was retrieved from the *Kaggle* plaform, and contains sensor and failure data for the timespan of a year. In total, 7900 observations were made, containing data on Temperature, Humidity, and another 12 sensors of unknown measurements/units. The observations had an hourly frequency. Each line is classified with a boolean value for the *Failure* column, being the target value to use with the supervised algorithms. Figure 4.2 is a snippet of the dataset's CSV that will be used:

```
ID,Date,Temperature,Humidity,Operator,Measure1,Measure2,Measure3,Measure4,Measure5,Measure6,Measure7,Measure8,Measure9,Measure10,Measure11,
Measure12,Measure13,Measure14,Measure15,Hours Since Previous Failure,Failure,?Date.year,?Date.month,?Date.day-of-month,?Date.day-of-week,?Date.
hour,?Date.minute,?Date.second
1,01-01-2016 00:00,67,82,Operator1,291,1,1,1041,846,334,706,1086,256,1295,766,968,1185,1355,1842,90,No,2016,1,1,5,0,0,0
2,01-01-2016 01:00,68,77,Operator1,1180,1,1,1915,1194,637,1093,524,919,245,403,723,1446,719,748,91,No,2016,1,1,5,1,0,0
3,01-01-2016 02:00,64,76,Operator1,1406,1,1,511,1577,1121,1948,1882,1301,273,1927,1123,717,1518,1689,92,No,2016,1,1,5,2,0,0
4,01-01-2016 03:00,63,80,Operator1,550,1,1,1754,1834,1413,1151,945,1312,1494,1755,1434,502,1336,711,93,No,2016,1,1,5,3,0,0
5,01-01-2016 04:00,65,81,Operator1,1928,1,2,1326,1082,233,1441,1736,1033,1549,802,1819,1616,1507,507,94,No,2016,1,1,5,4,0,0
6,01-01-2016 05:00,67,84,Operator1,398,1,2,1901,1801,1153,1085,1547,2005,477,1217,1632,1324,1854,1739,95,No,2016,1,1,5,5,0,0
7,01-01-2016 06:00,67,83,Operator1,847,0,2,1849,1141,1609,982,1159,672,1128,663,1114,1838,290,1192,96,No,2016,1,1,5,6,0,0
8,01-01-2016 07:00,67,76,Operator1,1021,2,1,185,170,952,1183,1329,427,1638,850,379,1529,755,844,97,No,2016,1,1,5,7,0,0
9,01-01-2016 08:00,65,80,Operator3,1731,2,0,1424,1176,1223,621,647,369,239,1196,1944,1583,1630,237,98,No,2016,1,1,5,8,0,0
10,01-01-2016 09:00,63,80,Operator3,415,0,0,1008,1086,1759,1946,1814,1754,1442,341,1097,1819,472,491,99,No,2016,1,1,5,9,0,0
11,01-01-2016 10:00,61,83,Operator3,525,2,2,603,1630,1740,583,646,319,933,1891,1371,1904,1586,1102,100,No,2016,1,1,5,10,0,0
12,01-01-2016 11:00,62,81,Operator3,1719,3,1,880,575,1722,952,890,669,782,1720,641,1620,1046,1759,101,No,2016,1,1,5,11,0,0
13,01-01-2016 12:00,62,76,Operator3,1116,0,0,588,166,1233,1362,322,1176,1669,1197,182,1641,1033,1427,102,No,2016,1,1,5,12,0,0
14,01-01-2016 13:00,60,82,Operator3,282,1,1,1406,1727,945,1631,714,1072,1051,1857,559,1309,1668,1453,103,No,2016,1,1,5,13,0,0
```

Figure 4.2: Dataset Sample

Table 4.2, represents the columns that are contained in the CSV, together with its base datatype and inferred meaning.

Column	Description	Type
ID	Sequential observation number	nvarchar
Date	Observation date	datetime
Temperature	Machine temperature value	int
Humidity	Machine humidity value	int
Operator	Unknown, inferred to be an operator (person)	nvarchar
Measure1	A given sensor (unknown) value	int
Measure2	A given sensor (unknown) value	nvarchar
Measure3	A given sensor (unknown) value	nvarchar
Measure4	A given sensor (unknown) value	int
Measure5	A given sensor (unknown) value	int
Measure6	A given sensor (unknown) value	int
Measure7	A given sensor (unknown) value	int
Measure8	A given sensor (unknown) value	int
Measure9	A given sensor (unknown) value	int
Measure10	A given sensor (unknown) value	int
Measure11	A given sensor (unknown) value	int
Measure12	A given sensor (unknown) value	int
Measure13	A given sensor (unknown) value	int
Measure14	A given sensor (unknown) value	int
Measure15	A given sensor (unknown) value	int
HoursSincePreviousFailure	Number of hours since the previous failure	nvarchar
Failure	Bit stating if the observation was of a failure	bit
Date_year	Year of observation	int
Date_month	Month of observation	nvarchar
Date_day_of_month	Day of observation	nvarchar
Date_day_of_week	Weekday of observation	nvarchar
Date_hour	Hour of observation	nvarchar
Date_minute	Minute of observation	nvarchar
Date_second	Second of observation	nvarchar

Table 4.2: Dictionary of the dataset columns

4.2 Solution Design

4.2.1 Technology Selection

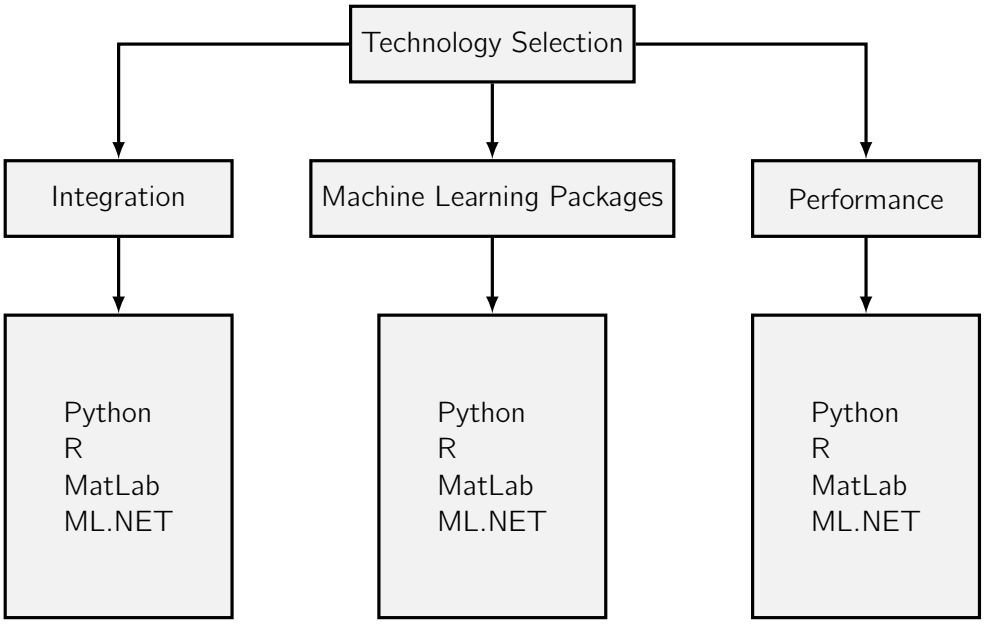
In order to choose the technology that would be used to build and process the Machine Learning models, the Analytic Hierarchy Process (AHP) method was used. AHP is a multi-criteria decision-making approach, which uses a multi-level hierarchical structure of objectives, criteria, subcriteria, and alternatives [43]. This method is composed by seven phases:

1. Build the decision hierarchical tree
2. Comparison between the hierarchical criteria
3. Define a relative priority for each criteria
4. Evaluate the priorities consistency
5. Build the comparison matrix for each pair combination of criteria
6. Calculate the composite priority
7. Decide the better alternative

In the context of this project, the following alternatives were considered: Python, R, MatLab and ML.NET. The chosen alternatives were all based, on a first criteria, on the tool's popularity index, but the ML.NET, which was selected given the partnership of Sistrade with Microsoft, allowing the usage of this tool. As for the criteria to be analysed, they were the following:

- Machine Learning Packages - This criteria is based on the amount, and quality, of the Machine Learning packages/libraries that each tool provides, or has available for use.
- Integration - The Machine Learning Predictor, after the model building, should be easily served over a common server, Windows or Linux based.
- Performance - Relates to both the tool's general performance, for instance in processing the input data, and the performance in building the machine learning model(s), since they should be re-built in a regular basis, using the continuous flow of shopfloor data.

The following diagram presents the relative priorities for each criteria and the given alternatives.



The next step was to build a matrix containing the relative priorities by each pair of criteria. This matrix is represented in table 4.3.

Table 4.3: Relative Priorities between criteria using AHP method

	Machine Learning Packages	**Integration**	**Performance**
Machine Learning Packages	1	7	2
Integration	1/7	1	1/5
Performance	1/2	5	1
Sum	23/14	13	16/5

In the next step, step 3 of the AHP methodology, a normalized matrix was built, by diving each cell value by the sum of the respective column. Afterwards, a new column was added with the average of each row. This column represents the final relative priority for each criteria. This matrix is represented in table 4.4.

