UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

ALGORITHMS AND APPLICATIONS
FOR
DATA SCIENCE

PROFESSOR F. PICCIALLI

Project Report

Predictive Maintenance

Sara Amitrano
Gerardo Cappa
Angela Sarnataro

Academic year 2021-2022

Introduction

This work is placed in the field of Data Science with application to the area of predictive maintenance. The need to have a way to determine whether or not a particular machine will fail, as well as the nature of the failure, is essential for generation 4.0 industries. The main reason lies behind the following consideration: the repair or replacement of a faulty machine generally requires costs that are much higher than those required for the replacement of a single component. Therefore, the installation of sensors that monitor the state of the machines, collecting the appropriate information, can lead to great savings for industries.

Here we use the *AI4I Predictive Maintenance Dataset* from the UCI Repository to carry out an analysis that aims to respond to the needs just reported. In particular, the work is presented through a lineup that characterize a typical Machine Learning application. In the first place the dataset is explored to obtain a deeper knowledge that can guide in fully understanding the ground truth. Then, some preprocessing techniques are applied to prepare the data for the algorithms we will use to make our predictions. We consider two main tasks: the first consists in establishing whether a generic machine is about to suffer a failure while the second concerns the determination of the nature of the fault. Finally, a comparison is provided between the results obtained by the latter, evaluating both their performance through appropriate metrics, and their interpretability.

Contents

1 Task and dataset description

Since real predictive maintenance datasets are generally difficult to obtain and in particular difficult to publish, the data provided by the UCI repository is a synthetic dataset that reflects real predictive maintenance encountered in industry to the best of their knowledge.
The dataset consists of 10 000 data points stored as rows with 14 features in columns:

- UID: unique identifier ranging from 1 to 10000;
- Product ID: consisting of a letter L, M, or H for low (60% of all products), medium (30%) and high (10%) as product quality variants and a variant-specific serial number;
- Air temperature [K]: generated using a random walk process later normalized to a standard deviation of 2 K around 300 K;
- Process temperature [K]: generated using a random walk process normalized to a standard deviation of 1 K, added to the air temperature plus 10 K;
- Rotational speed [rpm]: calculated from a power of 2860 W, overlaid with a normally distributed noise;
- Torque [Nm]: torque values are normally distributed around 40 Nm with a standard deviation of 10 Nm and no negative values;
- Tool wear [min]: The quality variants H/M/L add 5/3/2 minutes of tool wear to the used tool in the process;
- Machine failure: label that indicates, whether the machine has failed in this particular data point for any of the following failure modes are true.

The machine failure consists of five independent failure modes:

- tool wear failure (**TWF**): the tool will be replaced of fail at a randomly selected tool wear time between 200 - 240 mins;
- heat dissipation failure (**HDF**): heat dissipation causes a process failure, if the difference between air- and process temperature is below 8.6 K and the tools rotational speed is below 1380 rpm;

- power failure (**PWF**):the product of torque and rotational speed (in rad/s) equals the power required for the process. If this power is below 3500 W or above 9000 W, the process fails;
- overstrain failure (**OSF**): if the product of tool wear and torque exceeds 11,000 minNm for the L product variant (12,000 M, 13,000 H), the process fails due to overstrain;
- random failures (**RNF**): each process has a chance of 0,1 % to fail regardless of its process parameters.

If at least one of the above failure modes is true, the process fails and the 'machine failure' label is set to 1. It is therefore not transparent to the machine learning method, which of the failure modes has caused the process to fail.

2 Exploratory Analysis

Our data exploration starts by checking that each entry is unique and there are no duplicates; this is done by veryfing that the number of unique ProductID corresponds to the number of observations. Then we print a report to look for missing values and check the data type for each column.

```
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   UDI                     10000 non-null  int64
 1   Product ID              10000 non-null  object
 2   Type                    10000 non-null  object
 3   Air temperature [K]     10000 non-null  float64
 4   Process temperature [K] 10000 non-null  float64
 5   Rotational speed [rpm]  10000 non-null  int64
 6   Torque [Nm]             10000 non-null  float64
 7   Tool wear [min]         10000 non-null  int64
 8   Machine failure         10000 non-null  int64
 9   TWF                     10000 non-null  int64
 10  HDF                     10000 non-null  int64
 11  PWF                     10000 non-null  int64
 12  OSF                     10000 non-null  int64
 13  RNF                     10000 non-null  int64
dtypes: float64(3), int64(9), object(2)
```

Figure 1: Columns report

To sum up even more:

- There is no missing data;
- There are no duplicate values;
- Six columns are numerical features, including UDI;
- Six columns are categorical features, including PrductID;
- Six columns are binary and represent the OneHotEncoded targets.

To make this distinction more clear the first numeric columns have been set to float type.

2.1 ID columns

Before going into more technical matters we deal with the two ID columns as the model we will use could get confused by them, since it is unrealistic to think that the failure of a machine depends on its identifier. However, while UDI results in being a copy of the dataframe index,

the column Product ID is made up of an initial letter followed by five numbers; there is a small chance that an hidden pattern lies behind this structure. However, the initial letter corresponds to the machine Type and the number sequences define three intervals based on the same feature; this allows to confirm that the Product ID column does not actually carry any more information than the feature Type and it is legit to drop it.

The following histogram shows the number sequences:

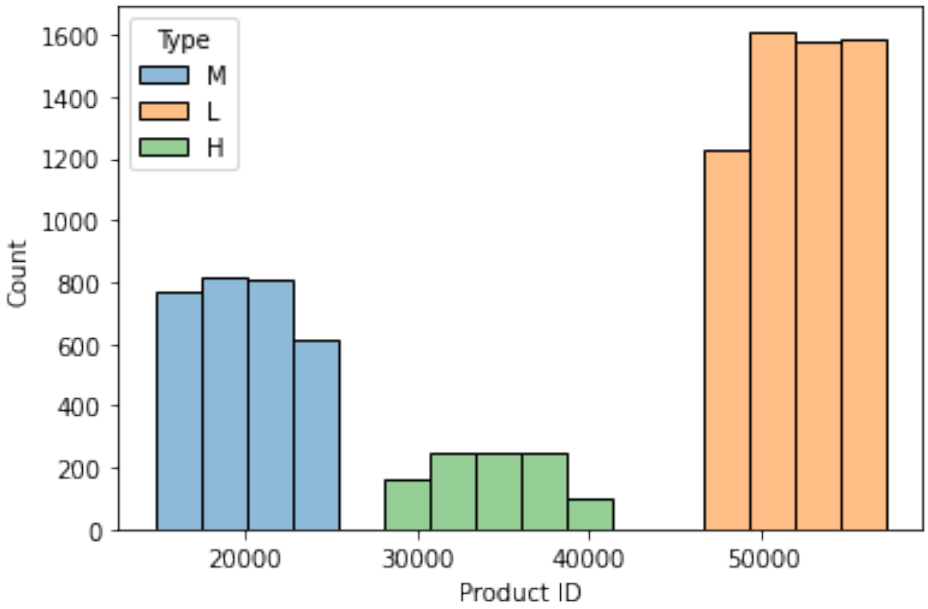


Figure 2: Type sequences

The following pie chart shows the percentages of machines by Type:

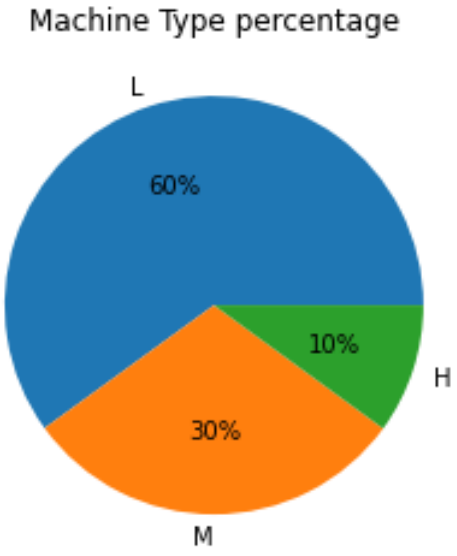


Figure 3: Percentages of types

- Type Low (L): 60%
- Type Medium (M): 30%
- Type High (H): 10%

2.2 Anomalies

In this section we observe the distribution of the target to find any imbalances and correct them before dividing the dataset.
The first anomaly respect to dataset's description is that when the failure is random (RNF), the Machine Failure feature is not set to 1.

	Machine failure	TWF	HDF	PWF	OSF	RNF
1221	0	0	0	0	0	1
1302	0	0	0	0	0	1
1748	0	0	0	0	0	1
2072	0	0	0	0	0	1
2559	0	0	0	0	0	1
3065	0	0	0	0	0	1
3452	0	0	0	0	0	1
3611	1	1	0	0	0	1
5471	0	0	0	0	0	1
5489	0	0	0	0	0	1
5495	0	0	0	0	0	1
5509	0	0	0	0	0	1
5553	0	0	0	0	0	1
5639	0	0	0	0	0	1
6091	0	0	0	0	0	1
6913	0	0	0	0	0	1
6960	0	0	0	0	0	1
7488	0	0	0	0	0	1
7868	0	0	0	0	0	1

Figure 4: RNF anomaly

Fortunately the machine failure RNF occurs in only 19 observations and it has a random nature therefore not predictable so we decide to remove these rows and secondly the entire RNF column, because it will contain only zeros.

Going forward we find out that in 9 observations Machine failure is set to 1 when all types

of failures are set to 0. We cannot understand if there really was a failure or not so let's remove these observations too.

