Data Mining – Prediction Project
Group Assignment

Vasco Moutinho 355422066
Simão Magalhães 355422067
Sofia Fernandes 355422038
Tiago Silva 355422039

Abstract

In the scope of the Data Mining curricular unit, it was proposed the development of an analysis and a study of the "Predictive Maintenance" database, in order to address a prediction problem, namely a classification problem. The database, entitled "db" in the R platform, presents 1000 observations and 10 variables, providing a set of predictive maintenance classification data for a given machine, with an identifier and an ID for each product, a column showing the type of quality, the different elements present during the operation of the equipment, a prediction associated with the existence of failures in the final product and the types of failures that occurred.

The main focus of the work is to do a classification analysis in order to predict when a machine will fail and the type of failure, and for this purpose four different algorithms were used (k-Nearest Neighbor (kNN), Naive Bayes, Decision Tree and Random Forest). The report will consist of an exploration and analysis of each algorithm to understand which one should be used, based on some metrics such as Accuracy, Sensitivity, Specificity, and Precision, aiming to achieve the best possible result for our final goal in this study.

Introduction

Making business forecasts as a support to the management of a company is increasingly important nowadays. The constant growth and development of companies raises the bar and requires them to create new ways to counter and predict the movements of competitors, suppliers and consumers.

It is in this perspective that was asked to explore the "Predictive Maintenance" database, seeking to analyze a prediction problem, more specifically classification. In this same study, different algorithms will be used to evaluate the problem, associating it later to the chosen data set, in order to extract the best possible conclusions.

The goal of the project is to predict when a machine is likely to fail and what type of failure will occur, so that maintenance can be planned in advance and downtime can be avoided.

Database Analysis

The database used - Machine Predictive Maintenance Classification Dataset - consists of 10,000 observations and 10 variables/attributes, and has information about a simulated manufacturing process, namely sensor readings from industrial equipment and the machine state during operation. There is no missing value. The variables present in the database are:

- *UID:* Unique identifier ranging from 1 to 10000.
- *ProductID:* It contains in a letter L (low quality - 50% of all products), M (medium quality - 30%) or H (high quality - 20%) and a variant-specific serial number.
- *Type:* Based on the previous column, contains a letter between L, M or H.
- *Air.Temperature.K:* Generated using a random walk process subsequently normalized to a standard deviation of 2 K around 300 K.
- *Process.temperature.K:* Generated according to a random walk process normalized to a standard deviation of 1 K, added to the air temperature plus 10K.
- *Rotational.speed.rpm:* Calculated from 2860 W power, overlaid with normally distributed noise.
- *Torque.Nm:* The torque values are normally distributed around 40 Nm with Ï*f* = 10 Nm and no negative values.
- *Tool.wear.min:* The H/M/L quality variants add 5/3/2 minutes of wear to the tool used in the process.

- *Target:* Binary variable that indicates 0, when there is no failure, and 1, when there has been a failure.
- *Failure.Type:* Indicator of the type of fault that can be tool wear fault, heat dissipation fault, overload fault, power failure or random fault.

Database Processing

Before starting the analysis, the dataset was prepared in order to meet what is intended to evaluate and make the database more cohesive and of better quality. As such, we began by naming the table "db" and then entering the "str()" command in order to observe if the structure of the columns was correct.

Having said this, it was verified that the UID, ProductID, Type, Target and Failure.Type columns were incorrectly defined as "integer" and "character". Their structure was therefore changed to "factor" using the "lapply()" command. For the remaining variables, defined as "integer" and "numeric", they were changed exclusively to "numeric" using the same function as above.

Next, the possible existence of undefined values ("NA") was tested, introducing the "sum(is.na())" command, from which it was concluded that they were not present in the dataset.

Later, the Target columns and the Failure.Type column were studied. This verification was important for the analysis because, when the Target value is equal to zero, that is, there aren't supposed to be failures, there were some products presenting some "random failure", as well as when, for some cases whose Target values were equal to one, there were results stating the non-existence of failures ("no failure"). Observing these two errors, it was decided to change the "random failures" to Target values equal to one, exclusively, leaving the Target equal to zero as a synonym for no failure, as well as changing all "no failure" results to Target equal to zero.