Table 4.4: Normalized Relative Priorities between criteria using AHP method

	Machine Learning Packages	**Integration**	**Performance**	**Relative Priority**
Machine Learning Packages	14/23	7/13	10/16	0,59
Integration	14/161	1/13	5/80	0,08
Performance	14/46	5/13	5/16	0,33

Afterwards, the Consistency Ratio (CR) was calculated, in order to measure how consistent the judgments have been relative to large samples of purely random judgments. The first step involved the calculation of the Consistency Index (CI), which can be found in A. In this case, the calculated CI was 0.02. Since $0.02 < 0.1$, the priorities are consistent. Given this, the next step is to build the comparison matrices, for each of the alternatives, one for each

criteria. This process is the same as the one made for the criteria matrices. The following tables are the comparison matrices with the extra "relative priority" column.

Machine Learning Packages:

Table 4.5: Machine Learning Packages comparison matrix

	Python	**R**	**MatLab**	**ML.Net**	**Relative Priority**
Python	1	5	7	9	0,61
R	1/5	1	7	5	0,26
MatLab	1/7	1/7	1	3	0,09
ML.Net	1/9	1/5	1/3	1	0,05

Integration:

Table 4.6: Integration comparison matrix

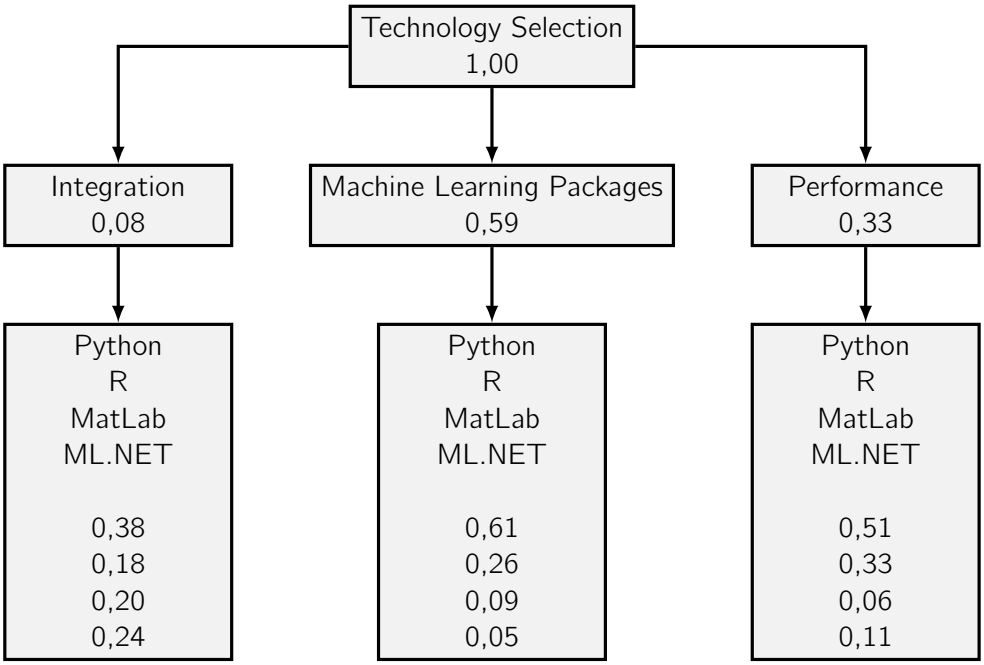
	Python	**R**	**MatLab**	**ML.Net**	**Relative Priority**
Python	1	3	7	1/3	0,38
R	1/3	1	5	1/5	0,18
MatLab	1/7	1/5	1	3	0,20
ML.Net	3	1/5	1/3	1	0,24

Performance:

Table 4.7: Performance comparison matrix

	Python	**R**	**MatLab**	**ML.Net**	**Relative Priority**
Python	1	3	5	5	0,51
R	1/3	1	7	5	0,33
MatLab	1/5	1/7	1	1/3	0,06
ML.Net	1/5	1/5	3	1	0,11

After all these steps, the initial diagram is updated with the calculated values:



The final step of this process, is to multiply the alternatives matrix by the criteria matrix:

$$\begin{bmatrix} 0,38 & 0,61 & 0,51 \\ 10,18 & 0,26 & 0,33 \\ 0,20 & 0,09 & 0,06 \\ 0,24 & 0,05 & 0,11 \end{bmatrix} X \begin{bmatrix} 0,59 \\ 0,08 \\ 0,33 \end{bmatrix} = \begin{bmatrix} \mathbf{0,56} \\ 0,28 \\ 0,089 \\ 0,085 \end{bmatrix}$$

Therefore, we can conclude that the first alternative should prove the best option for this set of criteria - Python.

4.2.2 System Architecture

In order to fulfill the requirements described in the previous section, an architectural diagram was envisioned, represented in Figure 4.3. This system has two main modules: a Data Configurator module and the actual Prediction model, named PrediMain in the figure 4.4.

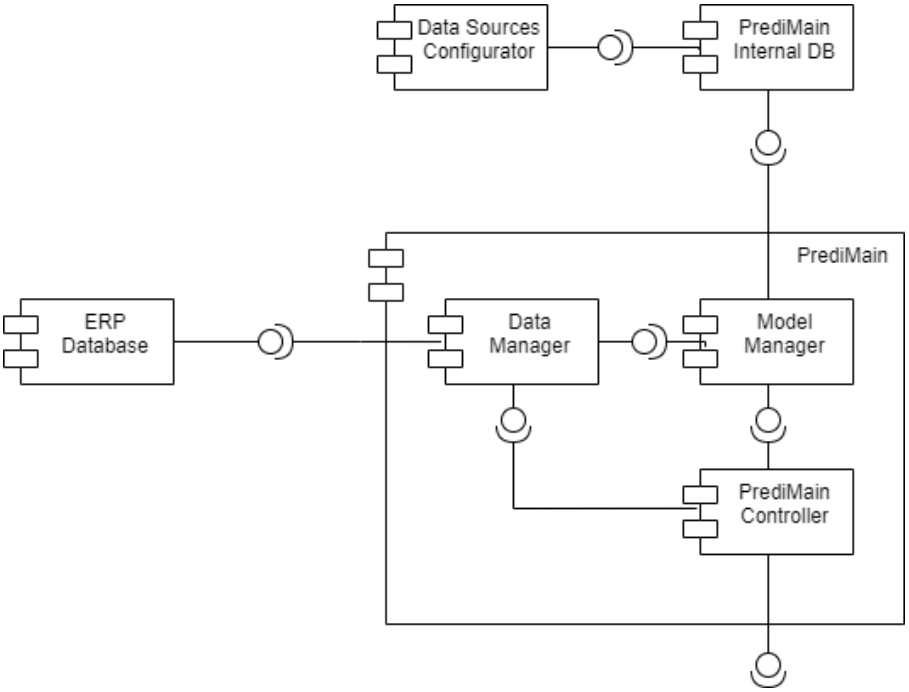


Figure 4.3: Envisioned System Architecture

.

However, two issues were identified with this possible approach. The first issue is that no clear separation of duties regarding the Model Building Data Cleaning and, most important, data retrieval, from both the ERP and PrediMain Database. With this in mind, a second approach, represented in Figure 4.4 was envisioned, having a Data Retrieval Modal that accesses the ERP Database and a Data Manager that would manage the temporarily stored data, and generated models, from the ERP in the internal database.

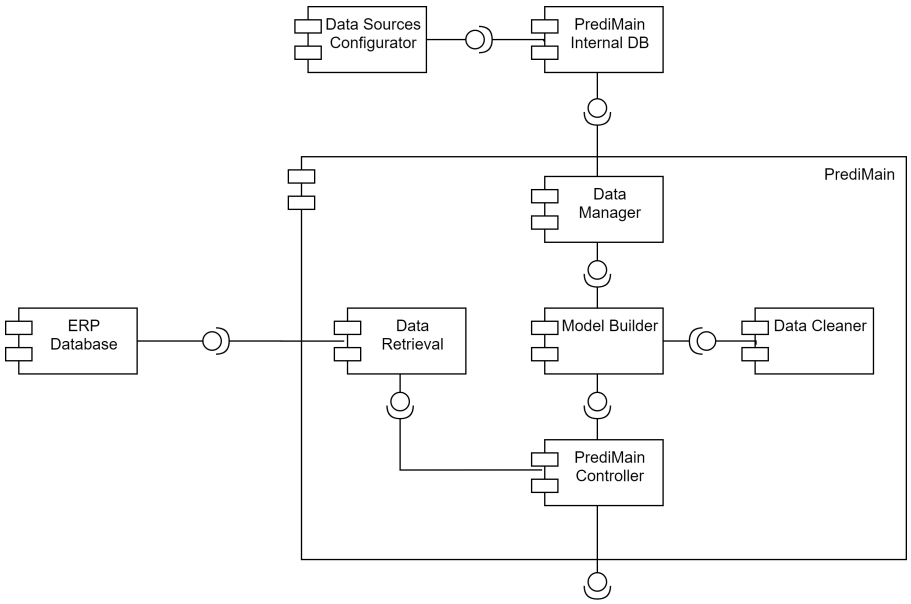


Figure 4.4: Final System Architecture

.

This internal database, that will serve as a repository for mainly two different entities/-datasets: the data sources available (as explained below) and the generated models. This Database should be prepared to hold unstructured information, given that the python object of the generated module should be directly stored.

