PROJECT REPORT

ON

Predictive Maintenance

A project I Submitted in partial fulfillment of the requirements

for the degree of

B. Tech in CSE (Artificial Intelligence and Machine Learning)

BY

Aashirwad Wardhan

Ayush Taparia

Aman Singh

Pritam Chowdhury

Oishik Mukhopadhyay

UNDER THE GUIDANCE OF

Dr. Ritamshirsa Choudhuri

Designation

CSE (Artificial Intelligence and Machine Learning) Department

TECHNO MAIN SALT LAKE

EM 4/1 SALT LAKE CITY, SECTOR V

KOLKATA – 700091

West Bengal, India

Techno Main Salt Lake

[Affiliated by Maulana Abul Kalam Azad University of Technology (Formerly known as WBUT)]

FACULTY OF CSE (AI & ML) DEPARTMENT

Certificate of Recommendation

This is to certify that they have completed their project report on **Predictive Maintenance**, under the direct supervision and guidance of **Dr. Ritamshirsa Choudhuri**. We are satisfied with their work, which is being presented for the partial fulfillment of the degree of B. Tech in CSE (Artificial Intelligence and Machine Learning), Maulana Abul Kalam Azad University of Technology) (Formerly known as WBUT), Kolkata – 700064.

Dr. Ritamshirsa Choudhuri
(Signature of Project Supervisors)
Date:

Dr. Sudipta Chakrabarty
(Signature of Project Coordinator)
Date:

Dr. Soumik Das
(Signature of the HOD)
Date:

Acknowledgement

We extend our sincere thanks to Dr. Ritamshirsa Choudhuri, our respected professor, for his invaluable guidance and continuous support throughout our project on Predictive Maintenance. His mentorship played a pivotal role in shaping our understanding of the subject matter, enabling us to navigate the complexities of the research.

We are grateful to Dr. Soumik Das, Head of the Department, and Dr. Sudipta Chakrabarty, our project supervisor, for providing us with the opportunity to undertake this project. Their trust and encouragement fueled our exploration into the realm of stock prediction and analysis. Their guidance was instrumental in our learning journey, allowing us to delve into the project's nuances.

In the face of challenges encountered during the project, Dr. Ritamshirsa Choudhuri's unwavering support helped us overcome obstacles and shape our ideas into a cohesive project. We express our gratitude to all those who contributed, directly or indirectly, to the successful completion of our research. This project has been a significant learning experience, and we appreciate the collaborative efforts that made it possible.

Date: 29

--------------------------	--------------------------	--------------------------
Aashirwad Wardhan	Ayush Taparia	Pritam Chowdhury
13030820057	13030820007	13030820021

--------------------------	--------------------------
Oishik Mukhopadhyay	Aman Singh
13030820035	13030821066

Abstract

Predictive maintenance is an advanced maintenance strategy that leverages data analytics, machine learning, and real-time monitoring to predict equipment failures before they occur. This proactive approach aims to optimize maintenance schedules, reduce downtime, and enhance the longevity of machinery. By analyzing data from sensors and historical maintenance records, predictive models can identify patterns and anomalies indicative of impending failures. The implementation of PdM involves integrating Internet of Things (IoT) devices, deploying sophisticated analytics platforms, and continuously updating predictive algorithms with new data. The benefits of PdM are multifold: it minimizes unplanned maintenance, lowers operational costs, and improves overall equipment effectiveness (OEE). Additionally, it enables more efficient resource allocation and extends asset life, contributing to sustainability goals. However, challenges such as high initial setup costs, data integration complexities, and the need for specialized expertise must be addressed to fully realize the potential of PdM. This report delves into the methodologies, applications, and impacts of predictive maintenance, offering insights into its role in transforming industrial maintenance practices.

Table of Contents

Introduction

In the evolving landscape of Industry 4.0, predictive maintenance has emerged as a crucial component for enhancing operational efficiency and reducing costs. At the heart of this technological revolution is the ability to anticipate machine failures and understand the nature of these failures. This predictive capability is essential because repairing or replacing an entire faulty machine can incur significantly higher costs compared to the preemptive replacement of a single malfunctioning component. By leveraging advanced data science techniques and installing sensors that continuously monitor machine conditions, industries can gather critical information that drives substantial cost savings and operational improvements.

This work focuses on utilizing the AI4I Predictive Maintenance Dataset from the UCI Repository to conduct a comprehensive analysis aimed at addressing the predictive maintenance needs of modern industries. The dataset provides a robust foundation for exploring and understanding machine behavior, failure patterns, and maintenance requirements. Initially, we delve into an exploratory data analysis (EDA) to gain a deep understanding of the dataset's structure, underlying patterns, and key features. This step is crucial for uncovering insights that inform the subsequent stages of the analysis, ensuring that our approach is data-driven and tailored to the specifics of the dataset.

Following the exploratory phase, we implement a series of preprocessing techniques designed to prepare the data for machine learning algorithms. This involves cleaning the data, handling missing values, normalizing features, and engineering new variables that can enhance predictive power. With the dataset refined and ready for analysis, we embark on two primary tasks: first, predicting whether a given machine is likely to experience a failure, and second, identifying the specific type of failure that might occur. These tasks are tackled using a variety of machine learning models, each selected for its potential to provide accurate and reliable predictions.

Finally, we compare the performance of these models using appropriate metrics such as accuracy, precision, recall, and F1 score. Additionally, we emphasize the interpretability of the models, which is vital for practical implementation in industrial settings. Understanding why a model makes certain predictions can help maintenance teams make informed decisions and take proactive measures. This work not only demonstrates the application of machine learning in predictive maintenance but also highlights the benefits of data-driven decision-making in reducing downtime and maintenance costs, ultimately contributing to the efficiency and competitiveness of Industry 4.0 enterprises

Scope of the Project

Predictive maintenance has emerged as a transformative approach to asset management, offering organizations a proactive alternative to traditional reactive and time-based preventive maintenance strategies. At the core of predictive maintenance is the integration of data-driven techniques, such as machine learning and advanced analytics, to continuously monitor the condition and performance of equipment and machinery. By leveraging sensor data, historical maintenance records, and other relevant information, predictive maintenance systems can generate accurate forecasts of when equipment is likely to fail or require servicing. This allows maintenance teams to intervene preemptively, addressing issues before they escalate into costly, unplanned downtime.

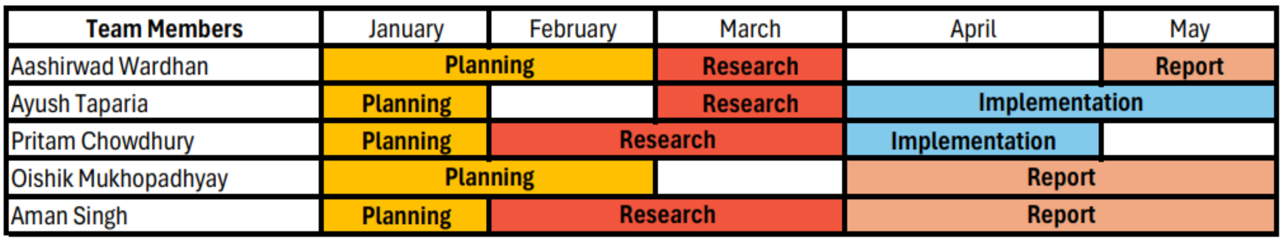
The benefits of adopting a predictive maintenance approach are significant and far-reaching. By avoiding reactive firefighting and unnecessary preventive maintenance, organizations can optimize their maintenance schedules, extend the useful lifespan of their assets, and improve overall equipment effectiveness. Predictive maintenance also enables more targeted and efficient utilization of maintenance resources, as maintenance activities can be precisely scheduled based on the predicted condition of the equipment. This, in turn, leads to reduced maintenance costs, improved productivity, and enhanced operational resilience. Furthermore, the data-driven insights generated through predictive maintenance can inform strategic decision-making, helping organizations make more informed investments in their asset management infrastructure and align maintenance practices with broader organizational goals.