Descriptive Analysis

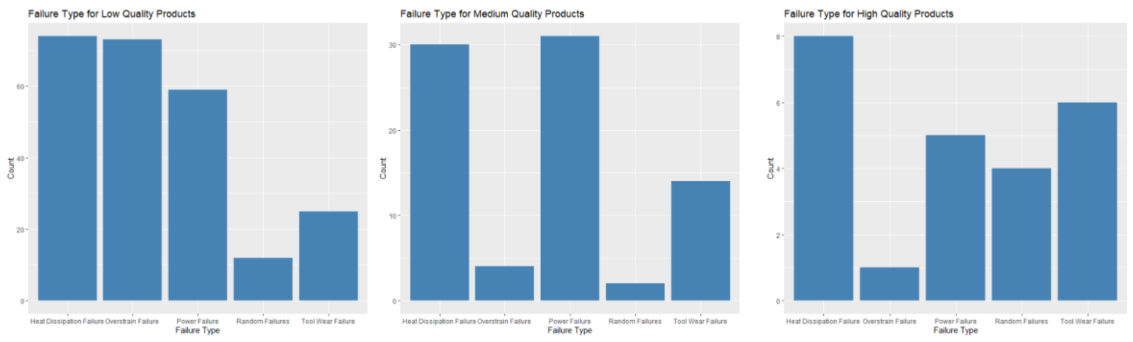
Once the database preparation was completed, a descriptive analysis of the observations was performed.

In a first instance, a statistical analysis of the numerical variables of the model was performed, and the mean, standard deviation, median, maximum and minimum were analyzed, as can be seen in Table 1.

db	**Mean**	**Standard Error**	**Median**	**Max**	**Min**
Air.temperature..K.	300,00493	2,000259	300,1	304,5	295,3
Process.temperature..K.	310,00556	1,483734	310,1	313,8	305,7
Rotational.speed..rpm.	1538,7761	179,284096	1503	2886	1168
Torque..Nm.	39,98691	9,968934	40,1	76,6	3,8
Tool.wear..min.	107,951	63,654147	108	253	0

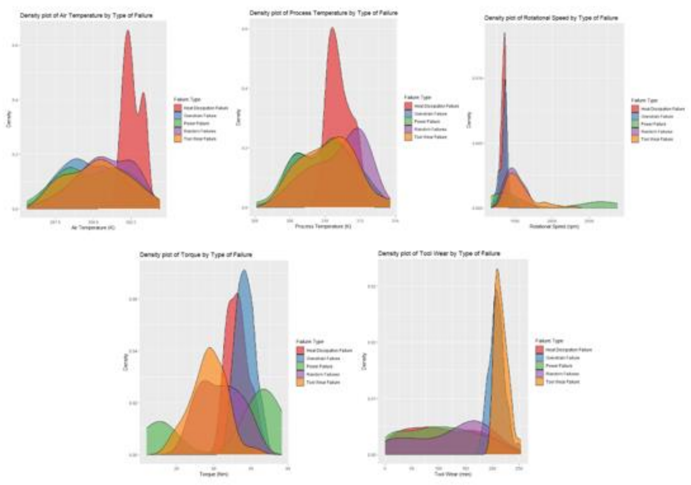
Table 1- Mean, standard error, median, maximum and minimum of variables

Next, the most recurrent failures were analyzed for the different types of product quality (high, medium and low quality).



Graphs 1,2,3 – Types of failures by quality of product

Through charts 1, 2 and 3, it was verified that the "heat dissipation failure" is recurrent for all three types of products. The "overstrain failure" is also a very present failure when the product is of low quality. Power failure, on the other hand, seems to be of high occurrence, especially in medium and low quality products. In order to understand the reasons for the presence of machine failures, density graphs were prepared relating the different numeric variables with the type of existing failures.



Graphs 4,5,6,7,8- Density Graphs associating numerical variables and the types of existing faults

Taking into account charts 4, 5, 6, 7 and 8, it is possible to conclude that failure in heat dissipation often occurs when the "Air Temperature" and the "Process Temperature" are relatively high, namely above 300 kelvins.

Regarding the rotational speed of the machine, there is a greater tendency for failure when it is below 1500 revolutions per minute (rpm). Regarding the "Torque", this factor presents a greater tendency for errors, which are present throughout its set of visualized values. However, there is a high peak between 40 and 60 newton-meters.

Finally, tool wear failure and overstrain failure appear to be significant when the machine usage time exceeds 125 minutes.

By analyzing Annex 1, which represents the correlations between the numerical variables, it is worth noting the positive relationship between "Air Temperature" and "Process Temperature", whose value is roughly 0.876. This correlation value indicates that the higher the air temperature, the higher the process temperature.

Finally, in order to make a brief analysis of the outliers present in the model, BoxPlots (Appendix 2) were produced for each of the numerical variables, highlighting the values of "Torque" and "Rotational Speed", since they were the only variables to present outliers. In order to analyze these values graphically, it was decided to create two statistics representing the outliers of both variables, entitled "outliers" and "outliers1", and then a scatterplot (Appendix 3) encompassing them. In the case of "Rotational Speed", the outliers began to appear near 2000 rpm. In the case of "Torque", they appeared for values below 20 and above 60 newton-meter.