	Machine failure	TWF	HDF	PWF	OSF
1437	1	0	0	0	0
2749	1	0	0	0	0
4044	1	0	0	0	0
4684	1	0	0	0	0
5536	1	0	0	0	0
5941	1	0	0	0	0
6478	1	0	0	0	0
8506	1	0	0	0	0
9015	1	0	0	0	0

Figure 5: Machine failure anomaly

Lastly, a multiple failure occurs in 23 observations, which means that more than one cause of failure is specified for those instancies. Here we make an important assumption, which consist in stating that each machine fault can correspond to only one cause. In this case, these observations result to be one more time ambiguous and will be dropped.

	Machine failure	TWF	HDF	PWF	OSF
69	1	0	0	1	1
1324	1	0	0	1	1
1496	1	0	0	1	1
3854	1	0	0	1	1
3943	1	0	0	1	1
4254	1	0	1	1	0
4342	1	0	1	1	0
4370	1	0	1	0	1
4383	1	0	1	0	1
4417	1	0	1	1	0
4462	1	0	1	0	1
4642	1	0	1	0	1
4643	1	0	1	0	1
4729	1	0	1	0	1
5394	1	0	0	1	1
5401	1	1	0	0	1
5909	1	1	0	1	1
6248	1	0	0	1	1
7083	1	0	0	1	1
8846	1	1	0	0	1
8926	1	0	0	1	1
9084	1	0	0	1	1
9974	1	0	0	1	1

Figure 6: Multiple failure

The global percentage of removed observations is 0.51%.

2.3 Outliers inspection

The goal of this section is to check if the dataset contains any outlier, which are usually misleading for machine learning algorithms. We begin by looking at a statistical report of the numerical features.

	Air temperature	Process temperature	Rotational speed	Torque	Tool wear	Machine failure
count	9949.000000	9949.000000	9949.000000	9949.000000	9949.000000	9949.000000
mean	300.000925	310.003508	1539.385164	39.921329	107.716554	0.030757
std	1.999866	1.484192	179.321578	9.903881	63.559923	0.172667
min	295.300000	305.700000	1168.000000	3.800000	0.000000	0.000000
25%	298.300000	308.800000	1424.000000	33.200000	53.000000	0.000000
50%	300.100000	310.100000	1504.000000	40.000000	108.000000	0.000000
75%	301.500000	311.100000	1612.000000	46.700000	162.000000	0.000000
max	304.500000	313.800000	2886.000000	76.600000	251.000000	1.000000

Figure 7: Statistics

We can guess the presence of outliers in Rotational Speed and Torque because the maximum is very different from the third quartile. To make this consideration more concrete we take a closer look at the situation with boxplots, using histograms to understand the distribution.

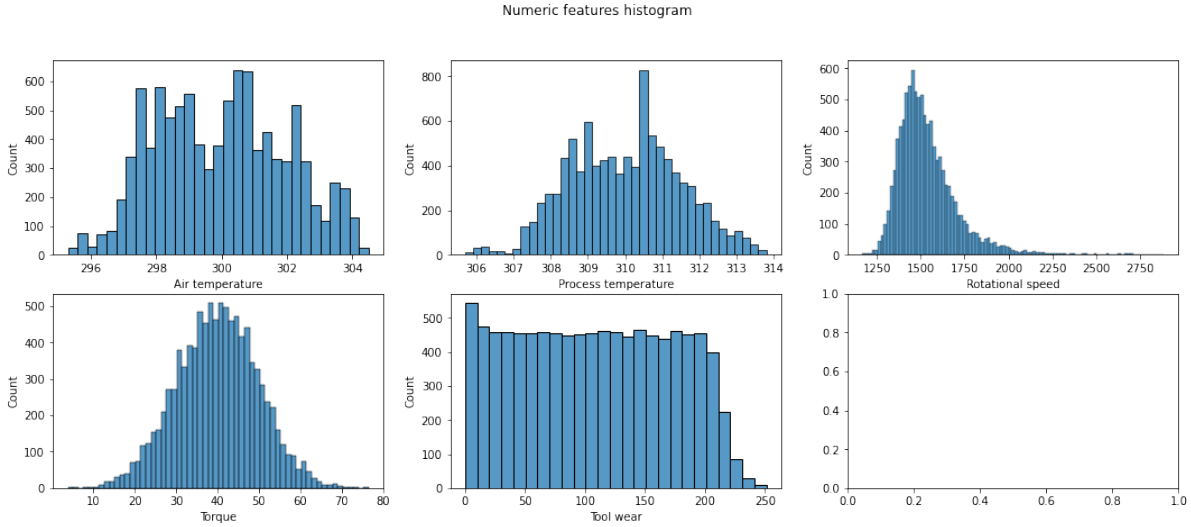


Figure 8: Features histogram

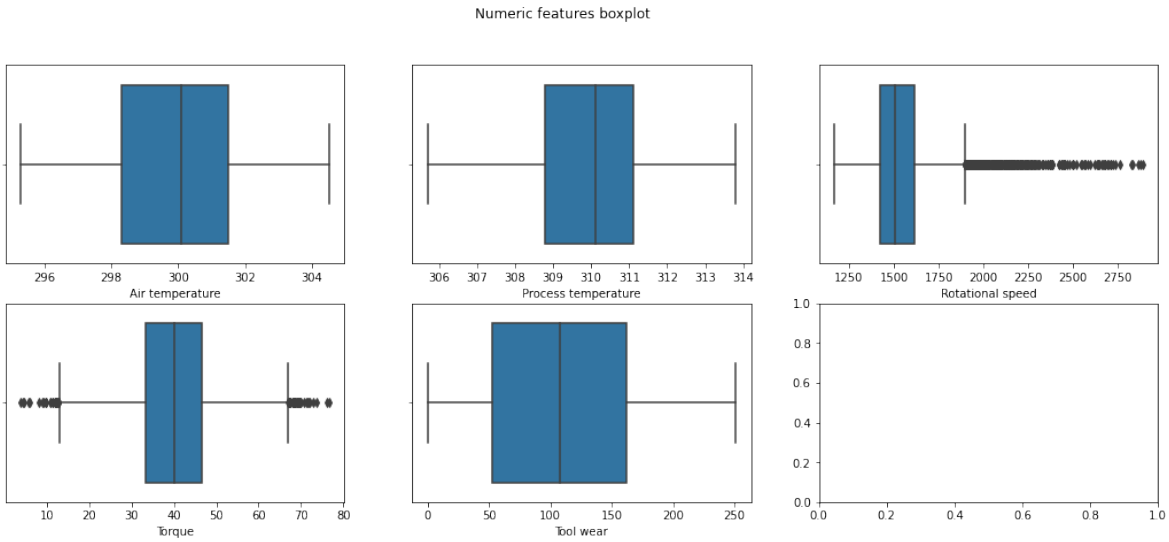


Figure 9: Boxplots

The boxplots highlight possible outliers in the features mentioned above, however in the case of Torque these are probably traceable to the way outliers are detected using boxplots (since the distribution is Gaussian it would be more appropriate to use the 3σ rule instead of the IQR); in the case of Rotational Speed the Gaussian distribution is skewed and it is not unrealistic to think that the few observation with high Rotational Speed are going to fail. As a result we keep the outliers for now and we reserve the right to decide whether to act on them or not after considering other aspects.

2.4 Data balancing

After solving the problems related to the consistency of the target, we perform an inversion of OneHotEncoding, mapping the four binary columns, that indicate the causes of the Machine failure, in a single categorical column that indicates the causes. This step is useful for the proper functioning of the multi-class classification algorithms that we will run next.

Another important consideration regards the extremely low occurrence of machine failures among the entire dataset, which percentage is equal only to 3.08%. Moreover, a pie plot showing the occurrence of the causes involved for each failure reveals a further degree of imbalance.

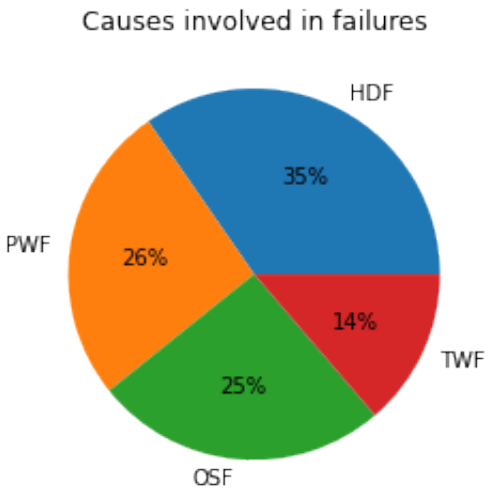


Figure 10: Failure causes

When dealing with machine learning problems classes imbalance is a great concern, as it can mislead both the models training process and our ability to interpret their results. As instance, if we build a model on this dataset that predicts that machines never fail, it should be 97% accurate. In order to avoid such effects and limit the preferential behaviour of the models with respect to individual classes we perform a data augmentation, with the aim of obtaining a ratio of 80 to 20 between functioning and faulty observations and the same percentage of occurrence between the causes involved in the failures.

Among the most common data augmentation techniques we identify:

- Under-sampling by deleting some data points from the majority class.
- Over-Sampling by copying rows of data resulting in the minority class.
- Over-Sampling with SMOTE (Synthetic Minority Oversampling Technique).

The first two choices however result in extremely simplistic approaches; in particular the first one has the disadvantage of decreasing the length of the dataset in a context in which the available data are already limited. Therefore we use the SMOTE procedure to generate new samples, which is very much like slightly moving the data point in the direction of its neighbors. This way, the synthetic data point is not an exact copy of an existing data point but we can also be sure that it is also not too different from the known observations in the minority class. To be more precise, the SMOTE procedure works as follows: it draws a random sample from the minority

class and for the observations in this sample, identifies the k nearest neighbors. It will then take one of those neighbors and identify the vector between the current data point and the selected neighbor. The vector will be multiplied by a random number between 0 and 1 and the synthetic data point is obtained by adding this vector to the current data point.