Overall, the transition from reactive or time-based maintenance to a predictive maintenance framework represents a significant leap forward in the field of asset management, leveraging the power of data and advanced analytics to drive greater reliability, efficiency, and cost savings across diverse industrial and commercial sectors

Concepts and Problem Analysis

Time Analysis:

Gantt Chart



Team Members	January	February	March	April	May
Aashirwad Wardhan	Planning		Research		Report
Ayush Taparia	Planning		Research	Implementation	
Pritam Chowdhury	Planning	Research		Implementation	
Oishik Mukhopadhyay	Planning			Report	
Aman Singh	Planning	Research		Report	

➔ **Planning:** In collaboration with our professor, Dr. Ritamshirsa, we explored various ideas and methodologies for predictive maintenance. We identified key indicators and algorithms tailored for predicting machine failures and their specific causes.

➔ **Research**: We conducted thorough research on predictive maintenance techniques, studying algorithms and tools used by industry experts. This involved understanding both the mathematical foundations and practical applications of these methods, allowing us to select the most suitable strategies for our analysis.

➔ **Implementation:** Using Python and its libraries, we performed data preprocessing, model training, and evaluation. We also developed a visualization tool with Streamlit to analyze machine health data, providing actionable insights for maintenance teams to predict failures and determine necessary maintenance actions.

Team Structure

Our team operates with a collaborative spirit, fostering open communication and shared decision-making. Regular team meetings serve as forums for idea exchange, progress updates, and issue resolution, ensuring a cohesive and dynamic workflow. Recognizing the dynamic nature of data science projects, we have established flexible timelines and contingency plans to adapt to unforeseen challenges. By leveraging the diverse skills of each team member and maintaining a strong communication framework, we are confident in our ability to not only meet but exceed the objectives outlined in our predictive maintenance project.

In summary, our team, consisting of Aashirwad Wardhan, Ayush Taparia, Pritam Chowdhury, Oishik Mukhopadhyay, and Aman Singh, collaborates seamlessly to tackle the intricacies of predictive maintenance. With Aashirwad, Oishik, and Aman leading research and planning, and Ayush and Pritam focusing on code implementation, we ensure a comprehensive approach to the project. Oishik and Aman will consolidate our findings into a detailed report, capturing the essence of our research journey. Through effective communication, adaptability, and a commitment to excellence, our team aspires to make meaningful contributions to the field of predictive maintenance technology.

Software Configuration Management:

Software Configuration Management (SCM) is a critical aspect of managing software projects effectively. It involves the systematic control of software artifacts, including source code, documentation, configurations, and dependencies. Here is a suggested approach to SCM for the predictive maintenance project:

1. Version Control: Utilize a version control system, such as Git, to manage source code and track changes. Maintain a central repository where team members can collaborate, share code, and track version history. Use branching and merging strategies to manage concurrent development and ensure code stability.

2. Dependency Management: Maintain a list of software libraries, packages, and dependencies used in the project. Utilize dependency management tools such as Maven, Pip, or npm to manage and resolve dependencies automatically. Document the versions and configurations of dependencies to ensure consistency across development and deployment environments.

3. Configuration Management: Maintain configuration files that capture the project's settings and parameters. Use configuration management tools such as Ansible or Chef to automate the provisioning and management of software configurations across different environments. This ensures consistent behavior and reduces configuration-related errors.

4. Build Automation: Automate the build process to ensure reproducibility and consistency. Use build automation tools like Jenkins or Travis CI to build, test, and package the software artifacts. Define build scripts or configuration files that specify the necessary steps to compile, test, and deploy the project.

5. Release Management: Establish a release management process to manage the deployment and distribution of software releases. Utilize release management tools to package and distribute the software artifacts, document release notes, and manage versioning. Clearly define release milestones and communicate changes and updates to stakeholders.

6. Documentation and Change Control: Maintain documentation that captures the project's design, architecture, and installation instructions. Establish a change control process to track and manage modifications to the software artifacts. Utilize issue tracking systems like Jira or GitHub Issues to capture and prioritize feature requests, bug reports, and change requests.

Adhering to SCM practices ensures proper versioning, reproducibility, and traceability of the software artifacts throughout the project's lifecycle. It also helps in maintaining a stable and manageable codebase, facilitating collaboration among team members, and reducing the risk of errors and inconsistencies.

Quality Assurance Plan:

A well-defined Quality Assurance (QA) plan is crucial for ensuring the accuracy, reliability, and performance of the predictive maintenance project. Here is a suggested QA plan for the project:

1. Test Strategy: Define a comprehensive testing strategy that outlines the various types of testing to be performed, such as unit testing, integration testing, system testing, and acceptance testing. Identify the testing tools and frameworks to be used, and establish guidelines for test case creation, execution, and reporting.

2. Test Environment: Set up a dedicated test environment that closely resembles the production environment. This environment should include the necessary hardware, software, and data configurations required for testing. Ensure that the test environment is isolated from the production environment to prevent any impact on live systems.

3. Test Data: Prepare a representative and diverse set of test data that covers different machine types, operating conditions, and failure modes" in the Test Data section. The test data should include both historical and simulated data to validate the accuracy and robustness of the predictive models. Ensure the test data covers different machine types, operating conditions, and failure modes.

4. Test Execution: Develop and execute test cases based on the defined testing strategy. Test the various components of the project, including data preprocessing, model training, evaluation, and prediction. Conduct both functional and non-functional testing to validate the correctness of the models, accuracy of predictions, and performance under varying conditions.

5. Bug Tracking and Reporting: Implement a bug tracking system to capture and prioritize issues identified during testing. Assign severity levels to each bug and track their resolution. Generate regular test reports that summarize the test coverage, test results, and identified issues. Communicate test findings to the development team for prompt resolution.

6. Documentation: Maintain comprehensive documentation that describes the testing process, test cases, and test results. Document any issues encountered, their resolution, and lessons learned throughout the project. This documentation serves as a reference for future iterations, enhancements, and maintenance of the predictive maintenance system.

7. User Acceptance Testing: Involve end users, such as maintenance engineers or industry experts, in the testing process. Conduct user acceptance testing to gather feedback on the usability and effectiveness of the system. Incorporate user feedback to improve the user interface, interpretability of results, and overall user experience.

By following a well-structured QA plan, the predictive maintenance project can ensure the reliability and accuracy of the predictive models, enhance user satisfaction, and mitigate risks associated with incorrect predictions or system failures.

Risk Management:

Effective risk management is essential to identify, assess, and mitigate potential risks that may impact the success of the predictive maintenance project. Here is a suggested approach to risk management for the project:

1. Risk Identification: Identify potential risks that may arise during different stages of the project, such as data collection, preprocessing, model development, training, and deployment. Risks could include data quality issues, inaccurate predictions, infrastructure failures, or delays in project timelines. Involve the project team, domain experts, and stakeholders to brainstorm and identify risks.

2. Risk Assessment: Assess the identified risks based on their likelihood of occurrence and potential impact on the project. Prioritize risks based on their severity and potential consequences. This assessment helps in allocating appropriate resources and attention to high-priority risks.

3. Risk Mitigation Strategies: Develop strategies to mitigate and control identified risks. This may include measures such as data validation and cleaning processes, implementing backup and recovery mechanisms, conducting thorough testing and validation of models, and ensuring scalability and robustness of the deployment infrastructure. Assign responsibilities for risk mitigation activities to the appropriate team members.

4. Risk Monitoring and Contingency Planning: Continuously monitor and review identified risks throughout the project lifecycle. Regularly assess the effectiveness of risk mitigation strategies and update them as necessary. Develop contingency plans to address unforeseen risks or events that may impact the project. Maintain open communication channels within the team to promptly address emerging risks.

5. Stakeholder Communication: Communicate identified risks, mitigation strategies, and progress in addressing risks to project stakeholders. Keep stakeholders informed about any potential impacts on project timelines, deliverables, or expected outcomes. Engage stakeholders in risk management discussions and decision-making processes.

6. Documentation: Maintain a risk register or risk log that documents identified risks, their assessment, mitigation strategies, and outcomes. This documentation serves as a reference for future projects and helps in sharing lessons learned and best practices.