Discussion and classification analysis

Since forecasting methods require the construction of a model in advance, we began by dividing the dataset "db" into two parts, with 70% allocated to train the model, and the remaining 30% to test it. Note that this division was done twice, i.e., first the dataset was divided taking into account the dependent variable Target, resulting in a "trainset" and a "testset" for this variable. Later the same was done for the dependent variable Failure.Type, where a 70% split for training and 30% for testing was obtained, resulting in the "trainset_FT" and the "testset_FT".

The first model aims to predict the target variable Target, in which it is intended to predict whether there will be an error (Target = 1) or not (Target = 0). The second model, on the other hand, aims to predict the Failure.Type class, that is, if there is an error and what type of error it would be.

For both models we used as independent variables all variables except, logically, the UDI, Product.ID, since they are only identifiers of each observation and do not influence the variables under study, and Failure.Type in the case of the first model, and Target in the case of the second model.

In order to deal with class imbalance of the Target and Failure.Type variables, the undersample method was used, applying the downsampling command from the "caret" package.

Class imbalance occurs when the target variables are disproportionately represented, i.e. one class has significantly more instances than another. This can be problematic when training predictive models, as the model may be biased toward the majority class and have difficulty capturing patterns from the minority class.

For both variables the downsampling command was used in order to reduce the proportion of instances of Target class 0, while for the Failure Type variable the aim was to reduce the proportion of instances of the majority class "No Failure".

Regarding the algorithms used for the classification analysis, they are the K-Nearest Neighbor (KNN), Naive Bayes, Decision Tree and Random Forest.

K-Nearest Neighbor

The k-NN algorithm classifies the test object under analysis according to the mode of the k closest training objects. In the case of this algorithm, normalized data was used, changing categorical variables to numeric ones, using the "fastDummies" package, in order to be employed by the algorithm. This is because kNN is a distance-based algorithm, which means that it calculates the similarity between data points based on their distances in a multidimensional space, so categorical variables (in this case the Type variable), cannot be used directly in their original form, since they do not have a natural distance metric. Therefore, it is necessary to convert categorical variables into numerical representations using dummy variables.

When applying the algorithm, the problem of what would be the optimal number of neighbors to analyze (k) was encountered. After trial and error, it was realized that 2 was the number of neighbors to analyze for the Target and Failure.Type variables, because they are the ones that allow obtaining a higher accuracy (effectiveness), of about 89.07% and 62.38%, respectively. These values provide the percentage of predictions that were correct.

Three other metrics were also measured for the Target variable, namely sensitivity, specificity and precision, which presented values of 90.16%, 59.05% and 98.38% respectively.

Sensitivity is a metric that measures the proportion of positive instances correctly identified by the model in relation to the total number of positive instances in the dataset. In this case, the model correctly identified approximately 90.16% of the positive instances relative to the total number of positive instances.

In turn, specificity is a metric that measures the proportion of negative instances correctly identified by the model relative to the total negative instances in the dataset. In this case, the model correctly identified approximately 59.05% of the negative instances out of the total negative instances.

Finally, precision is a metric that measures the proportion of instances correctly classified as positive (true positives) relative to the total instances classified as positive (true positives + false positives). In this model, approximately 98.38% of the instances classified as positive are true positives.

For the variable Failure.Type, the remaining metrics presented the results present in Table 2.

Classe	***Sensitivity***	***Specificity***	***Precision***
Heat Dissipation Failure	52,94%	89,01%	5,23%
No Failure	63,78%	82,86%	99,02%
Overstrain Failure	65,22%	96,94%	14,15%
Power Failure	62,07%	95,29%	11,39%
Random Failures	0%	85,34%	0%
Tool Wear Failure	23,08%	96,15%	2,54%

Table 2 - KNN metrics for the variable Failure.Type

Since, for this and the other algorithms, a balanced data set will be being evaluated, accuracy alone may not be the best indicator of model performance. In such cases it is often useful to consider additional metrics that provide a more comprehensive assessment such as precision and specificity.

Naive Bayes

The Naive Bayes algorithm, based on the Bayesian theorem, predicts the class of objects under analysis and the hypothesis with the maximum probability of happening. Regarding the evaluation metrics, the previous metrics were observed again to compare the two algorithms.

Through this method, an accuracy of approximately 85.1% was obtained for the model regarding the Target variable, and 46.62% for the model regarding the Failure.Type variable. Comparing these values to the previous algorithm, one can verify that they are slightly lower for the Target variable and much lower for the Failure.Type variable, i.e., less efficient.