Regarding this architecture, there were some design decisions that were made over its own alternatives. The first, is the usage of the internal database. The alternative was to store the model information in the same database of where the input data for the system would be coming from. However, by using an alternative database the Non-Functional Requirement is better served and fulfilled, and this way the system can be fully independent, with the exception of the data retrieval, from the systems that it will connect to. This also reduces the Data Manager sub-component complexity and improves the general system performance, not having to be dependent from the data source's database performance, since the data needed for the model will be stored in this system (as explained below, in the process view). Finally, this allows the model to be stored in any given format, while using the source's database there would be restrictions since the source database is likely to be a relational-database, and could not hold the generated model and unstructured data.

Data Configurator Module

As for the Data Configuration Module, its main objective is to answer the requirement of the system being as generic as possible (requirements NFS01 and NFS02). This subsystem allows a dynamic configuration of the different available datasources. Using an SQL datasource as an example, it provides a configuration of the connection string parameters needed for connecting to an SQL Database. This allows the system to be used as a Software-as-a-Service (SaaS), besides having the possibility of having it installed on-site.

The data sources information are stored in the PrediMain Internal Database, identified by a Data Source unique (and auto generated) identification number. Besides holding the

data needed for the connection, it also needs to store information on the database objects that contain, and should be used, to gather all the necessary data for the machine learning algorithms. In this case, a list of SQL Columns (and respective tables) should be configured, along with its datatypes and classification (numerical, categorical or a *datetime* type).

The Connection Data should be stored using the Advanced Encryption Standard (AES) cipher standard, in order to protect this sensitive information from being in plain text in the database. Using AES, being a Symmetric Key algorithm, allows the decryption of the connection string in order to be used during the system's runtime.

Prediction Module

As for the prediction module, it was decided to conceive it as a three-layer module. The bottom layer is responsible for managing the access to the system's internal database, named Data Manager component.

The Middle Layer is where the Model Building and Prediction will be made. For that, a component specialized in data cleaning was created, in order to clearly separate those critical processes from the model management and building itself.

Finally, the top level is comprised by a Controller, that offers an endpoint so that external applications can use to trigger all the prediction process: model building and the actual predictions.

The interaction between all of this components, and the main data flow of the system, is represented in the Process View Diagram - Figure 4.5. In the end of the process, the model is stored in the database, together with its creation date.

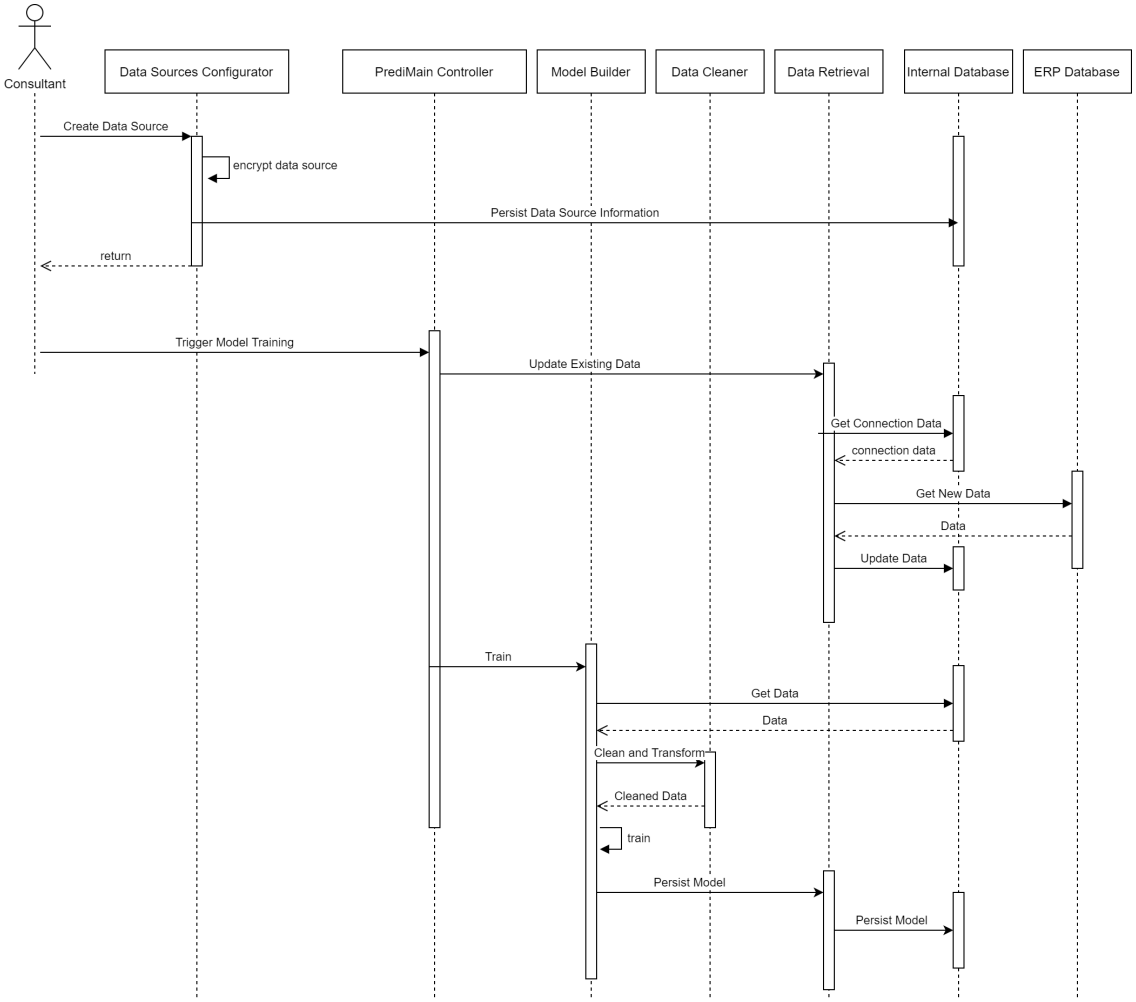


Figure 4.5: System Process View

.

In order to improve the performance of the system, a few constraints can be applied, regarding the model generation. Since generating a model it usually one of the most time and computationally heavy stages, the model should only be generated if one of the following premises is not true:

- No model exists already
- More than a certain (configurable) number of days have passed
- More than a certain (configurable) percentage of new data is available

This allows a good balance between an up-to-date model and performance concerns when request a prediction. The updated model, once generated, then replaces the old one stored in the internal database, to be used in subsequent prediction requests, until it is again classified as deprecated.

Chapter 5

Solution Implementation

This chapter describes both PrediMain modules, regarding its implementation and describing the decisions that were taken.

5.1 Data Configurator Module

The Data Configuration Module is a two-layer module (as described in section 4.2.2) using the shared PrediMain No-SQL Database. Since MongoDB uses BSON (Binary JSON) to store its documents, it was a natural decision to use a javascript-based technology for managing this data, besides having the possibility to hold unstructured data. For this, using the *mongoose* npm package, a model was defined using the following schema:

```
var schema = mongoose.Schema(
  {
    name: String,
    description: String,
    type: {
      type: String,
      enum: ['sql'],
      default: 'sql'
    },
    engine: {
      type: String,
      enum: ['mssql'],
      default: 'mssql'
    },
    serveraddress: {
      type: String
    },
    databasename: {
      type: String
    },
    authmode: {
      type: String
    },
    username: {
      type: String
    },
    password: {
      type: String
    },
    classcolumn:String,
    machineIDcolumn:String,
    features: [{
      name: {
        type: String,
        required: true
      },
      type: {
        type: String,
        enum: ['numerical', 'categorical', 'datetime', 'text'],
        default: 'text'
      }
    }]
  },
  { timestamps: true }
);
```

Listing 5.1: Mongoose Schema

The fields "name" and "description" are arbitrary and descriptive only. The "type" and "engine" refers to the database engine that is going to feed the PrediMain for this specific datasource. While the priority and main focus of this work is the Prediction Module and for now only SQL is supported, it was already developed having in mind that more engines might be supported in a near future. In this case, they would be added to the *enum* array. As for the "serveraddress" and "databasename" refers to the address of the server that hold the database, either a name or the ip address, and database name is the database of that server that will feed PrediMain. Authmode field is SQL Specific, where either a login via user name and password (configured in those fields respectively) or by using the windows

account, where the system validates the account name and password using the Windows principal token in the operating system.

The other important decision was related to how the query to get all the data needed would be stored. One option was to store it as an actual SQL Query, but it would decrease the adaptability of the system and it would be more error prone. The option that was taken, was to store the column names of the fields that are mandatory for the algorithms processing: the column that hold the class values (failure or not) and the column that identifies the machine related to that set of sensor values/observations. All the other columns that are used to predict the class - features - are configured separately and individually, stored in a feature's array. Each feature, it characterized by its name (corresponding to the column name) and by a type, so that the PrediMain knows how the value should be treated (for instance, as a date or a numerical value).

All of this information is managed by a controller class, containing CRUD methods and other functions to filter/query the documents. These methods are callable by using HTTP(s) Requests, to the API, consumed by the frontend webpage, developed in Angular.