By proactively identifying and managing risks, the predictive maintenance project can minimize potential disruptions, enhance project success, and ensure the reliability and usefulness of the predictive models for farmers.

Task and Data 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;

Exploratory Analysis

Our data exploration starts by checking that each entry is unique and there are no duplicates; this is done by verifying 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.

```
Features non-null values and data type:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 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   Target                   10000 non-null  int64
    9   Failure Type             10000 non-null  object

dtypes: float64(3), int64(4), object(3)
memory usage: 781.4+ KB
Check for duplicate values: False
```

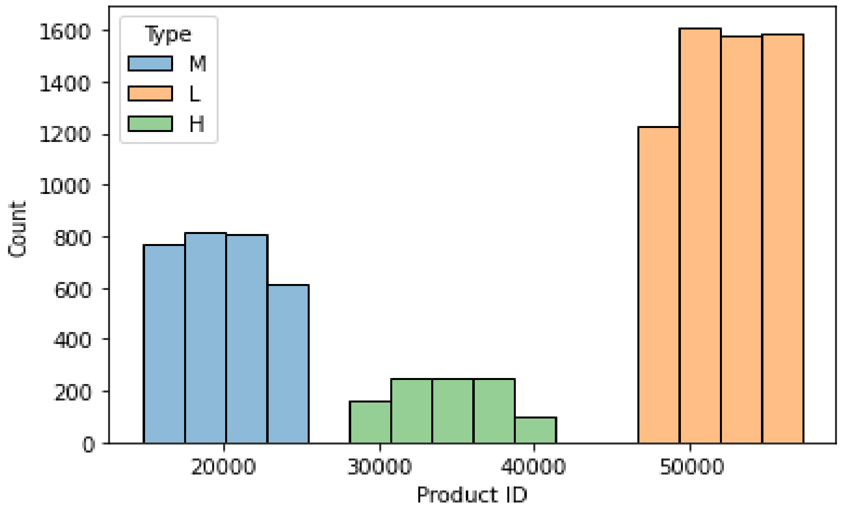
To sum up even more:

- There is no missing data;
- There are no duplicate values;
- Six columns are numerical features, including UDI;
- Three columns are categorical features, including ProductID.

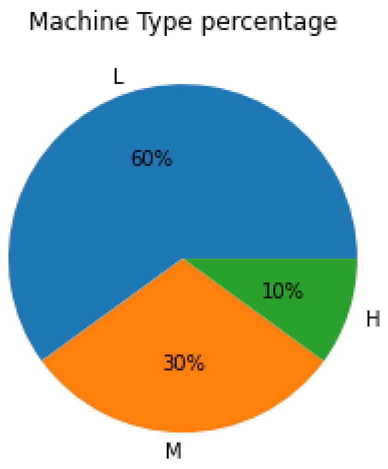
To make this distinction more clear we set numeric columns to float type.

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 a 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:



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



TARGET 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 with respect to dataset's description is that when the failure is random (RNF), the Machine Failure feature is not set to 1.

	Target	Failure Type
1221	0	Random Failures
1302	0	Random Failures
1748	0	Random Failures
2072	0	Random Failures
2559	0	Random Failures
3065	0	Random Failures
3452	0	Random Failures
5471	0	Random Failures
5489	0	Random Failures
5495	0	Random Failures
5509	0	Random Failures
5553	0	Random Failures
5639	0	Random Failures
6091	0	Random Failures
6913	0	Random Failures
6960	0	Random Failures
7488	0	Random Failures
7868	0	Random Failures

Fortunately the machine failure RNF occurs in only 18 observations and it has a random nature therefore not predictable so we decide to remove these rows.

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.

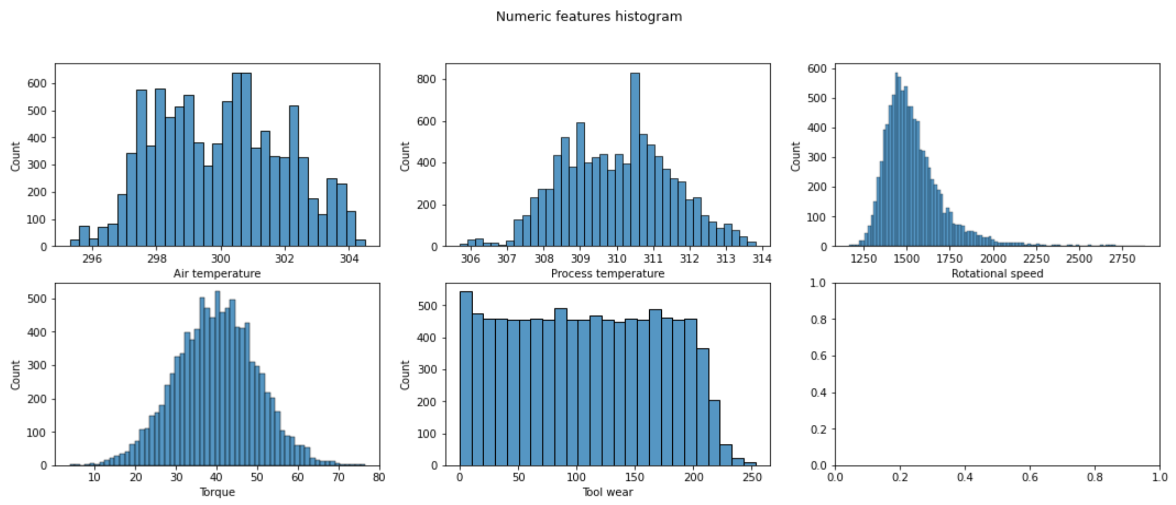
Target	Failure	Type
1437	1	No Failure
2749	1	No Failure
4044	1	No Failure
4684	1	No Failure
5536	1	No Failure
5941	1	No Failure
6478	1	No Failure
8506	1	No Failure
9015	1	No Failure

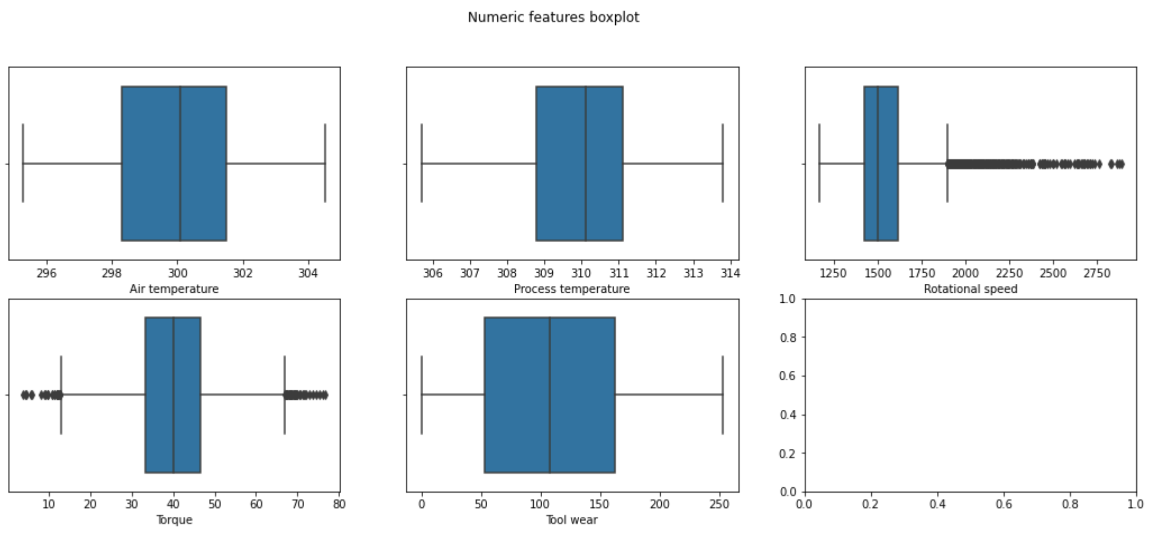
Our actions did not alternate the original data very much.

OUTLIERS INSPECTION

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

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.