We also checked the remaining metrics of this model for the Target variable, presenting a value of 85.53% for sensitivity, 73.34% for specificity and 98.89% for precision. This means that the model correctly identified 85.53% of positive instances relative to total positive instances, approximately 73.34% of negative instances relative to total negative instances, and about 98.89% of instances classified as positive are truly positive.

For the variable Failure.Type, the remaining metrics showed the results in Table 3.

Classe	*Sensitivity*	*Specificity*	*Precision*
Heat Dissipation Failure	91,18%	95,72%	19,62%
No Failure	45,2%	99,05%	99,92%
Overstrain Failure	91,3%	97,35%	21%
Power Failure	86,2%	91,59%	9,09%
Random Failures	50%	70,48%	0,34%
Tool Wear Failure	76,92%	91,27%	3,69%

Table 3 - Naive-Bayes metrics for the variable Failure.Type

Decision Tree

In addition to the two algorithms analyzed above, we also used the analysis of decision trees for the two independent variables and their different models. Decision trees represent sets of decisions that create rules for classifying a set of data. The decision tree is built in two stages, these being the construction and the "trimming" of the tree. At the beginning of its construction, all training examples are found at the root. In addition, branches reflecting "noise" or outliers are identified and removed when the tree is "trimmed".

Since this model can be run based on two splitting criteria, i.e., information gain and the Gini Index, it was thought relevant to analyze the two methods in order to later compare the results obtained.

Before building the decision trees, the packages "rpart.plot", "rpart" and "rattle" were activated, starting the construction of the model. We began by building it according to the information gain criterion, using the "repart" function, considering a minimum number of observations in a node and in a leaf equal to 4 and 3, respectively, for the two dependent variables. Next, the originated tree was verified using the command "fancyRpartPlo "t, as shown in Figure 1 and Annex 4. Finally, the tree was pruned using the complexity parameter, according to the "prune" function, obtaining Annex 5 and Figure 2.

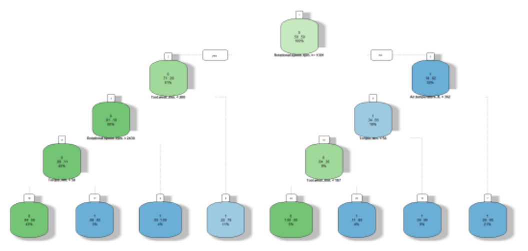


Figure 1. Target's Decision Tree according to the information gain criterion

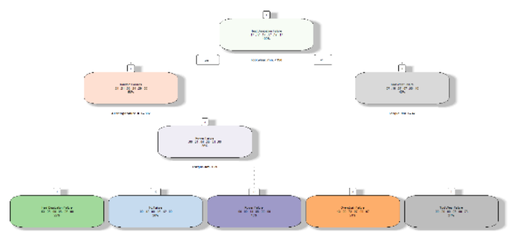


Figure 2: Pruned Decision Tree of the variable Failure Type according to the information gain criterion.

According to the prediction made from the Target variable (Figure 1), it is visible that the probability of no failure for a "rotational speed" greater than or equal to 1381 revolutions per minute is 61%. After the analysis of this branch, we proceeded to the analysis of the remaining branches, with the example that the probability of no failure occurring when the "tool wear" is less than 200 minutes is 50%.

Furthermore, the same procedure is carried out for the variable Failure.Type (Figure 2), that is, all the branches are analyzed. Starting then with the first branch, the probability of occurrence of a "Random Failure" when the tool wear is less than 190 minutes is 58%, while the probability of occurrence of a "Tool Wear Failure", when the previous condition does not happen, is 42%. As with the Target variable, this analysis continues until all the ramifications are explained. With this, it is concluded that each of the ramifications reflects certain attributes that act as criteria for conclusions.

According to this model, regarding the Target variable, its accuracy before and after being pruned is 88.47% and 81.94%, respectively. With regard to the sensitivity metric, this model also presents a better result before pruning, equal to 88.23%, and after pruning equal to 81.66%. The same is not true for the specificity metric, where the value before pruning is 88.23%, and after pruning it is 89.52%. Precision, on the other hand, presents a value of 99.54% after "trimming" the tree, while before it had a value of 99.80%.

Regarding the dependent variable Failure.Type, the model presents a better accuracy after being trimmed, equal to 66.34%. Moreover, the classes concerning "Overstrain Failure" and "Tool Wear Failure" present the highest sensitivity values, 91.30% and 92.31%, as opposed to the "Random Failures" class, which presents a null value. On the other hand, the opposite happens when the specificity of the "Random Failures" class is analyzed. Finally, the class that presents the highest precision is the "No Failure" class, with a value of 99.17%.