The result is described in the following pie charts.

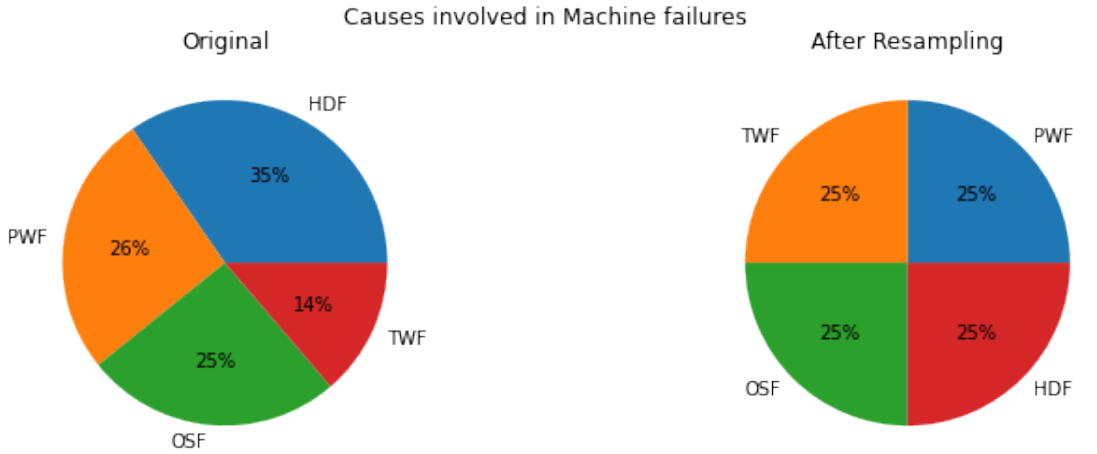


Figure 11: Failure cause rates

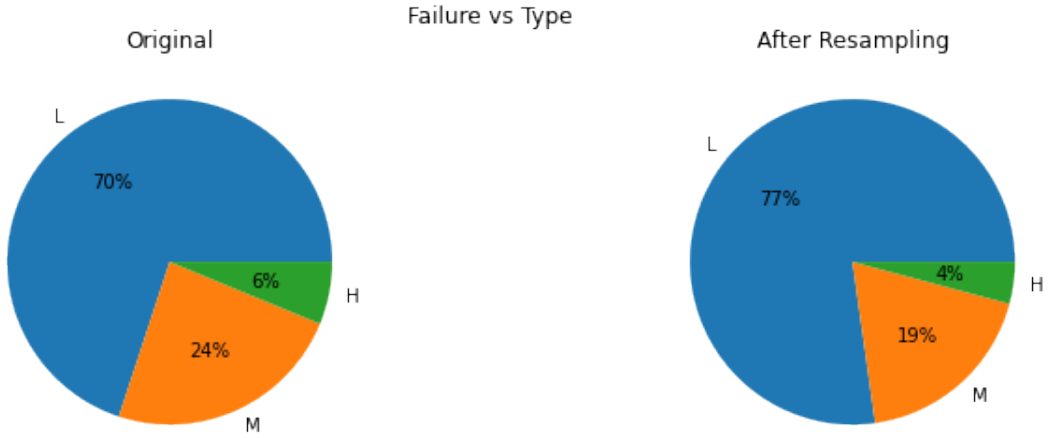


Figure 12: Failure vs type

As one can expect, the cases of Machine Failure mainly concern low quality machines, then those of medium quality and only a few times those of high quality. This difference is accentuated when the number of observations of non-functioning machines is (artificially) increased. However, from the kdeplots below it can be seen that this is not widely correlated with the features since differentiating according to the quality shows that distribution of the features does not present big differences, except for the two side peaks in Tool Wear (which is consistent with the data

description). This suggests that probably the fact that the majority of failures concern type L machines is due to the greater presence of this type in the dataset and therefore that the correlation with the failure of the machine is due to statistical reasons.

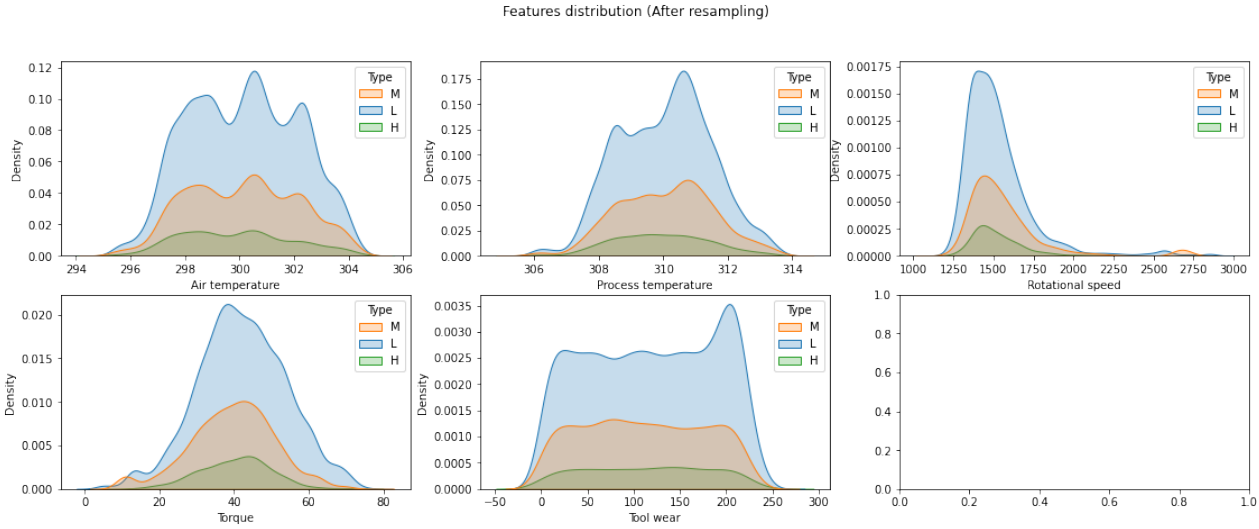


Figure 13: Features distribution

Finally, let's look at how the distribution of features has changed.