5.2 Prediction Module

As decided and described previously, the Prediction Module was developed in Python. According to the Defined Architecture and Responsibility-Separation concerts, the project contains the file structure displayed in Figure 5.1

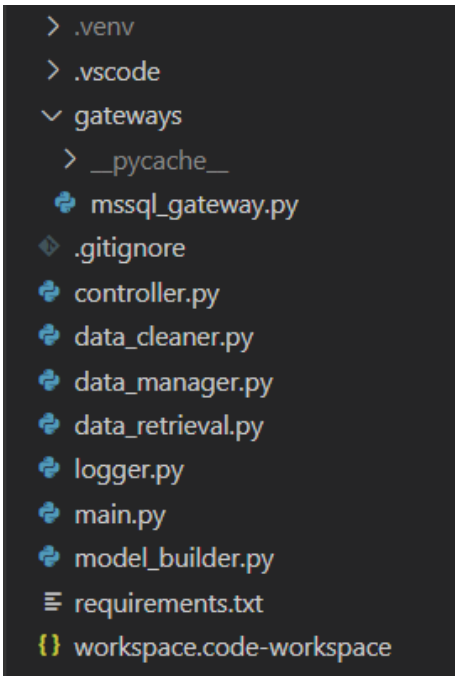


Figure 5.1: PrediMain File Structure

The file *requirements.txt* is a python environment specific used to store all the packages (and versions) used in the project, to make the project easier to deploy when installing in a

new system.
Regarding the Python files structure:

- main.py - The project entry point
- logger.py - A set of methods used to customize the project logging messages to console
- controller.py - The main system entity, responsible for initializing all other entities and to ensure the communication between them
- data_manager.py - Class used to communicate with the PrediMain internal database
- data_retrieval.py - Class used to communicate with external databases (configured as shown in the previous section, section 5.1)
- data_cleaner.py - Helper functions responsible for pre-processing the input data, such as removing NULL values and balance the dataset
- model_builder.py - Responsible for building and deciding the models to employ for a specific dataset

5.2.1 Database Gateways

Although the system as-is only supports SQL, as mentioned there is the possibility of supporting more database engines in the future. Just like in the Data Configurator Module, there was also a concern to make the connections to the external databases as generic and dynamic as possible. While Python does not have an abstract class concept, a similar function was implemented here,using the data_retrieval class.
A generic *get_gateway_object* method was created that, receiving the datasource configuration information, returns the correct instantiated gateway class:

```
from gateways.mssql_gateway import MSSQL_Gateway
class Data_Retrieval:
    def __init__(self):
        next

    def get_gateway_object(self,datasource):
        gateway = None
        if (datasource["engine"]=="mssql"):
            gateway = MSSQL_Gateway()
        return gateway
```

Listing 5.2: Data Retrieval class

Every Gateway class should implement the same set of methods, so that the Controller doesn't need to know what type of engine is being used.

```
import pyodbc
import pandas as pd
class MSSQL_Gateway:
    def __init__(self):
        next

    def get_conn(self,serveraddress,databasename,authtype,username=
    "",password=""):
        conn = None
        if (authtype=="windows"):
            conn = pyodbc.connect("Driver={SQL Server Native Client
     11.0};"
                                  "Server="+serveraddress+";"
                                  "Database="+databasename+";"
                                  "Trusted_Connection=yes;")
        else:
            conn = pyodbc.connect("Driver={SQL Server Native Client
     11.0};"
                                  "Server="+serveraddress+";"
                                  "Database="+databasename+";"
                                  "UID="+username+";"
                                  "PWD="+password+";"
                                  "Trusted_Connection=yes;")
        return conn

    def close_conn(self, conn):
        conn.close()

    def execute_query(self,query,conn,parse_dates=None,index_col=
    None):
        df = pd.read_sql(query, conn, parse_dates=parse_dates,
    index_col=index_col)
        return df
```

Listing 5.3: MSSQL Gateway class

5.2.2 Data Pre-Processing

Before building the actual models, the data goes through a pre-processing set of steps, in order to ensure that it is cleaned and in the correct format. The first verification has to do with data balancing. If the data is highly unbalanced, it can lead to over-fitted and biased models, who will fail to detect the rare (and helpful) events, being highly inaccurate. The strategy followed in PrediMain was to *OverSample* the data, since the number of failures (the ones we want to predict) is much lower:

```
    def balance(self, data, classes):
        new_data = data
        new_classes = classes
        if (((len(classes[classes == "Yes"])*100)/len(classes)) <
    5):
            # define oversampling strategy
            logger.log("INFO","Balancing Data")
            oversample = RandomOverSampler(sampling_strategy='
    minority')
            new_data, new_classes = oversample.fit_resample(data,
    classes)
        return new_data, new_classes
```

Listing 5.4: Data Balancing Operation

The second stage in the pre-processing step, is the Labelling process. A Machine Learning model can only process features that are numerical. However, we also have features that are categorical. In this case, they should be converted to a numerical form, using a *Label Encoding* strategy. This approach translates to converting each value in a column to a number, sequentially every time a new value appears. While this allows the models to be built without issues, it makes these numeric values possible to be misinterpreted as having an hierarchy/order, where a higher number represents something hierarchically higher that a lower number, which is mostly not the case.

To overcome this, a second strategy was applied, called *One-Hot Encoder*, on top of the previous one. What One-Hot Encoder does is to convert each category value into a new column, assigning 0 or 1 (false or true) to the new columns.

This approach does solve the issue with model misinterpretation, but can also quickly expand the model complexity as it can create a large number of new columns. The full snippet for the Labelling Process is then the following:

```
    def encode_categories(self,data,features,feature):
        labelencoder = LabelEncoder()
        features[feature] = labelencoder.fit_transform(features[
    feature])
        enc = OneHotEncoder(handle_unknown='ignore')
        enc_df = pd.DataFrame(enc.fit_transform(features[[feature
    ]]).toarray())
        features = features.join(enc_df)
        return features
```

Listing 5.5: Category Encoding Operation

As an example, tables 5.2.2, 5.2, and 5.3 shows the original, after Label Encoding and the sequential One-Hot Encoder process, respectively, using the second sensor which is stated to be categorical:

Date	Measure2
2016-01-01 9:00	0
2016-01-01 10:00	2
2016-01-01 11:00	3
2016-01-01 12:00	0
2016-01-01 13:00	1
2016-01-01 14:00	3

Table 5.1: Original sensor2 values

Date	Measure2
2016-01-01 9:00	0
2016-01-01 10:00	1
2016-01-01 11:00	2
2016-01-01 12:00	0
2016-01-01 13:00	3
2016-01-01 14:00	2

Table 5.2: Label Encoding result example

Date	Measure2_0	Measure2_1	Measure2_2	Measure2_3
2016-01-01 9:00	1	0	0	0
2016-01-01 10:00	0	1	0	0
2016-01-01 11:00	0	0	1	0
2016-01-01 12:00	1	0	0	0
2016-01-01 13:00	0	0	0	1
2016-01-01 14:00	0	0	1	0

Table 5.3: One-Hot Encoding result example

5.2.3 Classification Model

The Classification Models play a big role in PrediMain's system. They are used to detect and classify if a set of sensor observations should be considered a failure or not. As per the project's objectives, the idea behind PrediMain was to build a different set of models, including an ensemble one, and automatically decide which one fits best the training set of a given datasource. This way, every datasource has its own distinct model, suitable for their data.

The main Class for Classification Model building and selection is the Model Builder class. In this case, as a starting point for PrediMain, it was chosen to have the following algorithms available to be chosen:

- Logistic Regression (LR)
- K-Neighbors Classifier
- Decision Tree Classifier
- Support Vector Machines
- Ensemble Algorithm - Combination off all off the above, with an LR as the Meta Leaner

Since the problem is a classification one, a Logistic Regression was used for the Meta-Learner, as detailled in the state of the art, section 2.2.3. All of the models are then pushed into a Python Dictionary in order to be built and assessed, demonstrated by the following snippet:

```
    # get a stacking ensemble of models
    def build_classif_models(self):
        models = dict()
        models['lr'] = LogisticRegression()
        models['knn'] = KNeighborsClassifier()
        models['cart'] = DecisionTreeClassifier()
        models['svm'] = SVC()
        models['bayes'] = GaussianNB()
        models['stack'] = self.get_stacked_model()
        return models

    def get_stacked_model(self):
        # define the base models
        level0 = list()
        level0.append(('lr', LogisticRegression()))
        level0.append(('knn', KNeighborsClassifier()))
        level0.append(('svm', SVC()))
        level0.append(('bayes', GaussianNB()))
        # define meta learner model
        level1 = LogisticRegression()
        # define the stacking ensemble
        model = StackingClassifier(
            estimators=level0, final_estimator=level1, cv=5)
        return model
```

Listing 5.6: Classification Model Definition

Each of the algorithms goes through a standard training and testing stage. For more accurate test results, a repeated Stratified K-Fold cross validator was used, with 10 folds with 3 repetitions each. This means that a training set consisting of 90% of the total set, selected at random, is used with the remaining 10% used as the set for validation.

As for the algorithm scoring, it was initially planned to use the accuracy method. However, it was decided to use the AUC-ROC (area under the ROC curve, as detailed in section 2.2.4):

```
    def evaluate_model(self, model, X, y):
        cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,
    random_state=1)
        scores = cross_val_score(
            model, X, y, scoring='roc_auc', cv=cv, n_jobs=-1,
    error_score='raise')
        return scores
```

Listing 5.7: Classification Model Evaluation

5.2.4 Numerical Prediction Model

The classification models themselves wouldn't fulfill the PrediMain's objective of predicting machine failures, since they do not predict future values, only classify a set of observations. In this case, we would only apply the models to a current observation, which wouldn't be that helpful to the operators. Therefore, another set of machine learning algorithms is applied, in order to predict the next set of values for each sensor installed at the machine.