Next, the decision trees were drawn (Figures 3 and 4) according to the second criterion, that is, the gini index that is inserted in the CART algorithm. According to this criterion, the best attribute to start the division is the one where the sum of all gini indexes of each set is smaller.

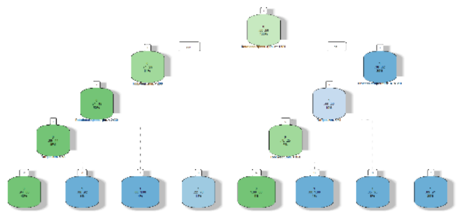


Figure 3: Decision tree of the Target variable according to the gini index criterion.

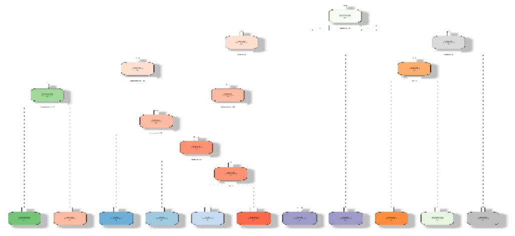


Figure 4. Decision tree of the variable Failure Type according to the gini index criterion.

As previously done with the information gain criterion, the subsequent decision trees were pruned, and Annexes 6 and 7 were obtained. Performing an analysis like the previous one, in the case of the Target variable, the probability of there not being any failure when the "rotational speed" is greater than or equal to 1381 rpm is equal to 61%, otherwise, the probability of there being any failure is 39%. In relation to the variable Failure Type, the probability of "Random Failures" given that the tool wear is less than 190 minutes is 58%. This process continues until the analysis of all ramifications is completed.

According to the model referring to the gini index criteria for the Target variable, it can be concluded that the decision tree has a higher accuracy value before pruning, equal to 86.30%. Besides this, the sensitivity value is higher in the same situation, equal to 85.98%. The values of the specificity and precision metrics correspond to 95.24% and 99.80%, respectively, also in the same context.

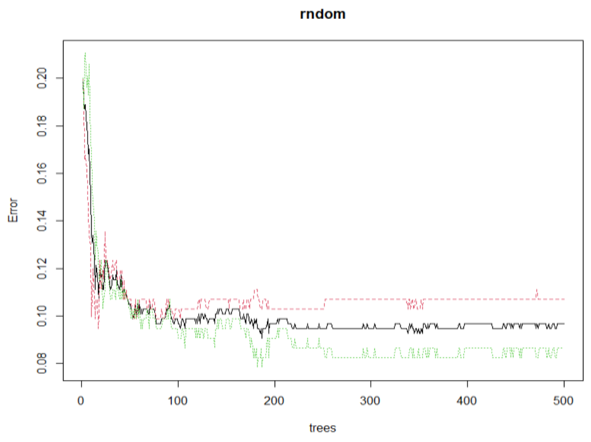
Finally, regarding the model for the variable Failure Type, it presents a lower accuracy after pruning the decision tree, in the value of 27.62% in comparison with an accuracy of 41.95% before pruning. According to the results obtained before trimming the decision tree, and as observed by the information gain method, the "No Failure" class presents the lowest sensitivity value, equal to 40.57%. Besides this, the "Tool Wear Failures" class presents the highest sensitivity value, equal to 92.31%, and the "Power Failure" class presents the highest specificity value, equal to 99.43%. The remaining classes don't present a great discrepancy in relation to the previously mentioned values. Regarding the precision metric, the classes show a great discrepancy between them, being the "No Failure" class the one with the highest value, 99,75%, while the "Random Failures" class presents the lowest value, equal to 0,28%.

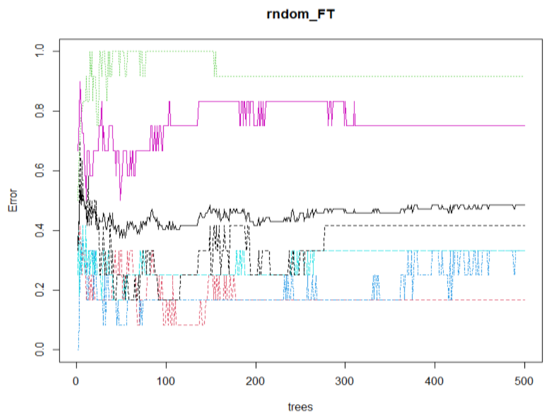
Thus, it can be concluded that for the Target variable, the most accurate decision tree method is when the criterion by information gain is applied. However, both the model created based on the information gain criterion and the gini index criterion do not present very different results. In turn, the criterion that best draws conclusions for variable Failure.Type according to the decision trees algorithm is the one by information gain, after the pruning has been done..