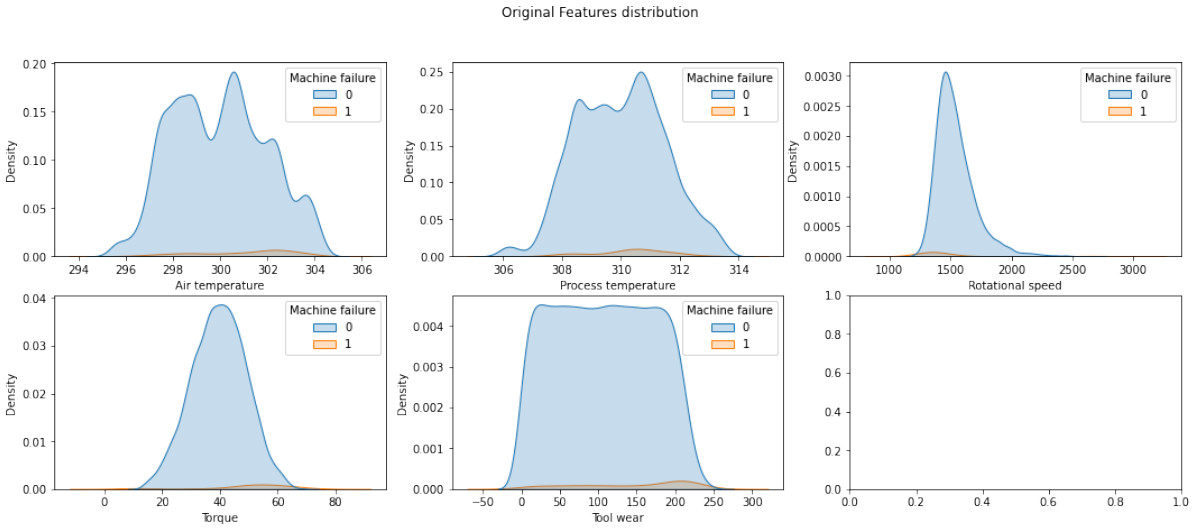


Figure 14: Original features distribution

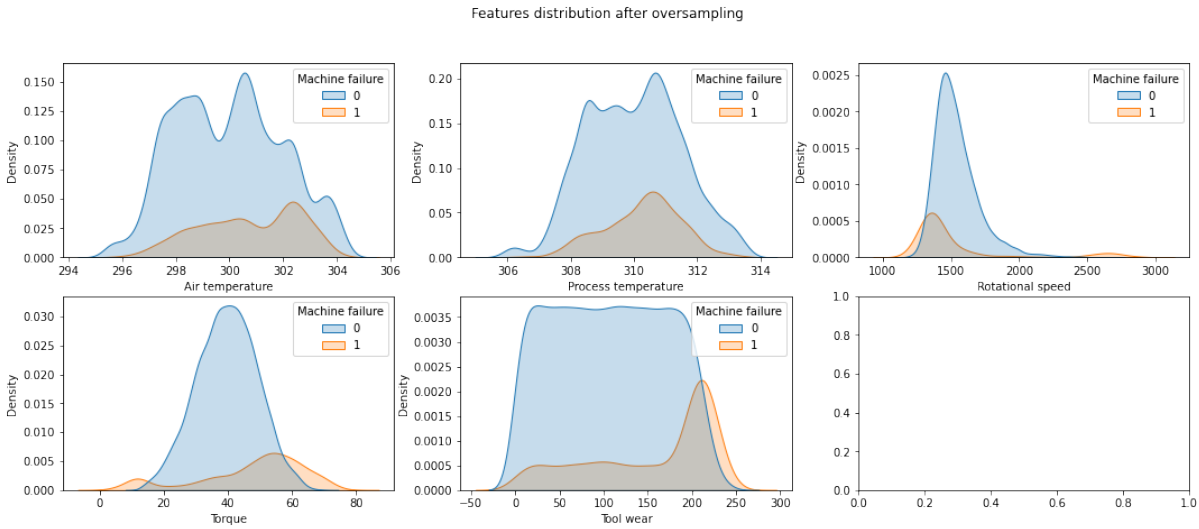


Figure 15: Oversampling features distribution

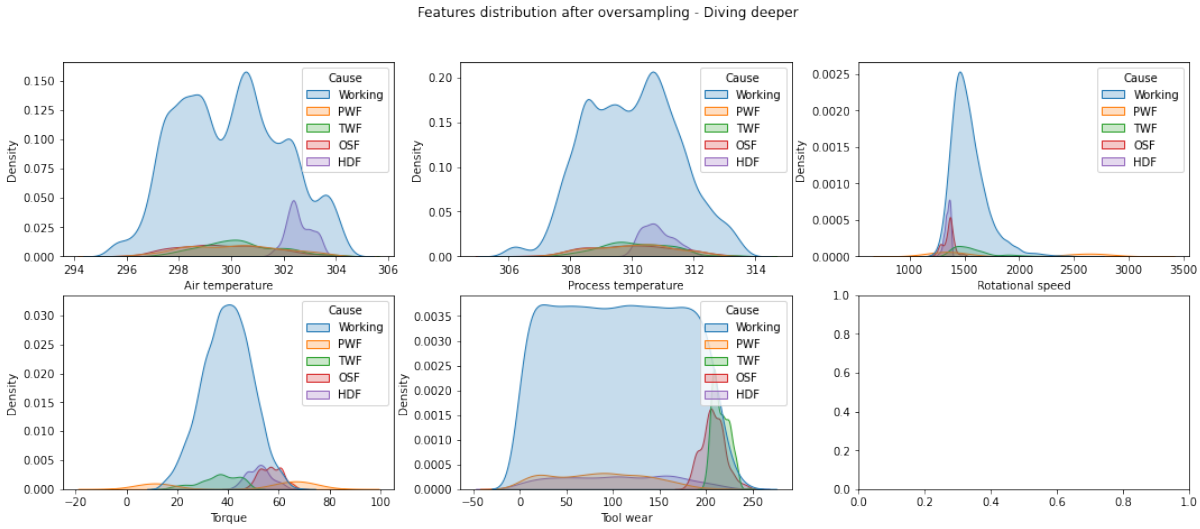


Figure 16: Oversampling features distribution- Diving deeper

The first thing we can observe is that the data augmentation was performed succesfully, as the feature distribution for faulty instancies have not been significantly distorted. It should also be noted that in Rotational Speed, Torque and Tool Wear the observations relating to failures have a density peak in extreme zones of the distribution. This implies that the outliers we discussed in Section 2.3 are not to be imputed to mistakes in the dataset building but rather to the natural variance of the same. This becomes even clearer when observing the distributions relative to the single causes of failure: in particular, an almost symmetrical behavior is recognized in Rotational Speed and Torque while in Tool Wear a clear separation is observed between PWF and HDF failures on lower values, and the peaks that are found at higher values relative to TWF and OSF.

This is perfectly consistent with the description of the targets reported in the "Task and dataset description" section.

2.5 Features scaling and Encoding

In order to make data exploitable for the algorithms we will run, we apply two transformations:

- First, we apply a label encoding to the categorical columns, since Type is an ordinal feature and Cause must be represented in one column. The mapping follows this scheme:
  Type: {L=0, M=1, H=2}
  Cause: {Working=0, PWF=1, OSF=2, HDF=3, TWF=4}

- Secondly we perform the scaling of the columns with StandardScaler. This is particularly useful for the good working of methods that rely on the metric space, such as PCA and KNN. It has been also verified that using StandardScaler leads to slightly better performances than using MinMaxScaler.

2.6 PCA and Correlation Heatmap

We run PCA to have a further way of displaying the data instead of making feature selection.

```
Explained variance ratio per component:
PC1    37.41
PC2    37.02
PC3    19.94
PC4     3.04
PC5     2.59
dtype: float64
Explained variance ratio with 3 components: 94.37
```

Figure 17: Explained Variance of PCA

Since the first three components are enough to almost fully represent the variance of the data we will project them in a three dimensional space.

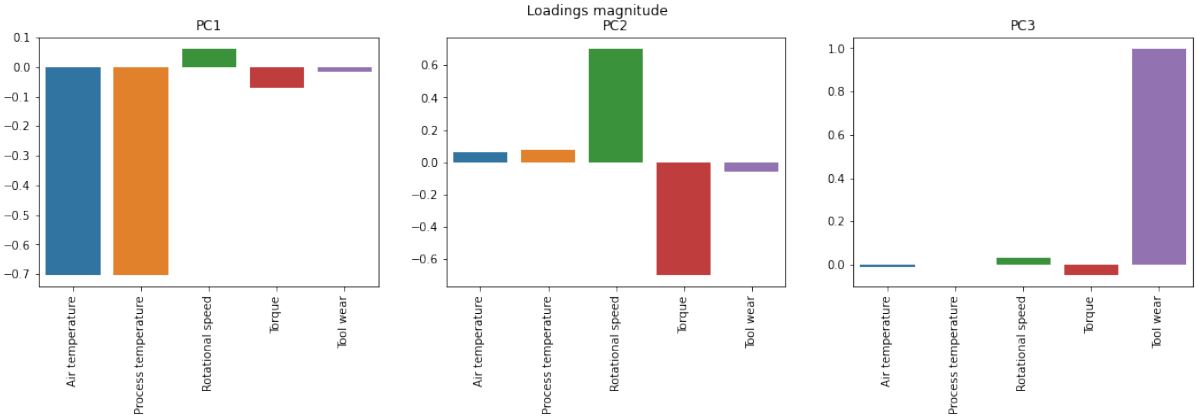


Figure 18: Principal Components Weights

The bar plot of Principal Components weights makes easy to understand what they represent:

- PC1 is closely related to the two temperature data;
- PC2 can be identified with the machine power, which is the product of Rotational Speed and Torque;
- PC3 is identifiable with Tool Wear.

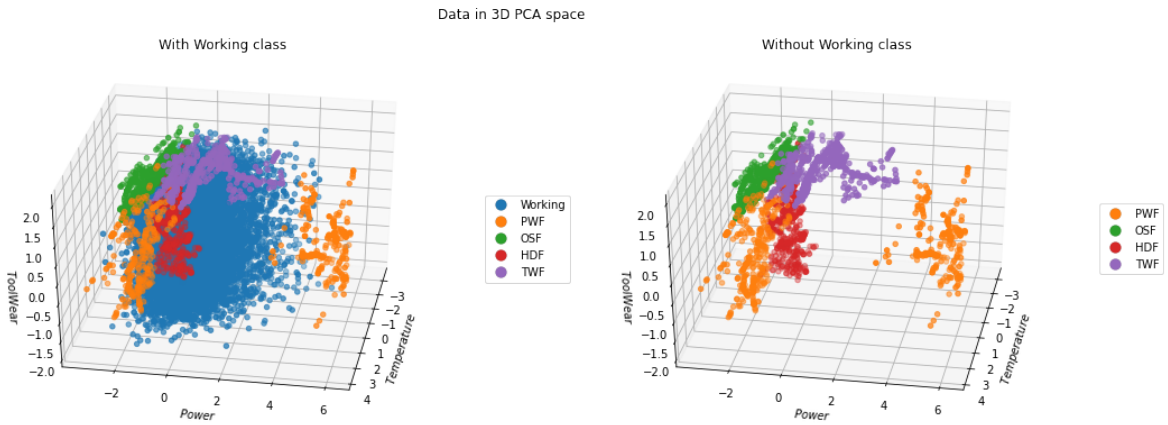


Figure 19: 3D PCA

The projection into the space generated by these three axes highlights that:

- TWF is the class of failures best separated from all the others and seems to depend almost entirely on PC3 (Tool Wear);
- PWF occupies two extreme bands along the PC2 (Power), it is independent of the other two components;

- The OSF and HDF classes are less separated than the others even if it can be observed that the first is characterized by a high Tool Wear and low power while the second is characterized by a high temperature and a low power.