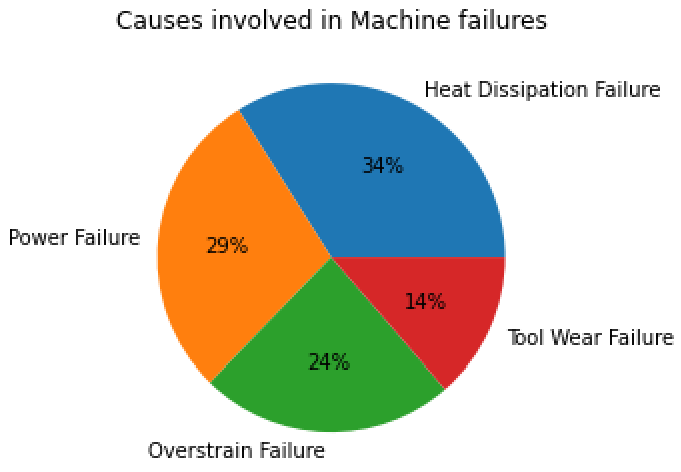


The boxplots highlight possible outliers in the features mentioned above, however in the case of Torque these are probably traceable to the way outliers are detected using boxplots (since the distribution is Gaussian it would be more appropriate to use the 3σ rule instead of the IQR); in the case of 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.

RESAMPLING WITH SMOTE

Another important consideration regards the extremely low occurrence of machine failures among the entire dataset, which percentage is equal only to 3.31%. Moreover, a pie plot showing the occurrence of the causes involved for each failure reveals a further degree of imbalance.

```
Failures percentage in data: 3.31
```



When dealing with machine learning problems, class imbalance is a great concern, as it can mislead both the models' training process and our ability to interpret their results. For 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 behavior 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.

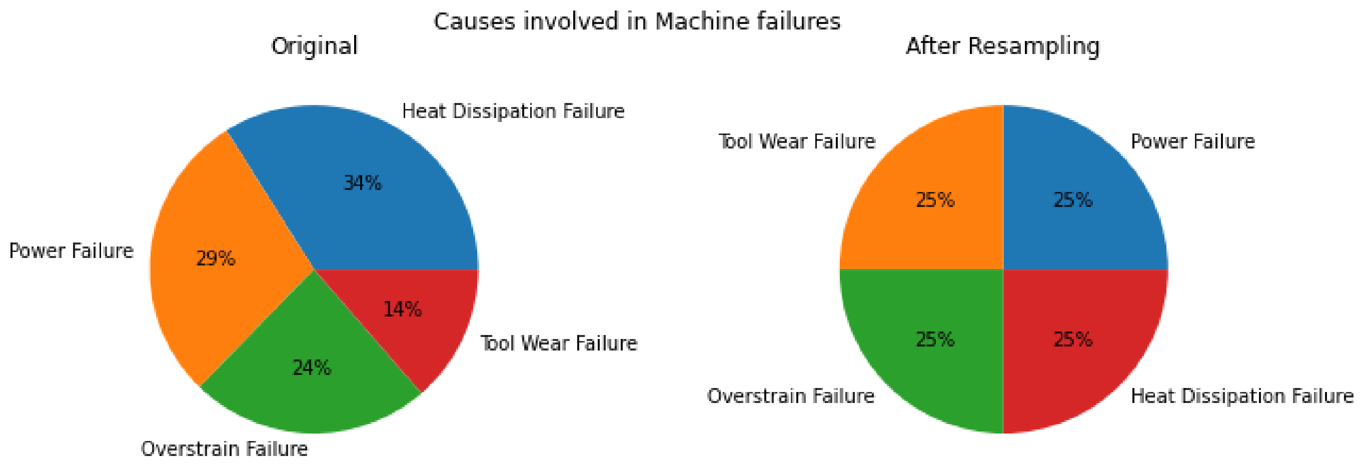
Among the most common data augmentation techniques we identify:

- 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).

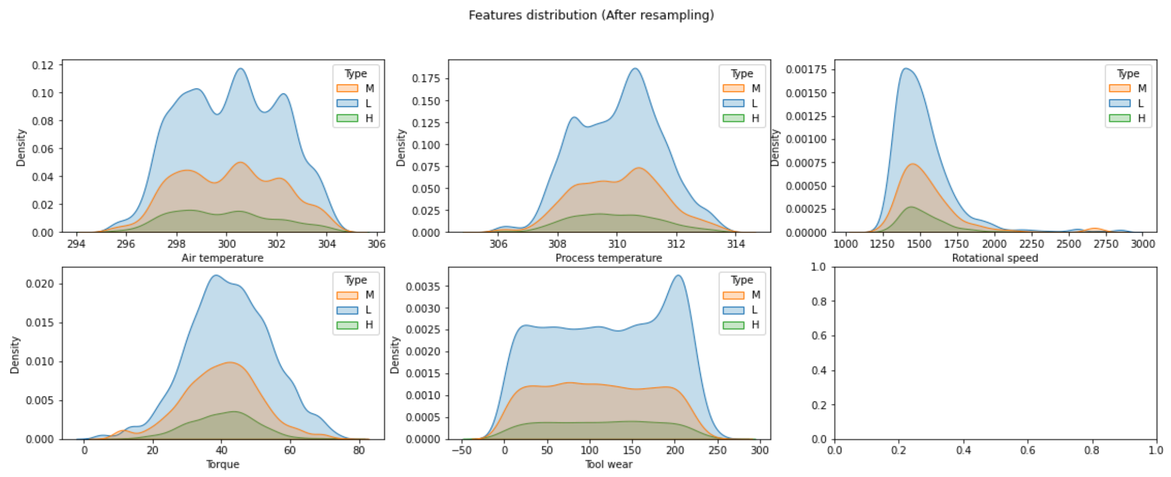
The first two choices however result in extremely simplistic approaches; in particular the first one has the disadvantage of decreasing the length of the dataset in a context in which the available data are already limited. Therefore we use the SMOTE procedure to generate new samples, which is very much 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.

Comparison after resampling

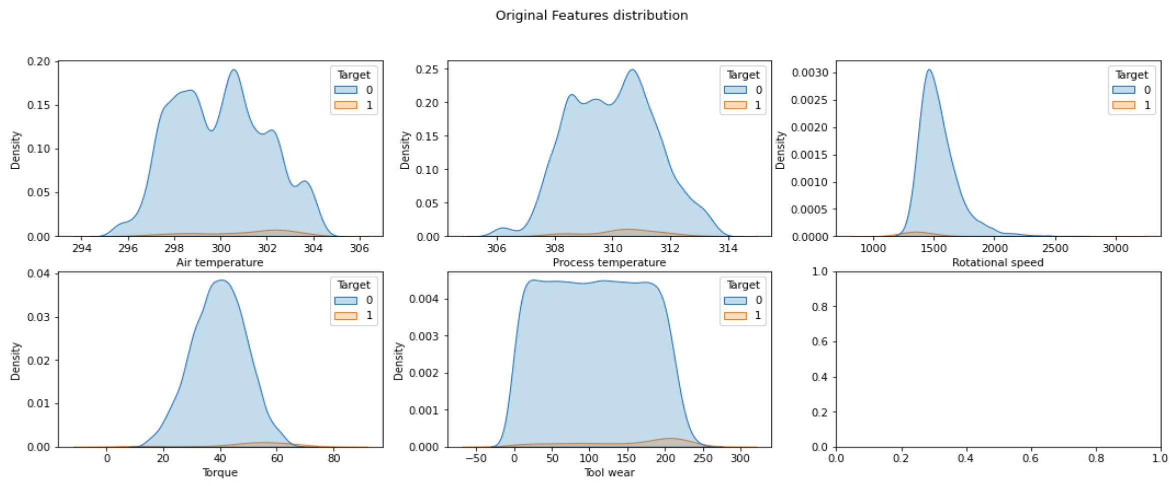
The result is described in the following pie charts.