<u>Random Forest</u>

Finally, as mentioned earlier, the Random Forest algorithm was used. This classifier is of high importance since it is able to handle a large number of data and observations and provide highly accurate results.

As a first task, we started by creating a classifier called "rndom" in order to "train" the data for the model. The model aims to predict the variable Class based on the other variables in the balanced dataset linked to Target, only excluding the following: Product.ID, Target, and Failure.Type, because the Class variable presents the same values as the Target variable. Through statistical analysis it was found that the estimate of the "out-of-bag" (OOB) error rate was 9.67%, which means that the model correctly classified 90.33% of the instances in the training dataset, since this type of error is an estimate of the model's prediction error on unseen data.





Graph 9 and 10- Plot "rndom" and "rndom_FT

It can be seen from graph 9, that the model consists, by default, of 500 trees, each having four attributes. The plot shows three distinct lines, where the red line corresponds to the average decrease in precision, the green line corresponds to the average decrease in the Gini index and the black line indicates the standard error of each measure.

Next, a classifier labeled "rndom_FT" was created using the balanced database linked to Failure.Type. In this case, it can be seen that from a statistical standpoint, the OOB error rate estimate was 48.61%, indicating that on average, the model is expected to make incorrect predictions for approximately 48.61% of the observations in the dataset that were not used during "training".

According to graph 10, the most important lines for analysis are: the green line represents the OOB error rate of the model as a function of the number of trees in the forest. The purple line expresses the model error rate on the training data, and the black line represents the overall error rate, which combines the out-of-bag error rate and the error rate on the training data. Therefore, it can be seen from the two graphs that, tendentially, the larger the number of trees, the lower the error rate present.

Subsequently, two confusion matrices were created for two "predict" functions using the two classifiers as a function of the testset. Thus, it was possible to ascertain the accuracy, sensitivity, specificity, and precision.

For rndom, the accuracy value was 0.889, which means that 88.9% of the predictions were made correctly. Furthermore, the sensitivity is given as 0.8867 (88.7%). The specificity was around 0.9524 (95.24%) and the precision is 0.9981 (99.81%).

In the case of "rndom_FT", the accuracy value was around 0.5062 (50.62%). Since the remaining indicators have different values for the different types of failures, it was decided to choose the ones with higher results. Starting with sensitivity, "Heat Dissipation Failure" was the class that presented the most relevant values, of 0.9705 (97.05%). The specificity values were higher for "Overstrain Failure" (98.12%) and, regarding precision, "No Failure" (99.65%) was the most relevant category.

Conclusion

In conclusion, in order to find out which models are more accurate for the dependent variables studied, we compared the results of the metrics between the different models for a given variable. Table 4 compares the accuracy values for the different algorithms, according to the analyzed model. The metrics accuracy, sensitivity, specificity and precision were analyzed in order to make a more precise prediction about each model. Furthermore, the sample was first divided into four datasets, in which 70% of the data corresponds to the training sample and the remaining 30% to the test sample, according to the corresponding dependent variable. Due to the fact that the database was not balanced, the undersampling method was used to balance the training sample, applying the downSample command to the training sample concerning the Target variable and applying the same command to the training sample concerning the Failure.Type variable, as explained above.

Regarding the Target variable model, we conclude that the kNN algorithm, for a k equal to 2, and the Random Forest algorithm perform a similarly accurate prediction for the Target model, with the kNN model having a slightly higher accuracy than the second algorithm, being equal to 89.07% and 88.90%, respectively. However, the Random Forest algorithm obtained a better prediction in the sensitivity, specificity and precision metrics, equal to 88.7%, 95.24% and 99.81%, respectively, compared to 90.16%, 59.05% and 98.38%, in the same order, for the kNN-related model.

In what concerns the model of variable Failure.Type, we conclude that the pruned decision tree algorithm, using the criterion by information gain, is the one that performs a more accurate prediction, having an accuracy of 66.34%. On the other hand, the pruned decision tree algorithm, according to the gini index criterion, is the one that performs a less realistic forecast, having an accuracy equal to 27.62%..