Figure 5.2 represents a plotted set of 100 values from a sample sensor, retrieved from the data that is going to be used as a test dataset.

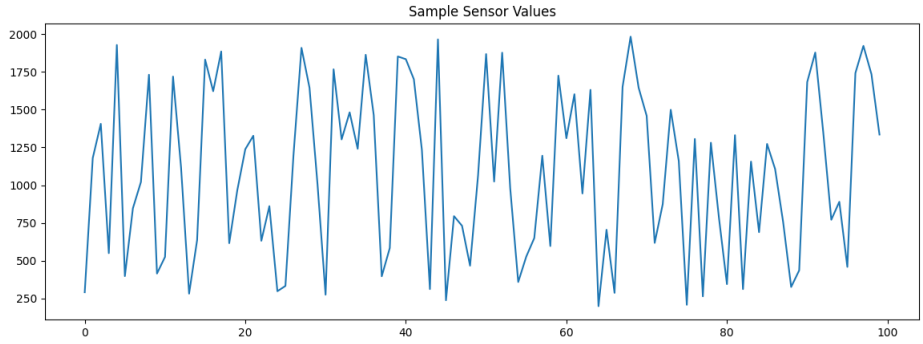
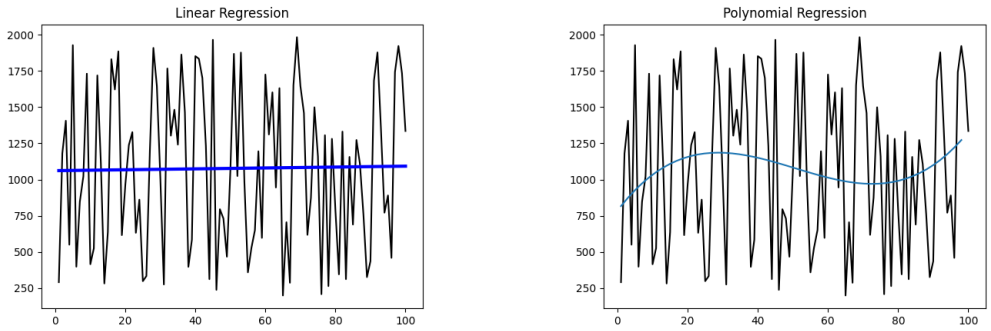


Figure 5.2: Sample Sensor Values

The simplest way to predict the next values would be to use a regression algorithm. However, the data, as it can be seen from the plot, cannot be represented by a linear equation, since it is highly dispersed in the y-axis. Figure 5.3 (a) and (b), represent a linear and a polynomial regression applied to the data, respectively.



(a) Sample Linear Regression

(b) Sample Polynomial Regression

Figure 5.3: Linear and Polynomial Regression Equations

Nonetheless the Polynomial approach is slightly better, it is still far from being an adequate model to use, even with a higher degree applied. Additionally, this data consists in time series: a sequence of data points recorded at specific times, where the temporal continuity is the predominant factor to consider [3]. To overcome this, a different approach was used, called ARIMA Model - AutoRegressive Integrated Moving Average. Autoregressive means that "it predicts future values based on past values", meaning that represents time-variable processes with stochastic features. Integrated means that the data values have been replaced with the difference between their values and the previous ones. As for Moving Average, it means that the model uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Each of these components are explicitly specified in the model as a parameter, with the standard notation of ARIMA(p,d,q):

- p: The number of lag observations included in the model

- d: The number of times that the raw observations are differenced
- q: The size of the moving average window

While it is possible to try each combination of parameters, it would be an exhausting and time-consuming task. For this, using the package *pmdarima*, an auto_arima method was used, which returns the best set of parameters for the algorithm in a specified range by testing each possible combination:

```
auto_arima(data,
    d=1, # non-seasonal difference order
    start_p=0, # initial guess for p
    start_q=0, # initial guess for q
    max_p=2, # max value of p to test
    max_q=2, # max value of q to test
    seasonal=True, # seasonal time series
    m = 24, # the seasonal period
    #D=1, # seasonal difference order
    start_P=1, # initial guess for P
    start_Q=1, # initial guess for Q
    max_P=1, # max value of P to test
    max_Q=1, # max value of Q to test
    information_criterion='aic', #score used to select the best
   model,
    trace=True,
    error_action='ignore',
    suppress_warnings=True,
    stepwise=True)
```

Listing 5.8: Auto Arima Model

The selection of the best model is set to be using the AIC (Akaike Information Criterion) estimator. AIC estimates the quality of each model, relative to each of the other model, therefore providing a reliable mean for the model selection task. Since it rewards a model's goodness of fit, while penalizing its complexity, the AIC performs a trade-off between overfitting and underfitting. It was considered to set the ARIMA model as Seasonal (called SARIMA), with a calculated timeframe between the failures (seasons). This calculation is translated to the average of months between all failures, and it's set in the parameter *m*. In this case, three additional parameters are added to the notation, being a SARIMA(P,D,Q)(p,d,q)m:

- P: The number of lag observations included in the model (seasonal component)
- D: The number of times that the raw observations are differenced (seasonal component)
- Q: The size of the moving average window (seasonal component)
- p: The number of lag observations included in the model
- d: The number of times that the raw observations are differenced
- q: The size of the moving average window
- m: The number of periods in each season

For PrediMain, the *m* parameter was set as 24, indicating that the data should be treated as hourly, with a 24-hours configuration.

After all this process is conducted, the system generates the next values for each sensor, using the selected ARIMA model, and applies the selected classification algorithm, getting an array of the predicted classes:

```
sensor_predictions = self.get_sensor_predictions(
            sensor_predictors, datasource_info.get("features")) #
    uses the ARIMA model
predictions = (classif_model.predict(sensor_predictions))

def get_sensor_predictions(self, models, features):
    df = DataFrame()
    for feature in features:
        if feature.get("type")=="numerical":
            fitted = models[feature.get("name")].predict(
                n_periods=30, return_conf_int=False)
            df[feature.get("name")] = fitted.tolist()
    return df
```

Listing 5.9: Failure Prediction

It was configured to make the ARIMA model predict 30 periods by default, which means that the next thirty hours are analysed for possible machine failures. It was chosen thirty hours, representing a day and a quarter, to achieve a good balance between the variable time (and usefulness) and predictions accuracy, where predicting further in time wields typically worse results.

Chapter 6

Experimentation and Evaluation

This chapter describes the Experimentation and Evaluation processes that occurred during, and after, the system development stage. It has a particular emphasis on the algorithm's performance evaluation, that is automatically generated and compared during the predictions and model assessment, as described in the previous chapter.

6.1 Data Analysis and DataSource Configuration

Since PrediMain is based on automatic model building and selection, it becomes even more important to conduct an analysis on the data that is going to be fed into the system, in order to properly configure all the fields that are going to be used. Before analysing some common metrics on the columns, some of them were discarded from the beggining since they would be already useless for the predictions, which are the columns indicating the day, month, year, hour, minute and second of the observation. Table 6.1, 6.2, 6.3 and 6.4 represent a set of metrics on every other column.

	Date	**Measure1**	**Measure2**	**Measure3**	**Measure4**
count	7905	7905	7905	7905	7905
unique			4	3	
top			2	1	
freq			2048	2669	
mean	2016-06-13 16:00:00	1093,46			1068,78
min	2016-01-01 00:00:00	155			155
25%	2016-03-23 08:00:00	633			605
50%	2016-06-13 16:00:00	1099			1056
75%	2016-09-04 00:00:00	1556			1530
max	2016-11-25 08:00:00	2011			2011
std		535,52			535,61

Table 6.1: Statistic on the datasource's columns (1)

	Measure5	Measure6	Measure7	Measure8	Measure9	Measure10
count	7905	7905	7905	7905	7905	7905
unique						
top						
freq						
mean	1075,30	1077,78	1089,26	1076,06	1082,00	1082,09
min	155	155	155	155	155	155
25%	604	626	625	609	634	618
50%	1075	1073	1091	1073	1076	1077
75%	1542	1541	1561	1540	1527	1547
max	2011	2011	2011	2011	2011	2011
std	533,72	533,42	537,14	537,24	531,27	538,06

Table 6.2: Statistic on the datasource's columns (2)

	Measure11	Measure12	Measure13	Measure14	Measure15
count	7905	7905	7905	7905	7905
unique					
top					
freq					
mean	1089,27	1089,17	1074,60	1091,16	1083,81
min	155	155	155	155	155
25%	628	630	605	621	616
50%	1093	1081	1064	1090	1078
75%	1548	1551	1537	1564	1552
max	2011	2011	2011	2011	2011
std	533,72	533,42	537,14	537,24	531,27

Table 6.3: Statistic on the datasource's columns 31)

	Operator	Temperature	Humidity	Failure
count	7905	7905	7905	7905
unique	8			2
top	Operator2			No
freq	1753			7830
mean		64,03	83,71	
min		5	65	
25%		62	80	
50%		64	84	
75%		66	87	
max		78	122	
std	538,06			

Table 6.4: Statistic on the datasource's columns (4)

From these analysis, a valuable insight can already be detected, which is related to the data types. While most of the columns should be treated as numeric, since they have a wide range

and variability, Measures 2 and 3, and the Operator column should be treated as categorical, even though the first two metrics are of a numeric data type. All of the other Measures are quite similar, having a close mean and the same minimum and maximum values. The *Failure* column is identified to have only two distinct values (yes and no), and therefore will be the class column.