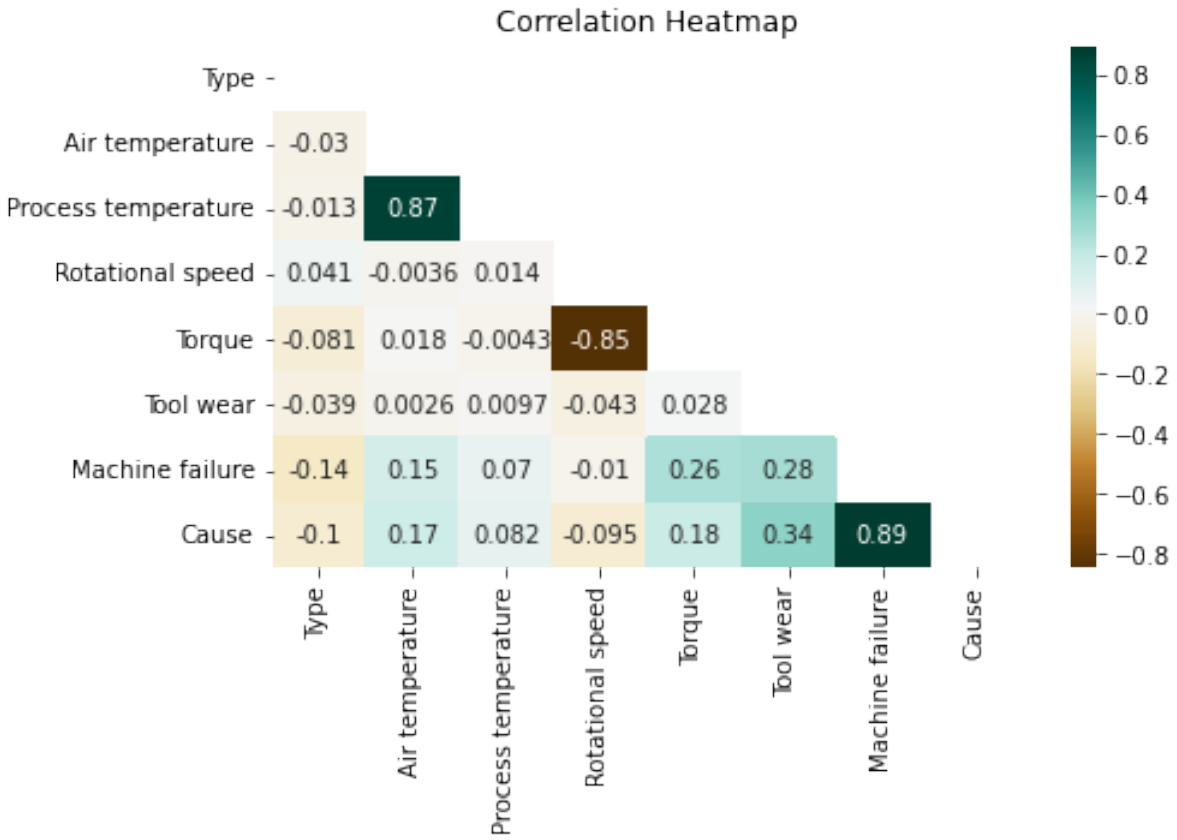


Figure 20: Correlation Heatmap

Unsurprisingly, we observe that the features related to temperature, as well as those related to power, are widely correlated. Furthermore, Tool Wear correlates well with both of our targets, confirming what we have observed by studying PCA. Finally, a less strong correlation is also observed between the torsion and the two targets.

2.7 Metrics

To evaluate the models we will use from a quantitative point of view, we resort to some metrics that summarize some characteristics of the classification results:

- Accuracy: expresses the fraction of instances that are classified correctly, it is the most intuitive metric that is usually used in classification tasks.

$$Accuracy = \frac{TP + TN}{TP + TN + FT + FN}$$

- AUC: can be considered as a measure of the separation between True Positives and True Negatives, that is, the ability of the model to distinguish between classes. In detail, it represents the area below the ROC curve, given by the estimate of the True Positive Rate (Recall) for each possible value of the True Negative Rate).

- F1: reports the classification capacity of the model to Precision and Recall, giving both the same weight.

$$F1 = 2\frac{Precision * Recall}{Precision + Recall}$$

Although generally effective, the AUC can be optimistic in the case of highly unbalanced classes, as happens in the binary task, while the F1 score is more reliable in this kind of scenario. We consider this last metric particularly significant as it is able to mediate the cases in which the machines that are about to fail are classified as functioning (Recall) and the one in which functioning machines are classified as about to suffer a failure (Precision). To be more specific we will give more importance to Recall than Precision, by evaluating also an "adjusted" version of the F1 through a β parameter:

$$F_\beta = (1 + \beta^2)\frac{Precision * Recall}{\beta^2 * Precision + Recall}$$

With the choice $\beta = 2$ (common in literature) a greater influence of the Recall is obtained. This choice is motivated by the fact that in order to optimize the costs for the maintenance of the machinery it is a good thing to limit the purchase of unnecessary replacement materials but it results far more important to avoid the possibility of having to replace a machinery after it is broken, since this second scenario generally has higher costs.

3 Binary classification

The goal of this section is to find the best model for binary classification of the dataset to predict whether or not there will be Machine Failure. Classification algorithms are part of data mining and use supervised machine learning methods to make predictions about data. In particular, a set of data already divided ("labeled") into two or more classes of belonging is provided as input thanks to which a classification model is created, which will than be used on new ("unlabeled") data to assign them to the appropriate class. The starting dataset is usually divided into three groups: the training dataset, i.e. the sample of data used to fit the model, the validation dataset, i.e. the sample of data used to provide an evaluation of a model fit on the training dataset while tuning model hyperparameters and the test dataset, which has the purpose of testing the model. At the beginning of a project a data scientist must make this division and the common ratios used are:

- 70% train, 15% val, 15% test.
- 80% train, 10% val, 10% test.
- 60% train, 20% val, 20% test.

In this project we use the ratio (80/10/10) for the split because we test the model for all of these strategies and find that it is the best one.
The classification techniques we choose to implement are the following:

- **Logistic Regression:** it estimates the probability of a dependent variable as a function of independent variables. The dependent variable is the output that we are trying to predict while the independent variables or explanatory variables are the factors that we feel could influence the output. For its simplicity and interpretability, we decide to use Logistic Regression as a Benchmark model, a basic model that represents the starting point for comparing the results obtained from other models.
- **K-nearest neighbors (K-NN):** algorithm based on the calculation of the distance between the elements of the dataset. Data is assigned to a certain class if close enough to the other data of the same class. Parameter K represents the number of neighboring data taken into account when assigning classes.
- **Support Vector Machine:** its aim is to find a hyperplane in an N-dimensional space (N

—the number of features) that distinctly classifies the data points while maximizing the margin distance, i.e. the distance between data points of both classes.

- **Random Forest:** it uses ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems. Random Forest uses bagging technique: it constructs a multitude of decision trees in parallel, all with the same importance, and the output is the class selected by most trees.

- **XGBoost:** is a gradient-boosted decision tree (GBDT) machine learning library. A Gradient Boosting Decision Tree (GBDT) is a decision tree ensemble learning algorithm similar to Random Forest, from which differs because it uses a boosting technique: it iteratively trains an ensemble of shallow decision trees, with each iteration using the error residuals of the previous model to fit the next model. The final prediction is a weighted sum of all of the tree predictions.

3.1 Feature selection attempts

Before going into the training of the models just mentioned we try to perform feature selection, exploiting the considerations we made about the correlation heatmap and the exploratory data analysis: just to remind, we noticed that the features "Process temperature" and "Air temperature" are positively correlated, and "Torque" and "Rotational speed" are negatively correlated. From the dateset description we see that the PWF failure occurs if the product between "Torque" and "Rotational speed" is in a certain range of values and, similarly, HDF failure occurs when the difference between "Air temperature" and "Process temperature" exceeds a certain value. For these reasons, completely deleting these columns seems to be a bad choice because important information can be lost but at the same time it is reasonable to see what happens if we combine them, taken by pairs, to create new features that still preserve a physical meaning. Therefore we proceed to compare the results obtained by fitting the classification models without tuning any parameter on the following datasets:

- the original one;

- the one obtained by removing the "Process temperature" and "Air temperature" columns, replacing them with a column of their product;

- the one obtained by removing "Torque" and "Rotational speed", replacing them with a

column of their product;

- a combine the previous operations.

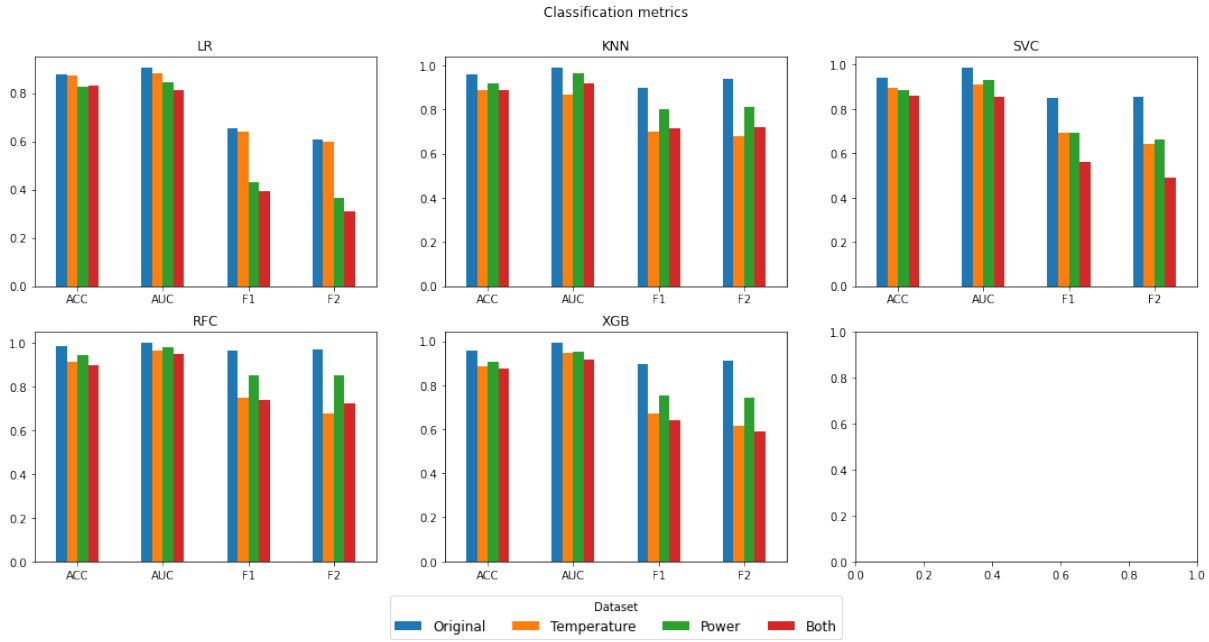


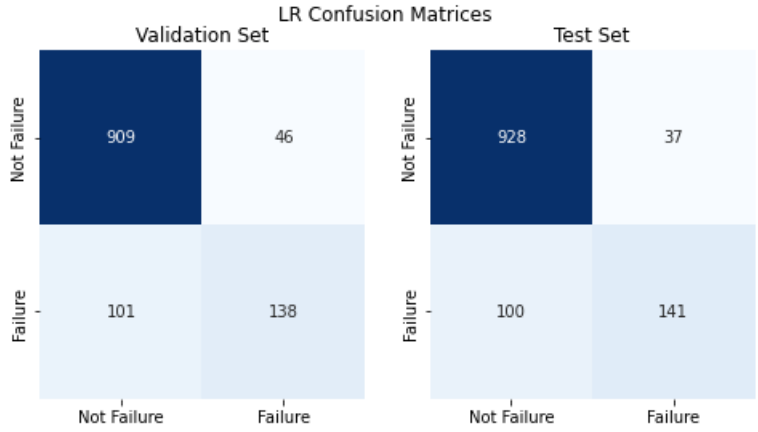
Figure 21: Comparison of models with different datasets