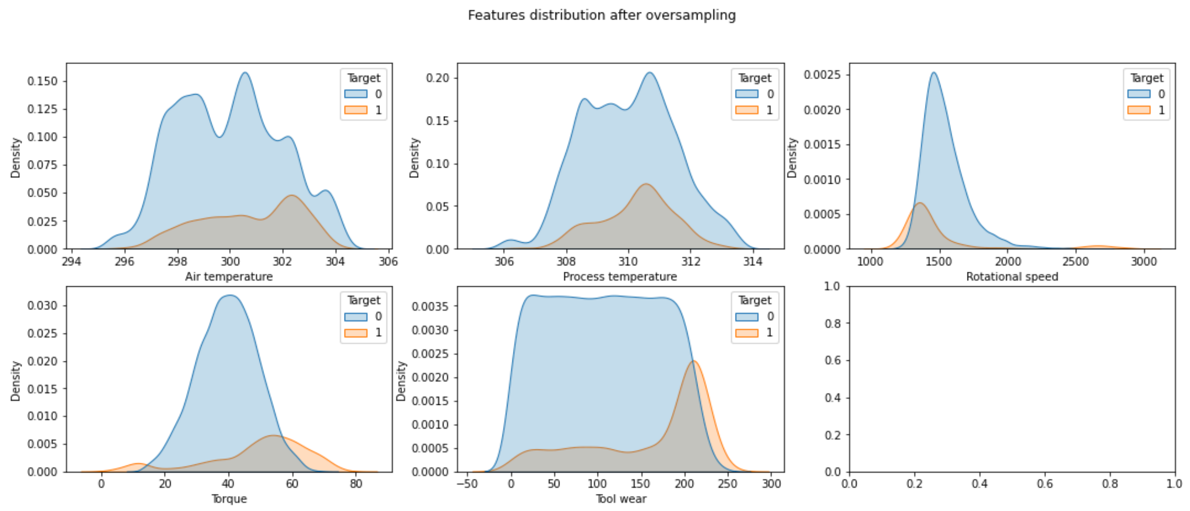


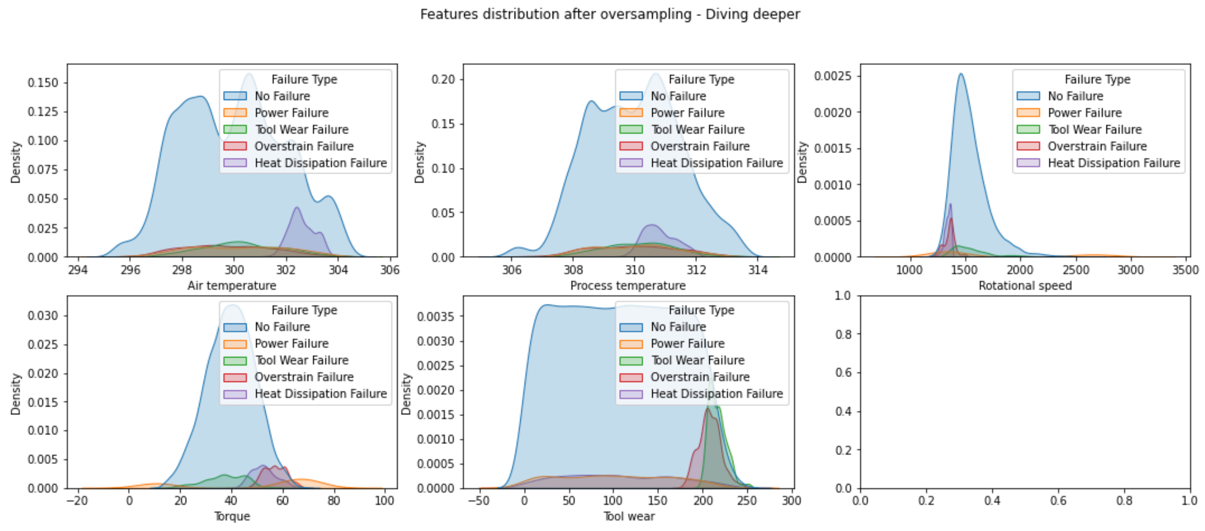
As one can expect, the cases of Machine Failure mainly concern low quality machines, then those of medium quality and only a few times those of high quality. This difference is accentuated when the number of observations of non-functioning machines is (artificially) increased. However, from the kdeplots 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 reaso



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







The first thing we can observe is that the data augmentation was performed successfully, as the feature distribution for faulty instances 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.

FEATURES SCALING AND ENCODING

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

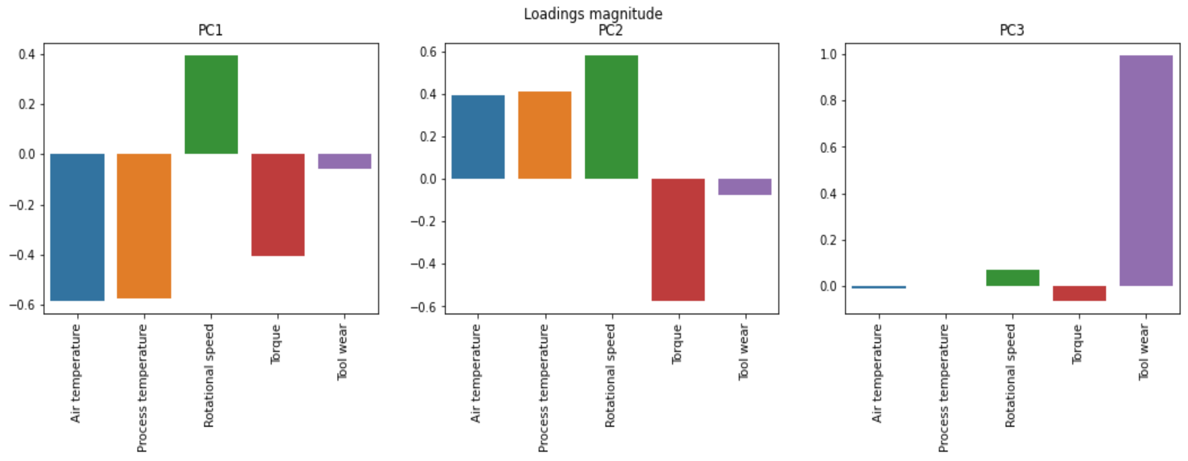
- First, we apply a label encoding to the categorical columns, since Type is an ordinal feature and Cause must be represented in one column. The mapping follows this scheme: Type: {L=0, M=1, H=2} Cause: {Working=0, PWF=1, OSF=2, HDF=3, TWF=4}
- Secondly we perform the scaling of the columns with StandardScaler.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

PCA AND CORRELATION HEATMAP

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

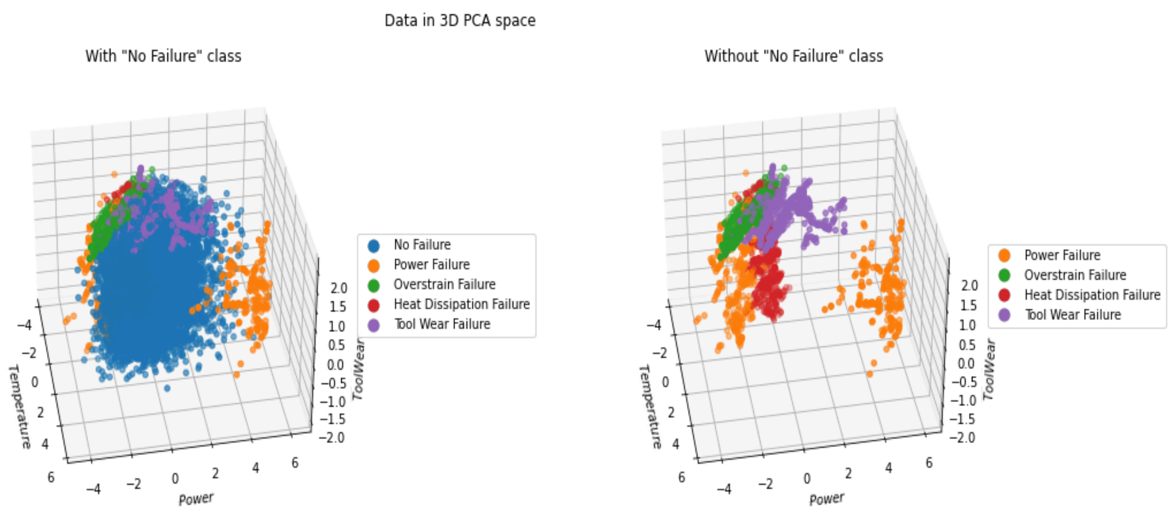
```
Explained variance ratio per component:
PC1    37.69
PC2    36.81
PC3    19.84
PC4     3.08
PC5     2.58
dtype: float64
Explained variance ratio with 3 components: 94.34
```

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.