This has proven to be a valuable step for correctly building prediction models, since if the configuration would rely on only analysis the actual data types, it would lead to wrong and useless models, since the categorical variables would be a standard numerical ones.

Using the visual interface for the configuration of the datasource, the PrediMain can now receive requests for predictions and process the data. Figure 6.1 shows the configuration that was made in the interface.

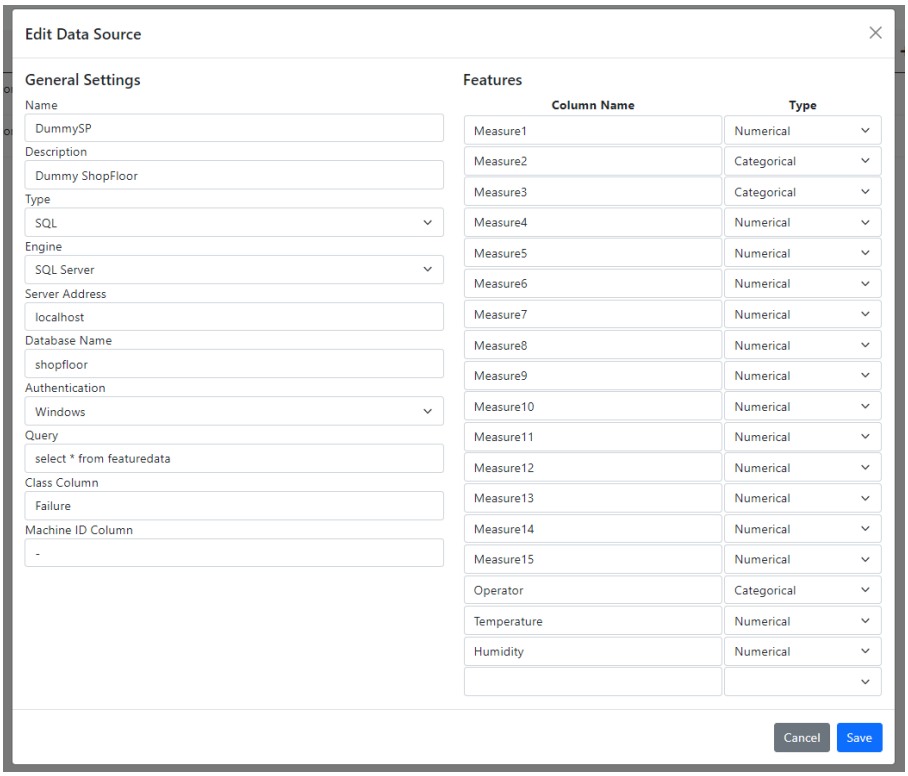


Figure 6.1: PrediMain datasource configuration

6.2 Evaluation of the Generated Models

One of the Evaluation Processes of the present system, is related to the Machine Learning Algorithms performance during the predictions that will be made.
Firstly, the Model Builder, chooses the technique/algorithm that better suits the given datasource that matches the incoming request. Since every dataset is different, an automated way of choosing the best technique, among the ones described in section 2.2 was used. However, an extended analysis was made using the literature dataset, in order to better describe and assess the system's performance and quality in deciding the suitable models.

Regarding the classification models, as mentioned in section 5.2.3, the system internally compares their AUC-ROC - Area Under the ROC Curve. For the dataset used to test PrediMain, table 6.5 represents the accuracy of each model.

Model	**Accuracy**
Random Forest	0,876
K-Nearest Neighbors	0,543
Decision Tree	0,843
SVM	0,540
Stacked	0,902

Table 6.5: Evaluated Models' accuracies

As inferred, and as expected, the stacked model performs better than every other model since it extracts the best features of its base models.

Regarding the numerical predictions, as mentioned previously, the system automatically selected the best ARIMA parameters to generate the ARIMA model, using *auto_arima*. Table 6.6 represents the generated combinations for each sensor and their AIC value (with a minimization strategy), which was the selected scoring metric.

	M1	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**	**M10**	**M11**	**M12**	**M13**	**M14**	**M15**
ARIMA(0,1,0)(0,0,0)[24]	1136,08	1153,13	1154,36	1142,89	1124,40	1156,99	1142,06	1151,49	1143,94	1155,04	1144,96	1166,74	1130,68
ARIMA(0,1,0)(0,0,1)[24]								1148,56					
ARIMA(0,1,0)(1,0,0)[24]											1138,281		
ARIMA(0,1,0)(1,0,1)[24]	1138,163			1146,887	1128,333			1150,56			1140,05		
ARIMA(0,1,1)(0,0,0)[24]		1106,408			1094,401		1106,047						
ARIMA(0,1,1)(0,0,1)[24]		1107,858			1096,094		1107,827						
ARIMA(0,1,1)(1,0,0)[24]		1107,898			1096,148		1107,786						
ARIMA(0,1,1)(1,0,1)[24]		1109,774											
ARIMA(0,1,2)(0,0,0)[24]		1107,411					1108,008						
ARIMA(1,1,0)(0,0,0)[24]	1116,884	1119,956	1137,48	1104,019	1110,798	1139,551	1120,469	1140,958	1125,563	1128,758	1125,833	1147,462	1114,729
ARIMA(1,1,0)(0,0,1)[24]	1117,507	1123,675	1139,413	1105,208		1141,546		1139,003	1127,534	1130,454	1123,368	1148,214	1115,68
ARIMA(1,1,0)(1,0,0)[24]	1117,337	1123,653	1139,396	1105,194	1112,797	1141,548	1121,103	1139,1	1127,533	1130,445	1123,128	1148,781	1116,157
ARIMA(1,1,0)(1,0,1)[24]		1125,626		1107,166				1140,995		1132,427	1125,126		
ARIMA(1,1,1)(0,0,0)[24]		1106,796					1107,989						
ARIMA(1,1,1)(0,0,1)[24]		1108,674											
ARIMA(1,1,1)(1,0,0)[24]		1108,685											
ARIMA(1,1,1)(1,0,1)[24]													
ARIMA(1,1,2)(0,0,0)[24]		1108,467			1097,717								
ARIMA(1,1,2)(0,0,1)[24]													
ARIMA(2,1,0)(0,0,0)[24]	1114,126	1114,353	1135,662	1100,014		1135,693		1129,679	1115,869	1123,598	1121,334	1127,324	1104,566
ARIMA(2,1,0)(0,0,1)[24]	1114,665	1116,179	1137,601	1101,374		1137,583		1128,543	1117,729	1125,564	1118,327	1122,349	1104,52
ARIMA(2,1,0)(1,0,0)[24]	1114,458	1116,177	1137,584	1101,383		1137,621		1128,486	1117,737	1125,56	1117,3	1125,062	1105,472
ARIMA(2,1,0)(1,0,1)[24]		1118,177		1103,374				1130,482	1119,72		1118,725		
ARIMA(2,1,1)(0,0,0)[24]									1117,62				
ARIMA(2,1,1)(0,0,1)[24]												1101,805	
ARIMA(2,1,1)(1,0,0)[24]													
ARIMA(2,1,1)(1,0,1)[24]													
ARIMA(2,1,2)(0,0,0)[24]													
ARIMA(2,1,2)(0,0,1)[24]													

Table 6.6: Generated ARIMA Combinations and Scoring

As an example Figure 6.2 represents the ARIMA model plotted on top of the first sensor's values.

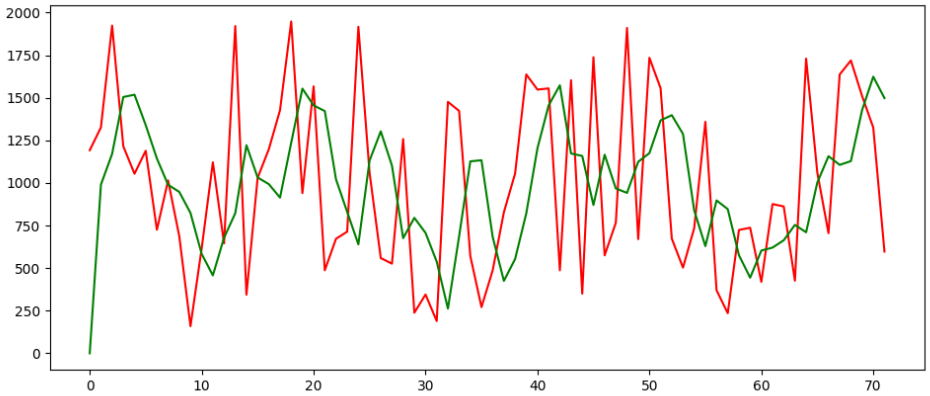


Figure 6.2: ARIMA Model Forecast for Sensor1

It is possible to observe that this SARIMA algorithm captures the trend (upward/downward) from the spikes pretty well, which is one of the most important factors to have in mind when using these type of approaches. Appendix C contains all of generated ARIMA Models. Most of the ARIMA models worked well by correctly capturing the spikes in the data, with an exception for two of the sensors. This issues are addressed as a future work to be done, in section 7.2.