From the results obtained, we observe that all the models applied to the entire dataset perform better than when they are applied to the ones created by reducing the number of features. The best performances and the modest number of features from which our dataset is composed encourage us to opt to avoid the feature selection step.

3.2 Logistic Regression Benchmark

As previously stated, we decide to use Logistic Regression as a Benchmark for our task. It represents an intermediate step between the basic model referred to in *Section 2.4* and the more complex models that we have described and we will explore in depth in the following sections. Now we look at the results obtained and at the interpretability of the model.

```
Validation set metrics:
ACC     0.877
AUC     0.904
F1      0.652
F2      0.605
dtype: float64
Test set metrics:
ACC     0.886
AUC     0.906
F1      0.673
F2      0.617
dtype: float64
```



	feature	odds
4	Torque	16.865848
3	Rotational speed	10.977195
1	Air temperature	4.328909
5	Tool wear	3.488565
0	Type	0.595855
2	Process temperature	0.368579

Figure 22: Logistic Regression

The odds of logistic regression allow us to understand how the model is working. In particular, an unrealistically high importance is given to Torque and Rotational Speed. This is mainly due to the natural variance in these features, which is especially high when looking only at the failure cases and tends to "deviate" the model. However it is reasonable to believe, on the basis of exploratory analysis, that the first four features have a significantly greater relevance than the last two. We also expect greater reliability of the odds values when we apply logistic regression to the multiclass task, since the effects that are spread here appears to be localized around certain types of failures.

3.3 Models

```
GridSearch start
Training KNN
Best params: {'n_neighbors': 1}
Training time: 0m 3s
Training SVC
Best params: {'C': 100, 'gamma': 1, 'kernel': 'rbf', 'probability': True, 'random_state': 0}
Training time: 2m 44s
Training RFC
Best params: {'max_depth': 10, 'n_estimators': 100, 'random_state': 0}
Training time: 3m 10s
Training XGB
Best params: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 700, 'objective': 'binary:logistic'}
Training time: 3m 6s
```

Figure 23: Models

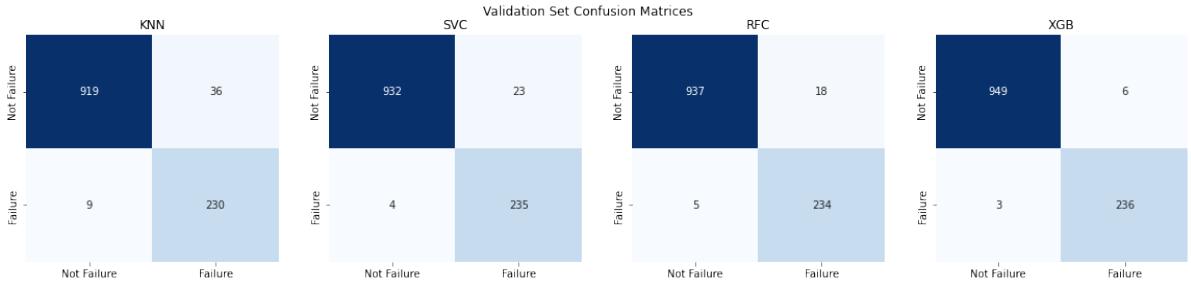


Figure 24: Validation set Confusion Matrices

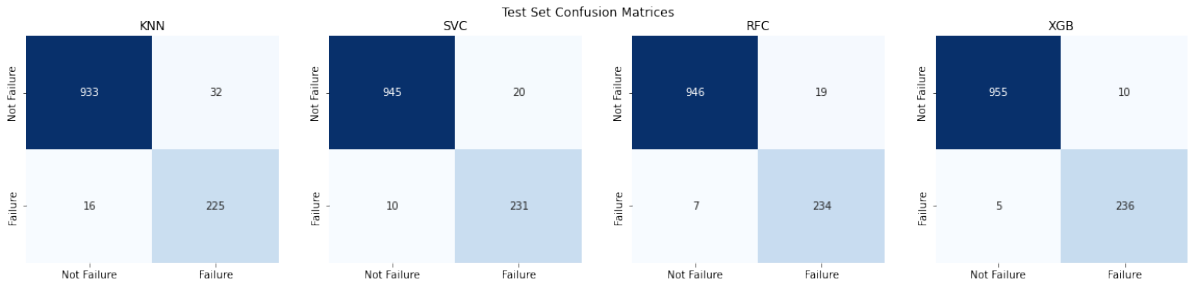


Figure 25: Test set Confusion Matrices

```
Validation scores:
       KNN    SVC    RFC    XGB
ACC  0.962  0.977  0.981  0.992
AUC  0.962  0.994  0.998  1.000
F1   0.911  0.946  0.953  0.981
F2   0.941  0.968  0.969  0.985
Test scores:
       KNN    SVC    RFC    XGB
ACC  0.960  0.975  0.978  0.988
AUC  0.950  0.990  0.998  0.999
F1   0.904  0.939  0.947  0.969
F2   0.921  0.951  0.961  0.975
```

Figure 26: Validation scores

All the selected models obtain similar results on the validation set (except KNN which is a little worse) and it is diffj7icult to determine if one works better than another by looking only at these values. Performance did not significantly drop when passing the test set, showing that overfitting was avoided. We comment on the results of the models by looking at the confusion matrices and the metrics obtained on the test set: in this way the formation of a hierarchy between the models used is slightly clearer, as all the metrics relating to a single model are smaller or larger than to the others and the time needed to search for the parameters is comparable, with the only exception of KNN. In particular KNN obtains the worst performances and XGB the best ones; in the middle we find SVC and RFC which achieve extremely similar results. About the parameters:

- A Gridsearch has been started on the parameters which, looking in the literature, appear to be preponderant for each specific model;
- The grid values to search for have been defined on the basis of literature and various tests, trying to keep the computational cost of finding the best values moderate.

It is interesting to observe that the optimal parameters for RFC and XGB are the polar opposite: the former prefers to use a few estimators and go into depth while the latter uses more estimators with fewer splits. Furthermore, it must be taken into account that although XGB is the best classifier from a quantitative point of view, this is not true for what concerns the qualitative side. Both SVC and XGB in fact lack clear ways to interpret the results, while on the contrary RFC allows to have a complete understanding of how the algorithm worked. In any case, to get an idea of which features had greater importance in making the predictions, we report the permutation feature importances in a bar plot.

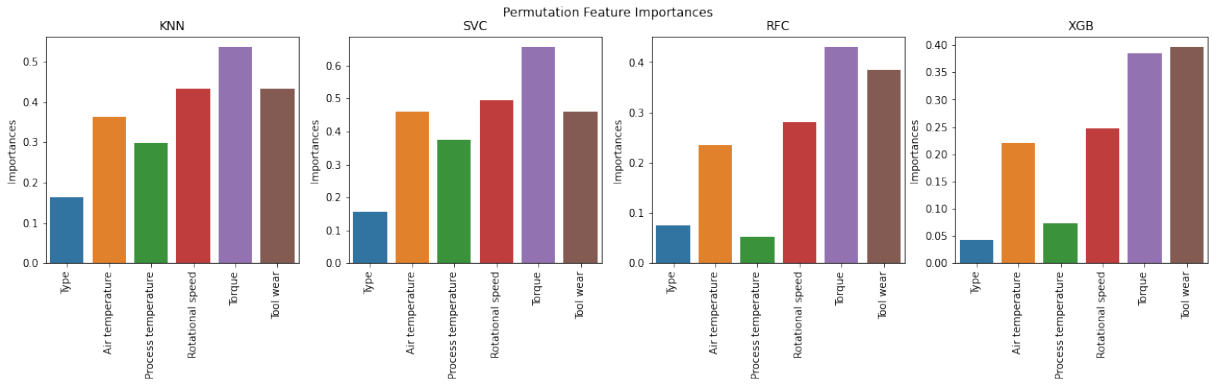


Figure 27: Permutation Feature Importances

Remarks on Feature importances:

- Type is the feature with the lowest significance, in accordance with what was observed during the exploratory analysis. However, its importance remains strictly positive in each of the cases considered and therefore removing it completely would have led to a decline in prediction performance, not justified by a significant computational gain;
- Unlike Logistic Regression, the models tested place great emphasis on Tool wear as well as Torque and Rotational Speed. Since the former alone is related to a specific category of failures and strongly distorts the kdeplot of Machine failure, we have a sign that our models still worked well.