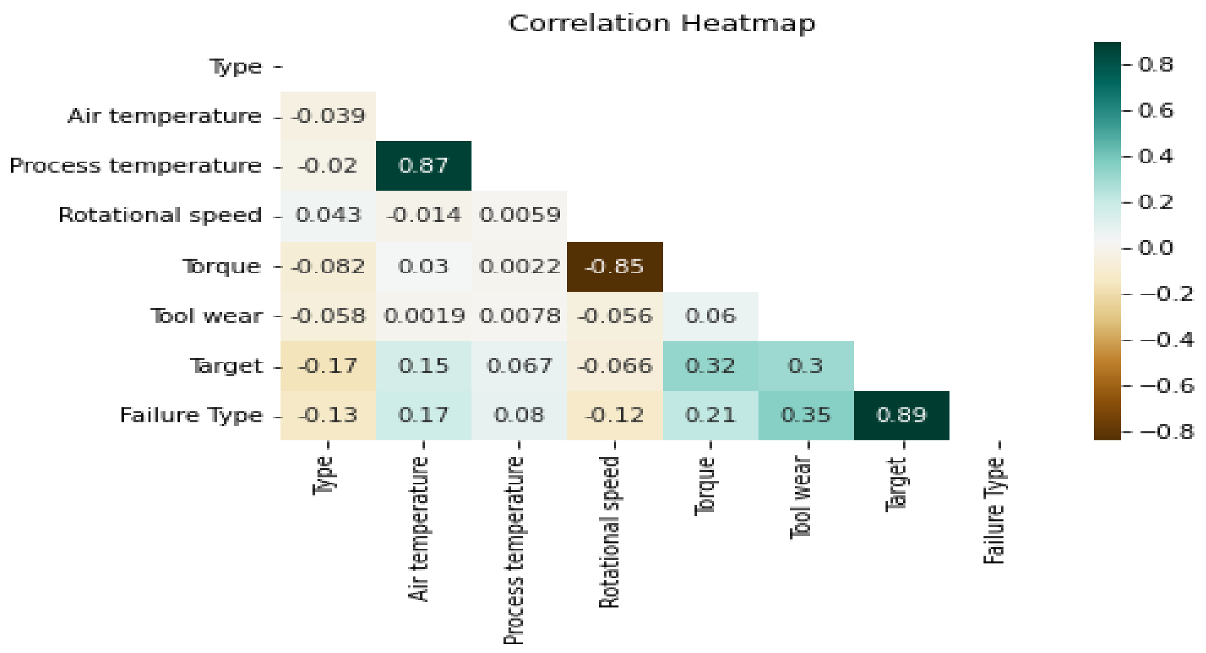
The bar plot of Principal Components weights makes easy to understand what they represent:

- 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.



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; 18
- 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.



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=(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(Precision∗Recall)/(Precision+Recall).** Although generally effective, 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: **Fβ=(1+β^2)(Precision∗Recall)/(β^2∗Precision+Recall)**

With the choice β = 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.

BINARY TASK

PRELIMINARIES

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.

FEATURE SELECTION ATTEMPTS

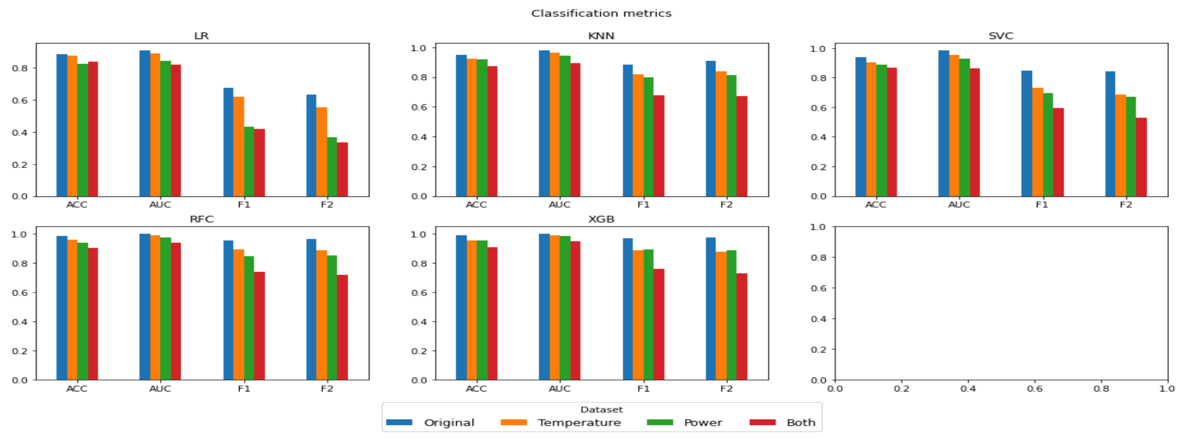
Before going into the training of the models just mentioned we try to perform feature selection, exploiting the considerations we made about the correlation heatmap and the exploratory data analysis: just to remind, we noticed that the features "Process temperature" and "Air temperature" are positively correlated, and "Torque" and "Rotational speed" are negatively correlated. From the dataset 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;

- combine the previous operations.

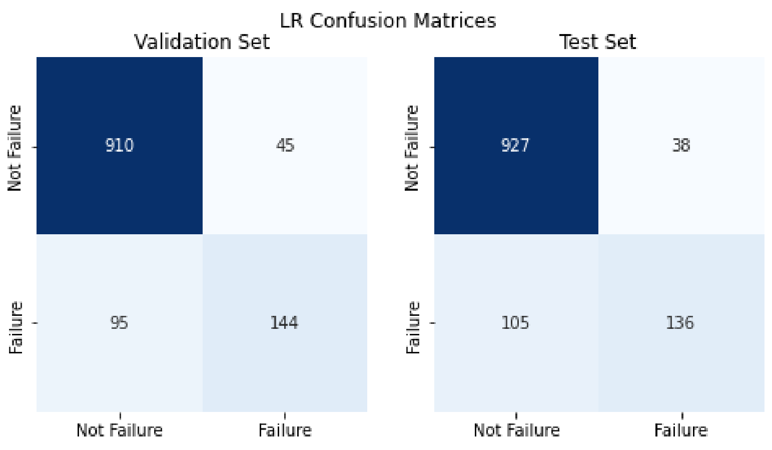


From the results obtained, we observe that all the models applied to the entire dataset perform better than when they are applied to the ones created by reducing the number of features. The best performances and the modest number of features from which our dataset is composed encourage us to opt to avoid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

LOGISTIC REGRESSION BENCHMARK

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.