Finally, Figure 6.3 presents the full output by PrediMain's logger, representing the full workflow and system's runtime processes.

```
[WARNING] 2021-09-18 16:29:42,293 Model is outdated. Building a new one.
[INFO] 2021-09-18 16:29:42,295 Evaluating Models
[INFO] 2021-09-18 16:29:44,243 Model is rf with an estimated accuracy of 0.857%
[INFO] 2021-09-18 16:29:44,437 Model is knn with an estimated accuracy of 0.543%
[INFO] 2021-09-18 16:29:44,638 Model is cart with an estimated accuracy of 0.836%
[INFO] 2021-09-18 16:29:45,123 Model is svm with an estimated accuracy of 0.541%
[INFO] 2021-09-18 16:30:03,291 Model is stack with an estimated accuracy of 0.894%
[INFO] 2021-09-18 16:30:03,292 Selected Model is stack with an estimated accuracy of 0.894%
[INFO] 2021-09-18 16:30:05,767 Storing generated model
[INFO] 2021-09-18 16:30:06,952 Building Sensor Predictor for Measure1 (1/13)
[INFO] 2021-09-18 16:30:12,168 Building Sensor Predictor for Measure4 (2/13)
[INFO] 2021-09-18 16:30:22,302 Building Sensor Predictor for Measure5 (3/13)
[INFO] 2021-09-18 16:30:28,409 Building Sensor Predictor for Measure6 (4/13)
[INFO] 2021-09-18 16:30:32,134 Building Sensor Predictor for Measure7 (5/13)
[INFO] 2021-09-18 16:30:36,297 Building Sensor Predictor for Measure8 (6/13)
[INFO] 2021-09-18 16:30:42,193 Building Sensor Predictor for Measure9 (7/13)
[INFO] 2021-09-18 16:30:46,922 Building Sensor Predictor for Measure10 (8/13)
[INFO] 2021-09-18 16:30:57,405 Building Sensor Predictor for Measure11 (9/13)
[INFO] 2021-09-18 16:31:02,273 Building Sensor Predictor for Measure12 (10/13)
[INFO] 2021-09-18 16:31:07,731 Building Sensor Predictor for Measure13 (11/13)
[INFO] 2021-09-18 16:31:18,729 Building Sensor Predictor for Measure14 (12/13)
[INFO] 2021-09-18 16:31:33,785 Building Sensor Predictor for Measure15 (13/13)
```

Figure 6.3: PrediMain's loggger output

Chapter 7

Conclusions

In this chapter, final remarks on PrediMain are made and an analysis on the PrediMain's current limitations is conducted.

7.1 Work Summary

PrediMain was built to meet three objectives: build a platform capable of doing predictive maintenance (by predicting if a machine will likely fail in the next hours), to evaluate whether an ensemble technique would perform better than a single type of algorithm, and to build a system capable of auto improving its models.

All objectives have been accomplished, ending with a system high a high accuracy (around ninety percent) and without having overfitting issues. Currently, it is capable of actively doing predictive maintenance, giving alerts on the likelihood of a machine breakdown in the next few hours, It can, abd should, be used as a basis for an even more developed system, given the future work mentioned in the next section.

This project had a very positive implication not only on a personal basis but also professionally, both by allowing to gain knowledge on state-of-the-art technologies and techniques and also to apply them into a real world problem and scenario.

7.2 Limitations and Future Work

While the objectives were accomplished, there is still some key issues that should be updated/added in a near future, in order to improve PrediMain even more. The first issue is related to the classification models. While they prove to be reliable, they are still used with, mostly, their default parameters. Since PrediMain is intended to be dynamic and auto-configurable (model parameterization-wise, as it is with the ARIMA predictions), a way of also automatically setting these parameters would be needed.

Finally, another key issue is regarding the numerical predictions. While they proved to be accurate, besides trying to improve the fitting (as noted previously), a new sequence prediction strategy should be implemented, specifically for the categorical variables. As of now, they use the ARIMA models which, while functionally working, are not the ideal models for these scenarios, since they are directly using the numerical output of the One-Shot Encoding.

Bibliography

[1] Päivi Parviainen et al. "Tackling the digitalization challenge: how to benefit from digitalization in practice". In: *International journal of information systems and project management* 5.1 (2017), pp. 63–77.

[2] Andreas Schumacher, Tanja Nemeth, and Wilfried Sihn. "Roadmapping towards industrial digitalization based on an Industry 4.0 maturity model for manufacturing enterprises". In: *Procedia CIRP* 79 (2019). 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy, pp. 409–414. issn: 2212-8271. doi: https://doi.org/10.1016/j.procir.2019.02.110. url: http://www.sciencedirect.com/science/article/pii/S2212827119302276.

[3] Marta Fernandes et al. "Fault Detection Mechanism of a Predictive Maintenance System Based on Autoregressive Integrated Moving Average Models". In: *Distributed Computing and Artificial Intelligence, 16th International Conference*. Ed. by Francisco Herrera, Kenji Matsui, and Sara Rodríguez-González. Cham: Springer International Publishing, 2020, pp. 171–180. isbn: 978-3-030-23887-2.

[4] Shan Wan et al. "Knowledge Management for Maintenance, Repair and Service of Manufacturing System". In: Sept. 2014.

[5] "Contents". In: *An Introduction to Predictive Maintenance (Second Edition)*. Ed. by R. Keith Mobley. Second Edition. Plant Engineering. Burlington: Butterworth-Heinemann, 2002, pp. v–xii. isbn: 978-0-7506-7531-4. doi: https://doi.org/10.1016/B978-075067531-4/50000-2.

[6] *Predictive Maintenance with MATLAB*. 2019. url: https://www.mathworks.com/campaigns/offers/predictive-maintenance-with-matlab.html.

[7] N. Jazdi. "Cyber physical systems in the context of Industry 4.0". In: *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*. 2014, pp. 1–4. doi: 10.1109/AQTR.2014.6857843.

[8] F. Shrouf, J. Ordieres, and G. Miragliotta. "Smart factories in Industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm". In: *2014 IEEE International Conference on Industrial Engineering and Engineering Management*. Dec. 2014, pp. 697–701. doi: 10.1109/IEEM.2014.7058728.

[9] Omkar Motaghare, Anju Pillai, and K.I. Ramachandran. "Predictive Maintenance Architecture". In: Dec. 2018, pp. 1–4. doi: 10.1109/ICCIC.2018.8782406.

[10] K. Wang. "Intelligent Predictive Maintenance (IPdM) System – Industry 4.0 Scenario". In: *WIT transactions on engineering sciences* 113 (2016), pp. 259–268.

[11] N. Jazdi. "Predictive Maintenance - Taking pro-active measures based on advanced data analytics to predict and avoid machine failure". In: *Analytics Institute*. Vol. 7. 2017.

[12] Brett Lantz. *Machine Learning with R*. 2nd ed. Packt Publishing, 2015. isbn: 978-1-78439-390-8.

[13] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. "From Data Mining to Knowledge Discovery in Databases". In: *AI Magazine* 17.3 (Mar. 1996), p. 37.

doi: 10.1609/aimag.v17i3.1230. url: https://ojs.aaai.org/index.php/aimagazine/article/view/1230.

[14] *Overview of the KDD Process*. http://www2.cs.uregina.ca/~dbd/cs831/notes/kdd/1_kdd.html. Accessed: 2020-12-26.

[15] Fátima Rodrigues. *Data Mining*. University Lecture. 2018.

[16] *Visitor Segmentation using K-means Clustering*. Accessed: 2021-03-06. url: https://medium.com/analytics-vidhya/visitor-segmentation-using-k-means-clustering-c874dcd41785.

[17] NVIDIA. *Overview of the KDD Process*. https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/. Accessed: 2021-01-06.

[18] *CLASSIFICATION: meaning in the Cambridge English Dictionary*. url: https://dictionary.cambridge.org/dictionary/english/classification.

[19] *Classifying data with decision trees: elf11.github.io*. url: https://elf11.github.io/2018/07/01/python-decision-trees-acm.html.

[20] Xindong Wu et al. "Top 10 algorithms in data mining". In: *Knowledge and Information Systems* 14 (Dec. 2007). doi: 10.1007/s10115-007-0114-2.

[21] Sarang Anil Gokte. *Most Popular Distance Metrics Used in KNN and When to Use Them*. url: https://www.kdnuggets.com/2020/11/most-popular-distance-metrics-knn.html.

[22] Wikipedia contributors. *Least squares — Wikipedia, The Free Encyclopedia*. Accessed: 2021-01-11. 2020. url: https://en.wikipedia.org/w/index.php?title=Least_squares&oldid=996886074.

[23] Evan Lutins. *Ensemble Methods in Machine Learning: What are They and Why Use Them?* Aug. 2017. url: https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f.

[24] Joseph Rocca. *Ensemble Learning, Bagging, and Boosting Explained in 3 Minutes*. 2020. url: https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205.

[25] Joseph Rocca. *Understanding Gradient Boosting Machines*. 2020. url: https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab.

[26] Gaurika Tyagi. *Ensemble models for Classification*. 2020. url: https://towardsdatascience.com/ensemble-models-for-classification-d443ebed7efe.