4 Multiclass classification

We now proceed to the second task of this project, that is predict not only if there will be a failure, but also the type of failure that will occur. So we are in the case of multiclass classification problems that make the assumption that each sample is assigned to one and only one label. This hypothesis is verified because in data preprocessing we removed all the ambiguous observations that belonged to more than one class.

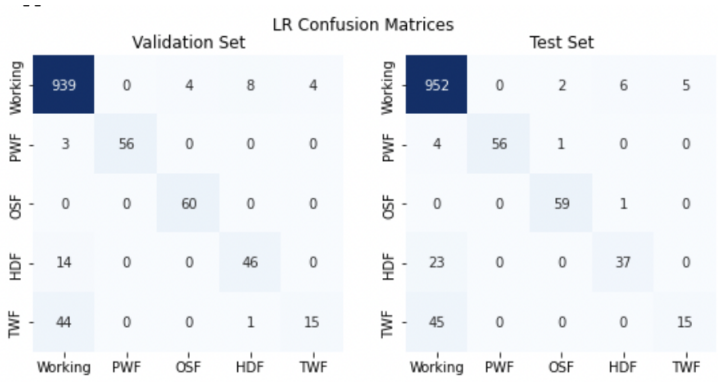
For multiclass targets, when we calculate the values of AUC, F1 and F2 scores, we need to set the parameter "average". We choose "average=weighted", in order to account for class imbalance: in fact, at the end of data preprocessing, we have 80% WORKING machine and 20% that fail.

As for binary classification task, we choose Logistic Regression as baseline model and we look for models that get higher values for the chosen metrics. In particular, we adapt to the multiclass case the models developed in the previous section. While many classification algorithms (such as K-nearest neighbor, Random Forest and XGBoost) naturally permit the use of more than two classes, some (like Logistic Regression and Support Vector Machines) are by nature binary algorithms; these can, however, be turned into multiclass classifiers by a variety of strategies. For our project, we decide to use "OnevsRest" approach, who involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. We choose it because it is computationally more efficient than other types of approach.

So, first let's look at how the Logistic Regression behaves:

```
Validation set metrics:
ACC    0.935
AUC    0.982
F1     0.924
F2     0.930
dtype: float64
Test set metrics:
ACC    0.928
AUC    0.984
F1     0.917
F2     0.922
dtype: float64
```

Figure 28: Logistic Regression metrics



	Type	Air temperature	Process temperature	Rotational speed	Torque	Tool wear
Working	1.678260	0.231005	2.713123	0.091098	0.059291	0.286651
PWF	0.735111	0.797016	1.053163	1045.288415	1904.941172	0.646302
TWF	0.034002	0.441272	1.732487	0.605616	304.765763	17345.864729
OSF	0.707729	6144.344664	0.003320	0.000450	0.552662	0.764262
HDF	1.263984	1.395851	0.743510	0.186778	0.087875	768.297317

Figure 29: Logistic Regression

In the table in the *Figure 29* there are, for every class, the Logistic Regression's odds that explain the contribution of each feature in the prediction of belonging to a specific class. By comparing this table with the PCA scatter and the comments we made, we understand that there is a complete agreement about the features that most affect the type of failure. For example, if we look at odds' values of PWF, we see that Rotational Speed and Torque are the ones that are most important for the forecast of belonging to this class. In the analysis of the PCA we stated that PWF seems to be dependent only on PC2, i.e. the Power that is the product of Rotational Speed and Torque. We can make similar considerations for other classes.

4.1 Models

For each model we launch the Gridsearch for hyperparameter optimization, using as metric to evaluate the model the weighted average F2 score. Similarly to the binary case, the Gridsearch has been started on the parameters that, looking in the literature, are found to be preponderant for each specific model and the grid values to look for have been defined according to the literature and several tests carried out.

```
GridSearch start
Training KNN
Best params: {'n_neighbors': 1}
Training time: 0m 3s
Training SVC
Best params: {'C': 100, 'gamma': 1, 'kernel': 'rbf', 'probability': True, 'random_state': 0}
Training time: 2m 25s
Training RFC
Best params: {'max_depth': 10, 'n_estimators': 700, 'random_state': 0}
Training time: 2m 49s
Training XGB
Best params: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 300, 'objective': 'multi:softprob'}
Training time: 9m 43s
```

Figure 30: Models

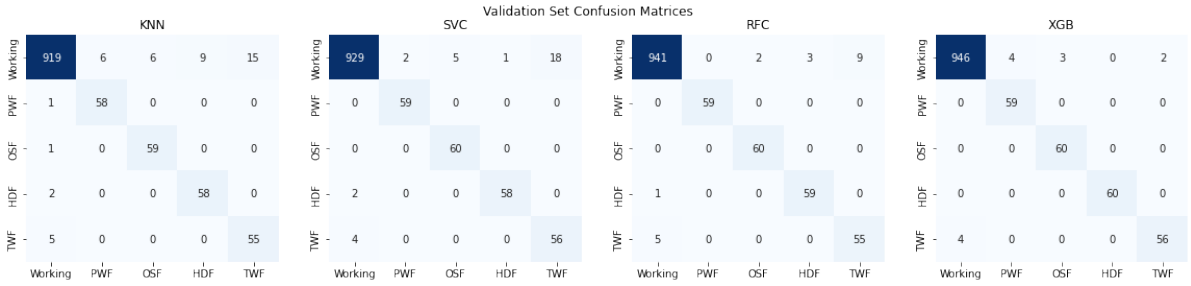


Figure 31: Validation set Confusion Matrices

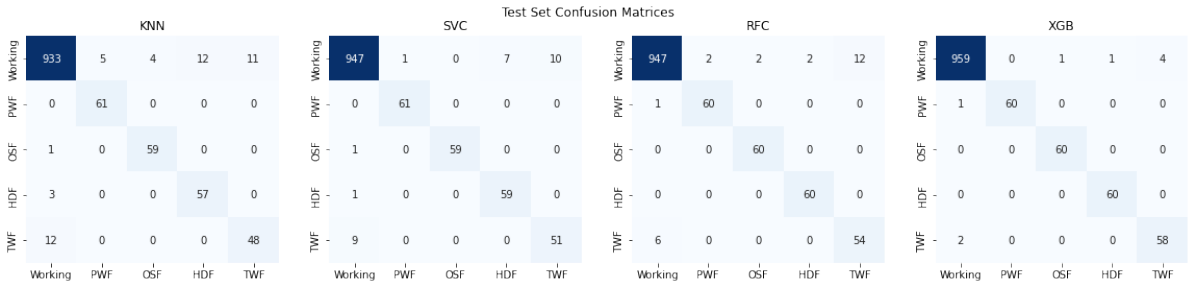


Figure 32: Test set Confusion Matrices

```
Validation scores:
        KNN     SVC     RFC     XGB
ACC   0.962   0.973   0.983   0.989
AUC   0.965   0.997   0.998   0.999
F1    0.963   0.974   0.983   0.989
F2    0.962   0.973   0.983   0.989
Test scores:
        KNN     SVC     RFC     XGB
ACC   0.960   0.976   0.979   0.993
AUC   0.953   0.995   0.998   0.999
F1    0.961   0.976   0.980   0.993
F2    0.960   0.976   0.979   0.993
```

Figure 33: Validation scores

By comparing the results obtained, we see that K-NN is the model that performs the worst and its accuracy is a little lower than Logistic Regression's one. Despite this, we cannot exclude it a

priori, as it still reaches high values for the metrics and, moreover, gives an immediate response. So, we can use it whenever we need to get an idea quickly about the situation and, then apply other models when we have more time.

All other models perform better than the benchmark and they obtain high values for the chosen metrics both for validation and test set.

SVC and RFC's performances are very similar each other and XGB performs better than them. If we look at the training phase, SVC and RFC take the same time, while XGB takes more than four times as much as them. So, since, the improvement obtained with XGB is only 1.5%, one can choose which model he prefers according to his needs.

While the best parameters for multiclass K-NN and SVC are the same as binary classification, for XGB and RFC the Gridsearch for the two types of task returns different parameters. Moreover, in the transition from binary to multiclass problem, the estimated training time remains the same for all models, except for XGB that triples it.

In order to understand how features contribute to predictions, let's look at the Permutation Feature Importances for each model.