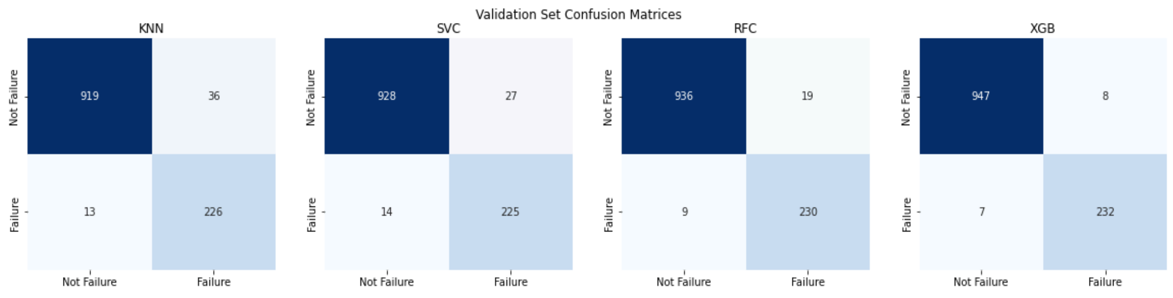
```
Validation set metrics:
ACC    0.883
AUC    0.905
F1     0.673
F2     0.629
dtype: float64
Test set metrics:
ACC    0.881
AUC    0.917
F1     0.655
F2     0.598
dtype: float64
```

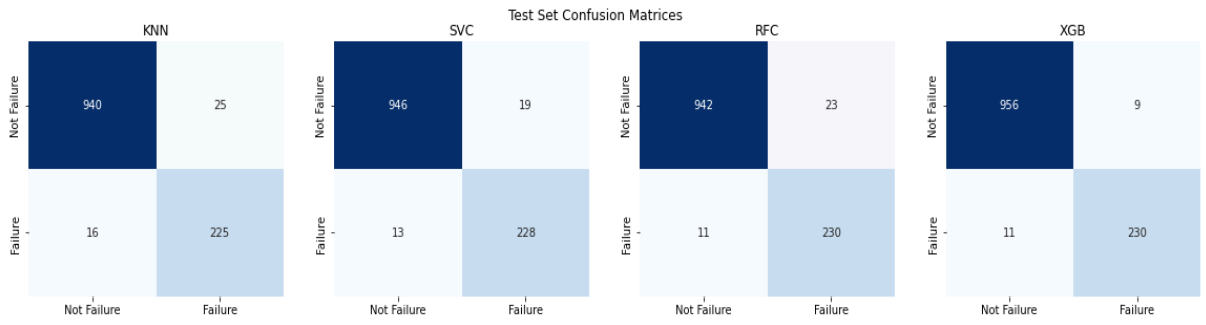


	feature	odds
4	Torque	16.696209
3	Rotational speed	9.394822
1	Air temperature	4.462500
5	Tool wear	3.483306
0	Type	0.520599
2	Process temperature	0.348815

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.

MODELS





```
 Validation scores:
          KNN    SVC    RFC    XGB
     ACC  0.959  0.966  0.977  0.987
     AUC  0.954  0.987  0.997  0.999
     F1   0.902  0.916  0.943  0.969
     F2   0.928  0.931  0.954  0.970
```

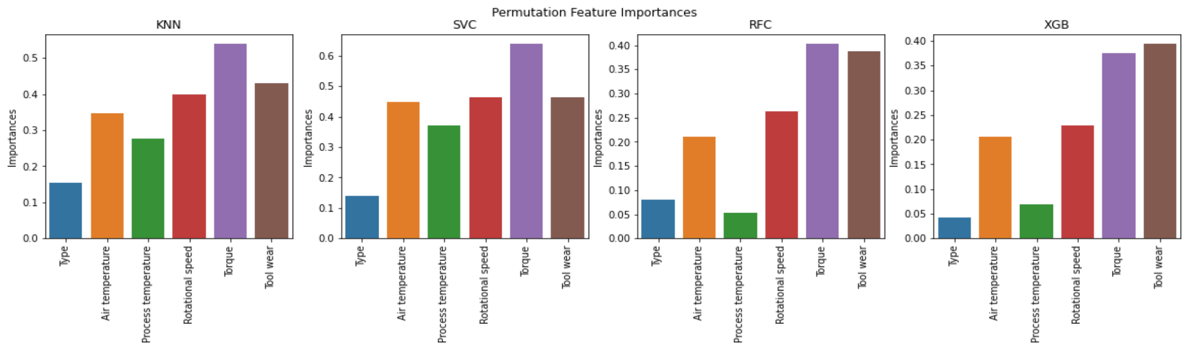
```
Test scores:
          KNN    SVC    RFC    XGB
    ACC  0.966  0.973  0.972  0.983
    AUC  0.954  0.992  0.997  0.998
     F1  0.916  0.934  0.931  0.958
    F2  0.927  0.941  0.945  0.956
```

All the selected models obtain similar results on the validation set (except KNN which is a little worse) and it is difficult 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 one to have a complete understanding of how the algorithm works. 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.



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.

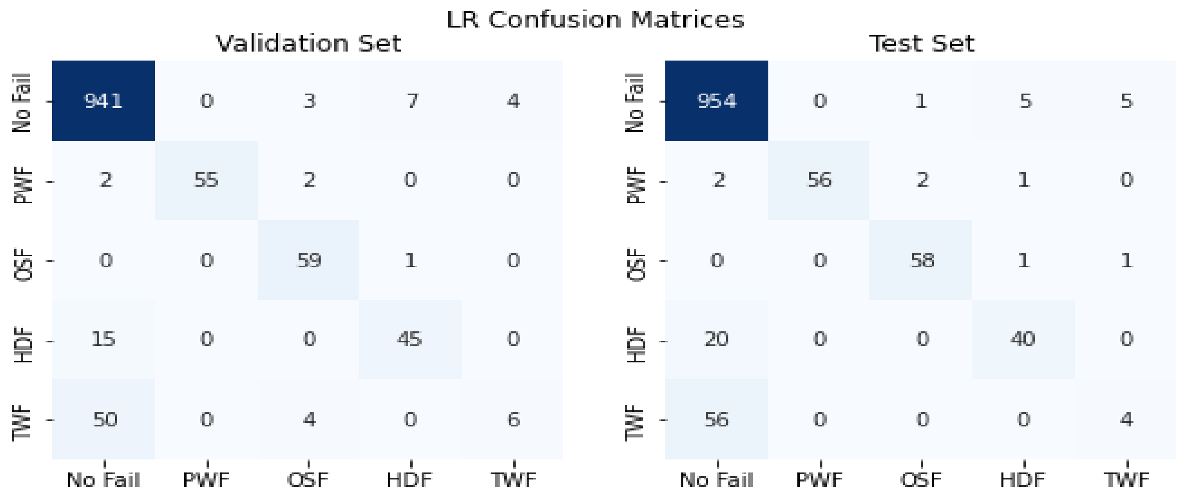
MULTICLASS TASK

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 machines and 20% that fail. As for binary classification tasks, we choose Logistic Regression as a 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 decided to use the "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.

LOGISTIC REGRESSION BENCHMARK

First let’s look at how the Logistic Regression behaves:

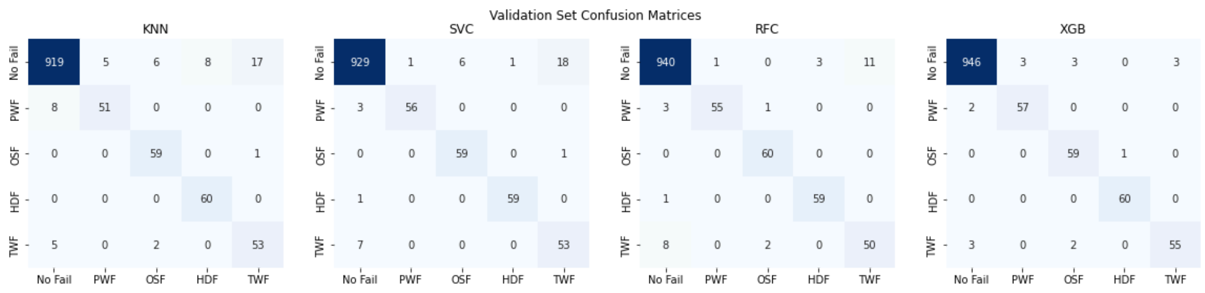


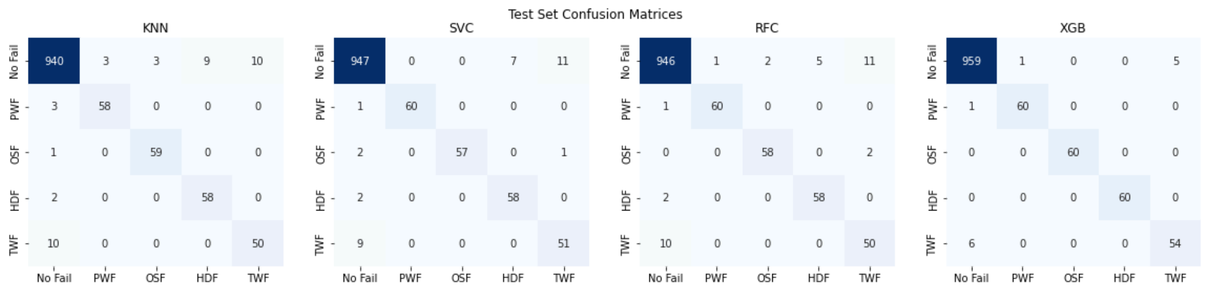
	Type	Air temperature	Process temperature	Rotational speed	Torque	Tool wear
No Failure	1.920864	0.224090	2.866848	0.106442	0.059894	0.287084
Power Failure	0.697373	0.824558	0.982471	944.048373	2822.376842	0.744856
Tool Wear Failure	0.036806	0.215535	2.777322	0.108932	8.748803	406.189304
Overstrain Failure	0.636280	4314.687600	0.004253	0.000458	0.398540	0.711005
Heat Dissipation Failure	1.045501	1.337837	0.739732	0.225326	0.118551	749.948357

In the table above 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.