[27] *What is Stacking in Machine Learning*. 2020. url: https://www.kaggle.com/questions-and-answers/199346.

[28] *Model Evaluation Metrics in Machine Learning*. url: https://www.kdnuggets.com/2020/05/model-evaluation-metrics-machine-learning.html.

[29] G. A. Susto et al. "Machine Learning for Predictive Maintenance: A Multiple Classifier Approach". In: *IEEE Transactions on Industrial Informatics* 11.3 (2015), pp. 812–820. doi: 10.1109/TII.2014.2349359.

[30] M. Paolanti et al. "Machine Learning approach for Predictive Maintenance in Industry 4.0". In: *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*. 2018, pp. 1–6. doi: 10.1109/MESA.2018.8449150.

[31] *MindSphere - MindConnect Documentation*. url: https://siemens.mindsphere.io/en/docs/mindconnect.

[32] *MindSphere Predictive Maintenance - Improve your bottom line with operational transparency*. url: https://www.plm.automation.siemens.com/global/en/resource/mindsphere-predictive-maintenance-nurture/88120.

[33] Peter Koen et al. "Providing Clarity and Common Language to the Fuzzy Front End". In: *Research-Technology Management* 44 (Mar. 2001), pp. 46–55.

[34] Atte Martikainen. "Front End of Innovation in Industrial Organization". PhD thesis. Nov. 2017.

[35] K. Dewulf. *Sustainable Product Innovation: The Importance of the Front- End Stage in the Innovation Process*. IntechOpen, 2013. isbn: 978-953-51-1016-3.

[36] Christian Krupitzer et al. *A Survey on Predictive Maintenance for Industry 4.0*. 2020. arXiv: 2002.08224 [cs.LG].

[37] Hongxia Wang, Xiaohui Ye, and Ming Yin. "Study on Predictive Maintenance Strategy". In: July 2016, pp. 52–56. doi: 10.14257/astl.2016.137.10.

[38] Boksberger Philipp E. and Melsen Lisa. "Perceived value: a critical examination of definitions, concepts and measures for the service industry". In: *Journal of Services Marketing* 25.3 (Jan. 2011), pp. 229–240. issn: 0887-6045. doi: 10.1108/08876041111129209. url: https://doi.org/10.1108/08876041111129209.

[39] Anna Whiting. "Six Steps to Crafting Effective Value Propositions". In: *ITSMA* (2012).

[40] Alexander Osterwalder et al. *Value proposition design*. Wiley, 2014.

[41] *WHAT IS QUALITY FUNCTION DEPLOYMENT (QFD)?* Accessed: 2021-02-13. url: https://asq.org/quality-resources/qfd-quality-function-deployment.

[42] *Machine Failure Prediction*. Accessed: 2021-03-05. url: https://www.kaggle.com/c/machine-failure-prediction/data?select=test.csv.

[43] E. Triantaphyllou and S. Mann. "USING THE ANALYTIC HIERARCHY PROCESS FOR DECISION MAKING IN ENGINEERING APPLICATIONS: SOME CHALLENGES". In: 1995.

Appendix A

AHP Calculations

The priority vector to be used in the CI calculus corresponds to the relative priorities found in table 4.4: (0.59,0.08,0.33). In order to calculate the needed eigenvalue, the following formula was used:

$$Ax = \lambda_{max} X$$

Where A is the comparison matrix in table 4.4 and x is the priority vector.

Therefore:

$$\begin{bmatrix} 1 & 7 & 2 \\ \frac{1}{7} & 1 & \frac{1}{5} \\ \frac{1}{2} & 5 & 1 \end{bmatrix} \text{X} \begin{bmatrix} 0.59 \\ 0.08 \\ 0.33 \end{bmatrix} \cong \lambda_{max} \begin{bmatrix} 0.59 \\ 0.08 \\ 0.33 \end{bmatrix}$$

$$\begin{bmatrix} 1.81 \\ 0.23 \\ 1.03 \end{bmatrix} \cong \lambda_{max} = \begin{bmatrix} 0.59 \\ 0.08 \\ 0.33 \end{bmatrix}$$

λ_{max} = average{1.81/0.59, 0.23/0.08, 1.03/0.33} $\simeq$ 3.02

The CI is then given by, where n is the number of criteria:

$$CI = \frac{\lambda_{max} - n}{(n-1)} = \frac{(3.02-3)}{(3-1)} = 0.015$$

Then,

$$RC = IC/0.58 = 0.015/0.58 = 0.02$$

Appendix B

Quantitative Evaluation Framework

Dimension	Functionality			
Factor	Functional			
		Wfk - Fullfilment (%)		
Requirement	Metric Evaluation	0	50	100
FF01 - Manage the available Data Sources	Client can create/edit/delete data sources to be used in the predictions	No access to functionality	Partial access to the functionality	Full access to the functionality
FF02 - Train the Machine Learning Model	Client can train the model *on-demand*	No access to functionality	-	Full access to the functionality
FF02 - Train the Machine Learning Model	The system automaticly trains the model when there is no previous model created or the model is dated	Not Working	-	Fully working
FF03 - Change the Algorithm Input Parameters	Client can change the algorithm input parameters	No access to functionality	-	Full access to the functionality
FF04 - Visualize the Model's Predictions	Client can visualize the predictions made in a graphical form	No access to functionality	Partial access to the functionality	Full access to the functionality

Figure B.1: Quantitative Evaluation Framework - Functional

.

Dimension	Functionality			
Factor	Content Quality			
		Wfk - Fullfilment (%)		
Requirement	Metric Evaluation	0	50	100
FCQ01 - All the presented information is well organized	The presented information must be coeherent and organized so that less experience users have no difficulty in understanding it	No	-	Yes
FCQ02 - The presented statistic should be easy to understand	The User Interface for showing the predictions that were madeshould not show technical data that could not possible beunderstood by the maintenance managers	No	-	Yes

Figure B.2: Quantitative Evaluation Framework - Content Quality

.

Dimension	Adaptability			
Factor	Versatility, Motivacional Aspects, Pedagogical Aspects, Maintenance			
		Wfk – Fullfilment (%)		
Requirement	Metric Evaluation	0	50	100
AV01 - The system should support any SQL Database for the input data	he system should support any SQL database, needing thespecification of all data needed for the connection (such as IPAddress, username, password and port number)	No	-	Yes

Figure B.3: Quantitative Evaluation Framework - Adaptability

.

Efficiency			
Audiovisual quality, Aesthetic elements and techniques, Navigation			
	Wfk – Fullfilment (%)		
Metric Evaluation	0	50	100
The Data Retrieval from the existent systems shouldn't affecttheir performance	No	-	Yes

Figure B.4: Quantitative Evaluation Framework - Efficiency

.

Appendix C

Generated ARIMA models for the sensors

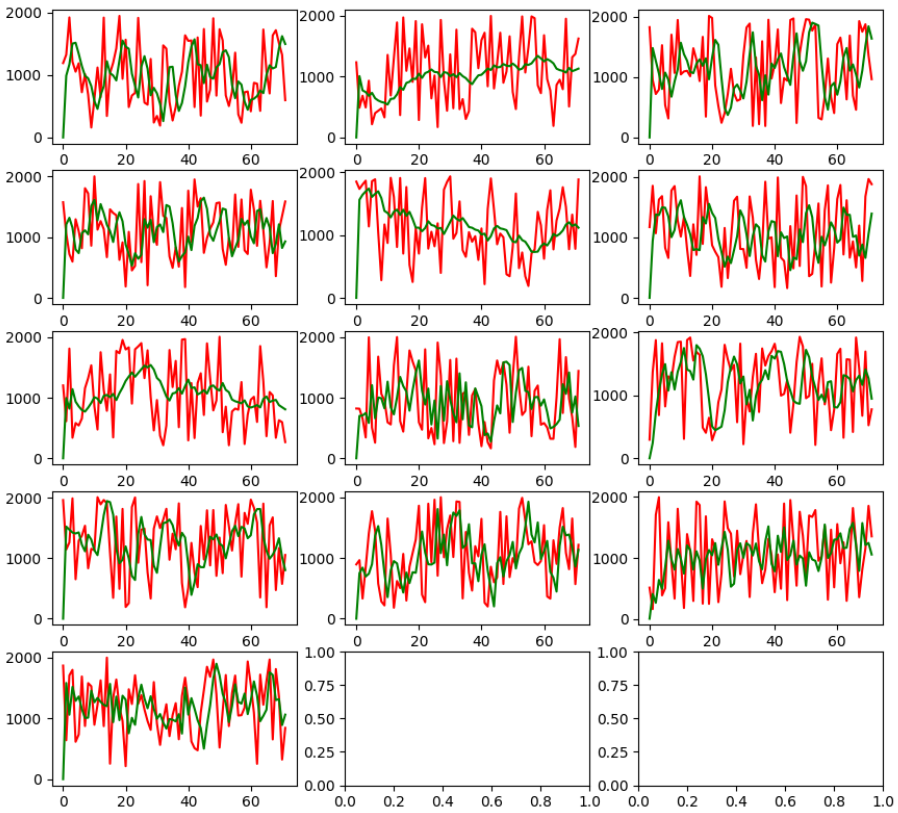


Figure C.1: Generated ARIMA models for the sensors

.