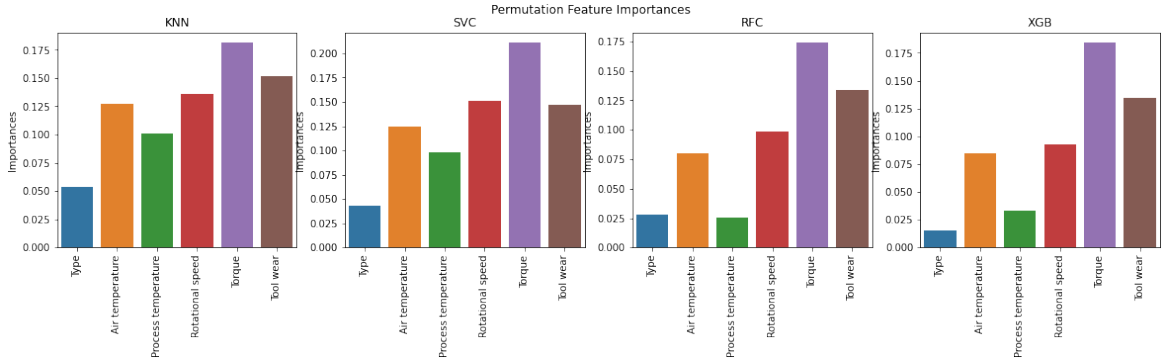


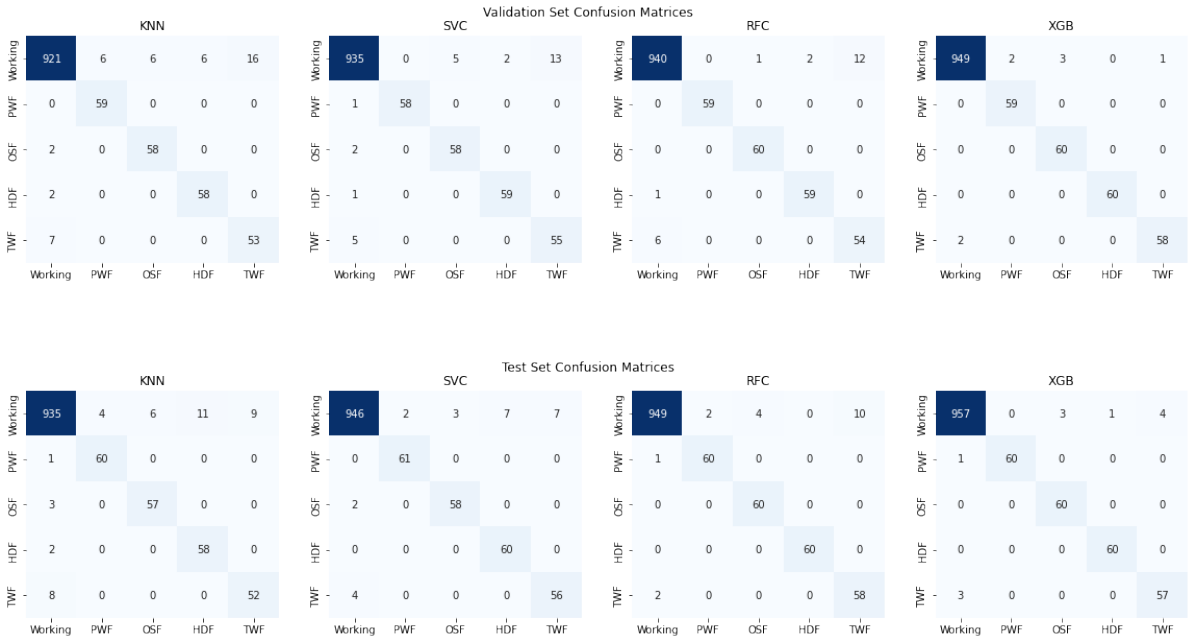
Figure 34: Permutation Feature Importances

From previous barplots we see that the models give more importance to Torque, Tool wear and Rotational Speed while the Type contribution is very low. This is in accordance with the observations made in the exploration of the dataset in *Section 1-2* and it is consistent with the Permutation Feature Importances of binary task.

K-NN is the one who gives more importance to Type, but, different from binary case, here we see that for every model the Type contribution is almost zero. So, we test the model on a new dataset, the old one from which we removed the column Type. For K-NN and SVC there is an insignificant improvement in the metrics' values, which were already very good. For RFC and

XGB we do not see any change on metrics' values. Since the training time for the different models is approximately equal in both cases, we let users choose which dataset to use.

```
GridSearch start
Training KNN
Best params: {'n_neighbors': 1}
Training time: 0m 4s
Training SVC
Best params: {'C': 100, 'gamma': 1, 'kernel': 'rbf', 'probability': True, 'random_state': 0}
Training time: 1m 54s
Training RFC
Best params: {'max_depth': 10, 'n_estimators': 500, 'random_state': 0}
Training time: 3m 16s
Training XGB
Best params: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 300, 'objective': 'multi:softprob'}
Training time: 9m 59s
```



Validation scores:

	KNN	SVC	RFC	XGB
ACC	0.962	0.976	0.982	0.993
AUC	0.962	0.996	0.998	1.000
F1	0.963	0.976	0.982	0.993
F2	0.962	0.976	0.982	0.993

Test scores:

	KNN	SVC	RFC	XGB
ACC	0.964	0.979	0.984	0.990
AUC	0.958	0.994	0.998	0.999
F1	0.964	0.980	0.985	0.990
F2	0.964	0.979	0.984	0.990

Figure 35: Results on dataset without Type

5 Decision paths

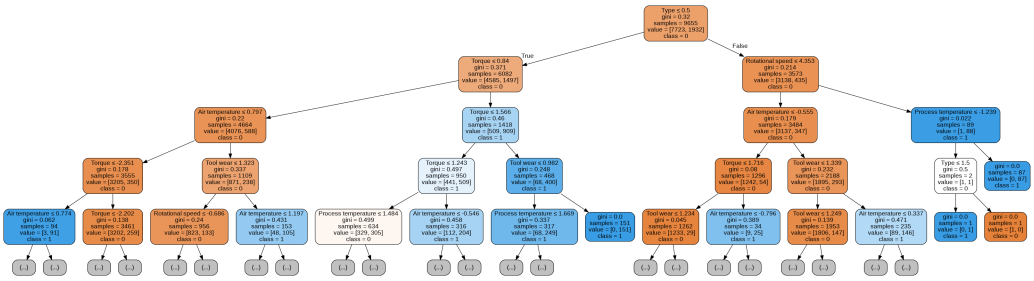


Figure 36: One Decision Tree from binary forest

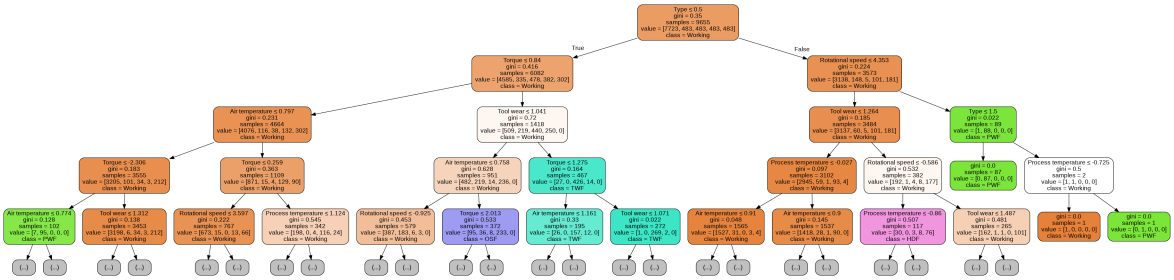


Figure 37: One Decision Tree from multi-class forest

Here we show the decision paths of one of the trees that make up the Random Forest for both tasks, truncated at depth=4. However this depth is enough to verify that trees require to be deep because the decision boundary are complex themselves and they are not overfitting. This is evident if one looks at the multi-class tree, where some kinds of failure do not appear before depth four, but also in the binary classification tree by looking at the evolution of the gini score while following most of the paths. A further remark can be made about the feature Type being the origin node of both graphs and separating the majority class (Low quality) from the other two at the first step. It appears just one time more in the upper side of the trees and shows sporadically again at the lowest floors, where its impact is scarce.

6 Conclusion

According to the analyses carried out and the results obtained, it is possible to make some conclusive considerations related to this project.

We decided to tackle two tasks: predict whether a machine will fail or not and predict the type of failure that will occur. Before developing the models we did data preprocessing to ensure the validity of the assumptions of applicability of the models and ensure the best performances.

Briefly, in preprocessing phase we have deleted some ambiguous samples, we applied a label encoding to the categorical columns and then we performed the scaling of the columns with StandardScaler. We also noticed the presence of some data points which at first we referred as outliers but later turned out to be part of the natural variance of the data and played an important role in the classification task. Then we ran PCA and found that most of the variance is explained by the first three components, that can be represented as the following features: combination of the two Temperatures, Machine Power (product of Rotational Speed and Torque) and Tool Wear. In according to this, we found that these are the features that contribute the most in the predictions when apply the models. Contrary to logical predictions, we demonstrated that the machine's type does not affect the presence of failure.

At the end, we can conclude that for both task the chosen models perform very well. For both tasks the best model is XGBoost and the worst is KNN; however the response time of KNN is instant while XGBoost takes more time and this further increase when we proceed with the multi-class classification task. The choice of the model depends on the needs of the company: for faster application one can use KNN while if one cares more about accuracy one can use XGBoost.

7 Bibliography

[1] *Cohen J.* - Machine Learning: Target Feature Label Imbalance Problems and Solutions - https://towardsdatascience.com

[2] *De Harder H.* - Interpretable Machine Learning Models - https://towardsdatascience.com

[3] *Hervé Abdi1, Lynne J. Williams* - Principal component analysis

[4] *Joos Korstanje* - SMOTE - https://towardsdatascience.com

[5] *Matzka S.* - Explainable Artificial Intelligence for Predictive Maintenance Applications' - 3th International Conference on Artificial Intelligence for Industries AI4I 2020

[6] *Matzka S., Torcianti A.* -Explainable Artificial Intelligence for Predictive Maintenance Applications using a Local Surrogate Model - 4th International Conference on Artificial Intelligence for Industries AI4I 2021

[7] *Narkhede S.* - Understanding AUC - ROC Curve - https://medium.com

[8] *Piccialli F.* - Algorithms and applications for Data Science slide course 2022

[9] *Ringnér Markus* - What is principal component analysis?

[10] *Sam* - F1 Score - https://medium.com

[11] *Shalev-Shwartz S., Ben-David S.* - Understanding Machine Learning: From Theory to Algorithms - Cambridge University Press (2014)

[12] *Swalin A.* - Choosing the Right Metric for Evaluating Machine Learning Models — Part 2 - https://medium.com

[13] *Tarang Shah* - About Train, Validation and Test Sets in Machine Learning - https://towardsdatascience.com