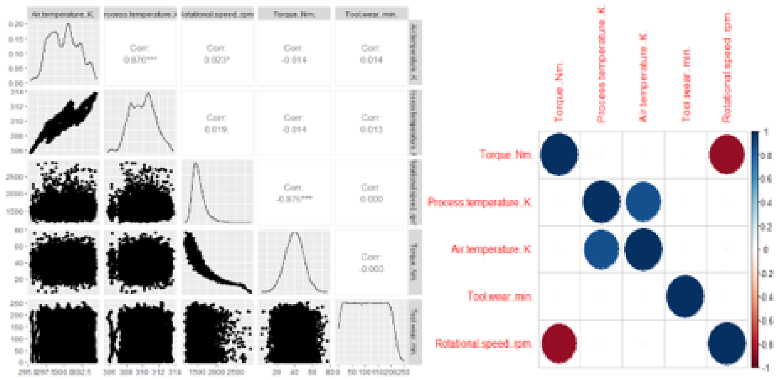
	Accuracy - Target	***Accuracy - Failure Type***
kNN	**0,8907**	0,6238
Naive Bayes	0,8510	0,4662
Árvore de Decisão - ganho de informação	0,8847	0,3366
Árvore de Decisão Podada - ganho de informação	0,8194	**0,6634**
Árvore de Decisão - *gini index*	0,8630	0,4195
Árvore de Decisão Podada - *gini index*	0,8194	0,2762
Random Forest	**0,8890**	0,5062

Table 4: Comparative table of Accuracy for each algorithm, according to the analyzed model.

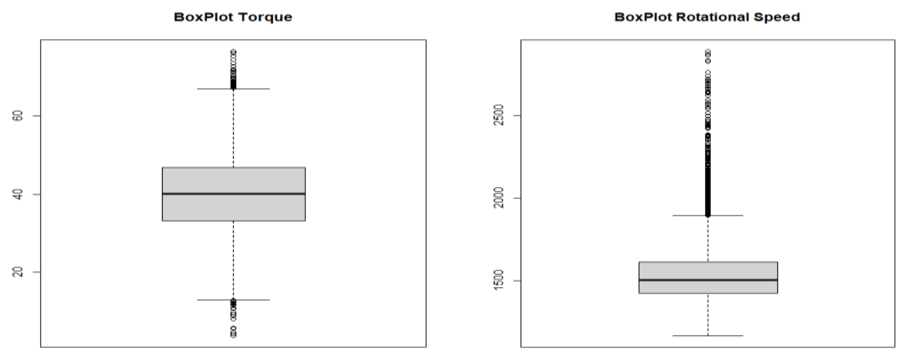
As the objective of the forecasts made is to predict the occurrence or not of some machine failure, using the Target dependent variable as reference, and what type of failure occurs, through the Failure.Type variable, this can be beneficial for management. In this way, the company can benefit from these forecasts to plan its production plan and try to solve the problem before a failure occurs, thus gaining a competitive advantage over other companies.

<u>Appendix</u>

Appendix 1 - Correlation Graphs.

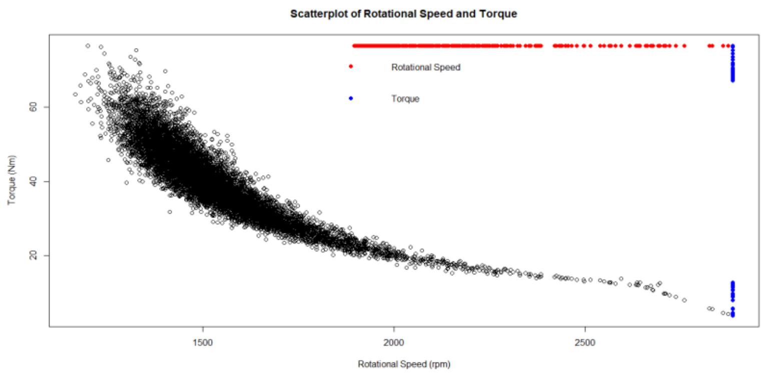


Appendix 2 - BoxPlots of Rotational Speed and Torque, respectively.