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.



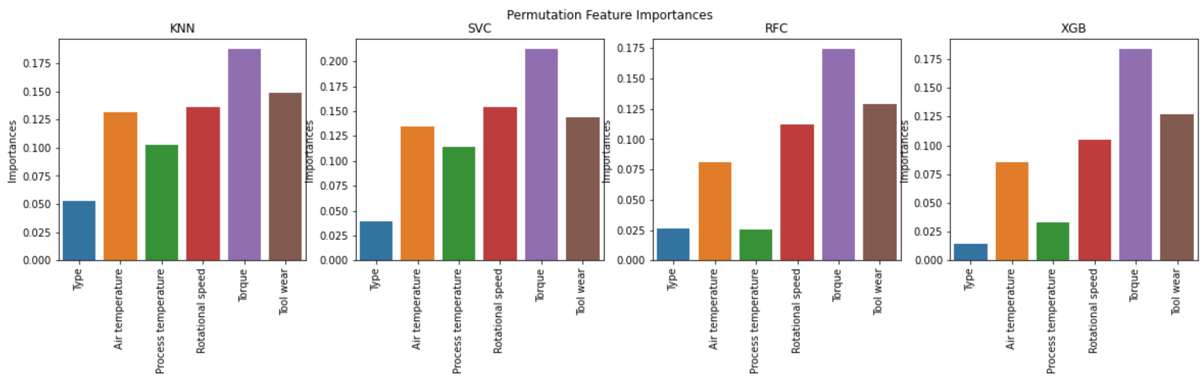


```
Validation scores:
       KNN    SVC    RFC    XGB
ACC   0.956  0.968  0.975  0.986
AUC   0.956  0.993  0.998  0.999
F1    0.957  0.969  0.975  0.986
F2    0.957  0.968  0.975  0.986
```

```
Test scores:
       KNN    SVC    RFC    XGB
ACC   0.966  0.973  0.972  0.989
AUC   0.956  0.995  0.997  0.999
F1    0.966  0.973  0.972  0.989
F2    0.966  0.973  0.972  0.989
```

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.

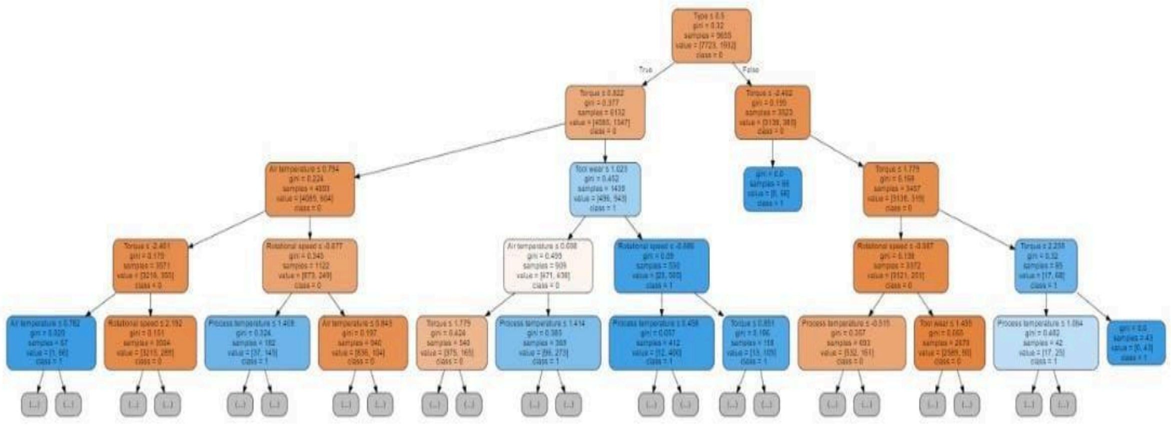
All other models perform better than the benchmark and they obtain high values for the chosen metrics both for validation and test set. SVC and RFC's performances are very similar to each other and XGB performs better than them. If we look at the training phase, SVC and RFC take the same time, while 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.

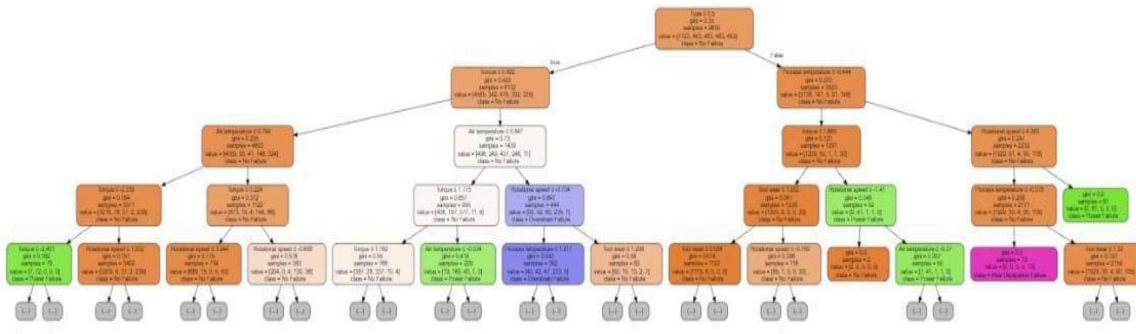


From previous barplots we see that the models give more importance to Torque, Tool wear and Rotational Speed while the Type contribution is very low. This is in accordance with the observations made in the exploration of the dataset in Section 1-2 and it is consistent with the Permutation Feature Importances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

DECISION PATHS

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.





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.

Future Scope

The field of predictive maintenance is poised for significant advancements as industries increasingly adopt digital transformation strategies. One promising area of future development is the integration of advanced machine learning algorithms and artificial intelligence techniques. These technologies have the potential to enhance the accuracy and robustness of predictive models, enabling more precise failure predictions and maintenance scheduling. Additionally, the incorporation of real-time data analytics through the Internet of Things (IoT) can provide continuous monitoring and immediate insights into machine health, leading to proactive and timely maintenance actions. This integration can minimize downtime, extend equipment lifespan, and optimize resource allocation, driving greater operational efficiency and cost savings.

Another exciting avenue for future research in predictive maintenance is the development of more sophisticated and interpretable models. As industries demand not only accurate predictions but also an understanding of the underlying factors contributing to machine failures, there is a growing need for models that offer transparency and explainability. Techniques such as explainable AI (XAI) can provide insights into how models make decisions, fostering trust and enabling maintenance teams to make informed decisions based on model outputs. Furthermore, the integration of predictive maintenance systems with enterprise resource planning (ERP) and maintenance management software can streamline workflows and improve coordination between different departments, ensuring that maintenance activities are seamlessly integrated into overall business operations. As these technologies and methodologies evolve, predictive maintenance will become an indispensable component of modern industrial practices.

References and Bibliography

- https://www.datarobot.com/wiki/predictive-maintenance/
- https://www.kaggle.com/code/gerardocappa/predictive-maintenance-final-project/input
- https://itrexgroup.com/blog/machine-learning-predictive-maintenance/
- https://arxiv.org/pdf/2205.09402