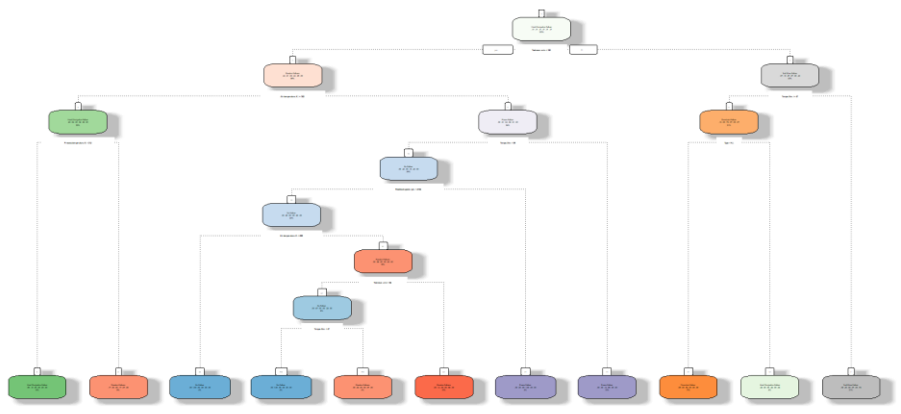


~

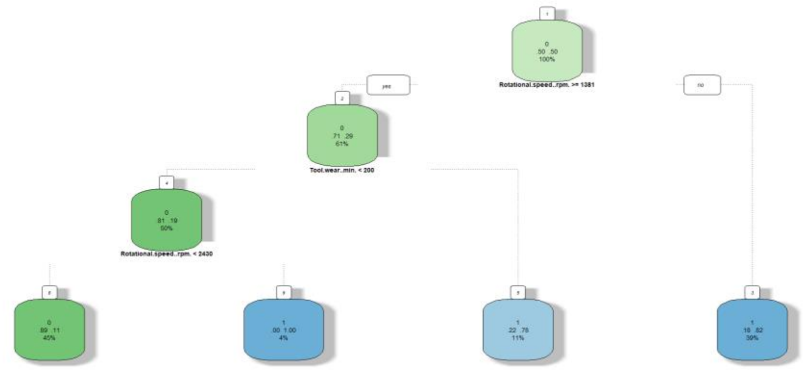
Appendix 3 - ScatterPlot to analyze Outliers of Rotational Speed and Torque.



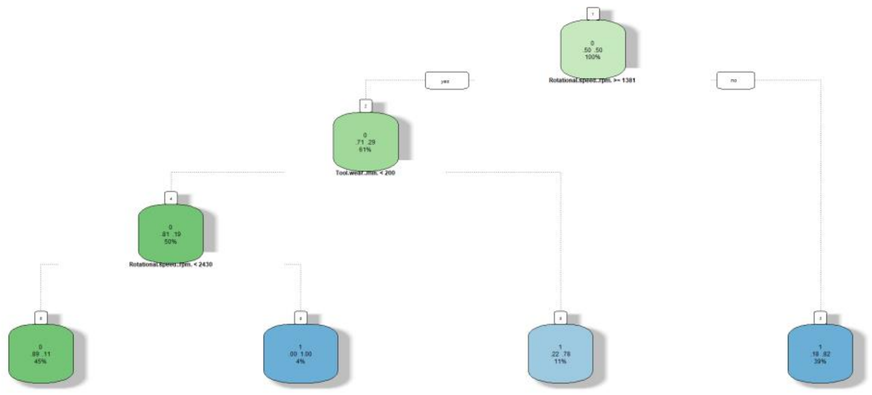
Appendix 4 - Failure Type Decision Tree, according to the information gain criterion.



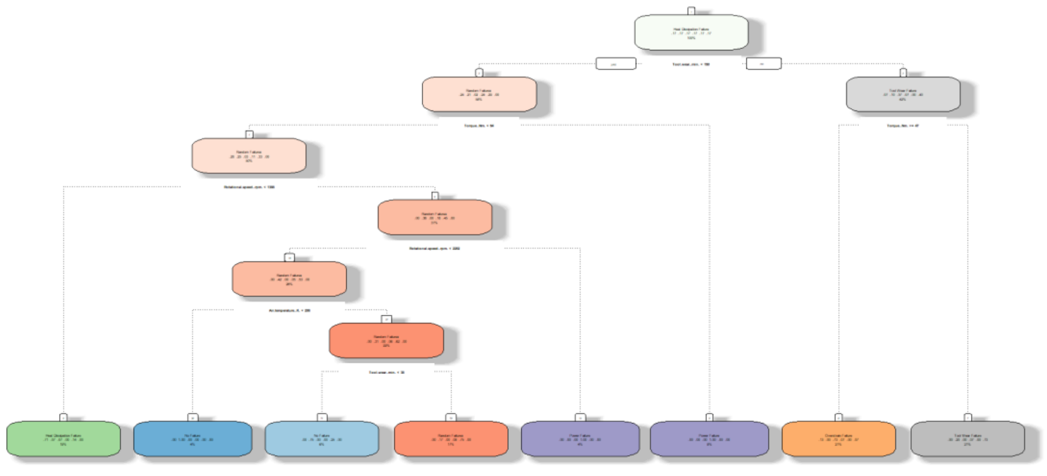
Appendix 5 - Pruned Decision Tree of the Target variable, according to the information gain criterion.



Appendix 6 - Pruned Decision Tree of the Target variable, according to the gini index criterion.



Appendix 7. Pruned decision tree of the variable Failure Type according to the gini index criterion.


Heat Dissipation Failure
Random Failure
Tool Wear Failure
Random Failure
Random Failure
Random Failure
Random Failure
Heat Dissipation Failure
No Failure
No Failure
Random Failure
Power Failure
Power Failure
Overstrain Failure
Tool Wear Failure