Machine Learning Project Report

MetroPT-3 Dataset

Student list:

Full name	Group	Section
Mohanned Kadache	03	01
Mohamed Nadjib Bentayeb	01	01
Larbi Saidchikh	04	02

Abstract

This project explores the MetroPT-3 dataset, a valuable resource for developing predictive maintenance models in metro trains. The dataset includes real world sensor readings from key components like pressure, temperature, motor current and air intake valves within a compressor's Air Production Unit (APU).

The main task of the project is to use this dataset to develop various robust predictive models tailored for failure prediction and anomaly detection and other maintenance-related tasks. This includes various established techniques like Linear Regression, Naive Bayes, KNN, SVM, and Random Forest to advanced methods including XGBoost, and Neural Networks.

One of the most important aspects of our methodology is the process of hyperparameter tuning, ensuring that each model is optimized. The optimization enhances the model predictive abilities, paving the way for more accurate identification of the maintenance issues.

Our project leads up in a comparative analysis of these models, offering metro train operators a powerful toolbox for proactive maintenance. This analysis not only empowers them with actionable insights but also sheds light on the strengths and limitations of various machine learning approaches for real-world predictive maintenance challenges. This, in turn, contributes significantly to the advancement of the field within the metro train industry.

Table of content

Introduction

The prediction of Air Production Unit (APU) failures in metro train operations is crucial for maintaining smooth operations and preventing costly disruptions. Traditional approaches relied on scheduled maintenance, rule-based systems and statistical analysis, but the dynamic nature of APUs demands a more proactive solution, yet these approaches often fall short in capturing the nuanced patterns indicative of impending failures.

This project tackles this challenge by leveraging the MetroPT-3 dataset. This unique resource contains real-world sensor readings from a compressor's APU, including pressure, temperature, and motor current. This rich data offers a powerful window into the complex interplay of factors affecting APU health.

Machine Learning emerges as a transformative solution to this problem, as it empowers us to develop predictive models that are able to analyze relationships within the MetroPT-3 dataset. These models can identify potential APU issues before they escalate, leading to optimized maintenance schedules and a significant boost in overall metro train reliability.

Dataset Description

The dataset was collected to support the development of predictive maintenance, anomaly detection, and remaining useful life (RUL) prediction models for compressors using deep learning and machine learning methods.

It consists of multivariate time series data obtained from several analogue and digital sensors installed on the compressor of a train. The data span between February and August 2020 and includes 15 signals, such as pressures, motor current, oil temperature, and electrical signals of air intake valves. The monitoring and logging of industrial equipment events, such as temporal behavior and fault events, were obtained from records generated by the sensors. The data were logged at 1Hz by an onboard embedded device. The fig1 represents a visual aid to visualize the sensors that are used.

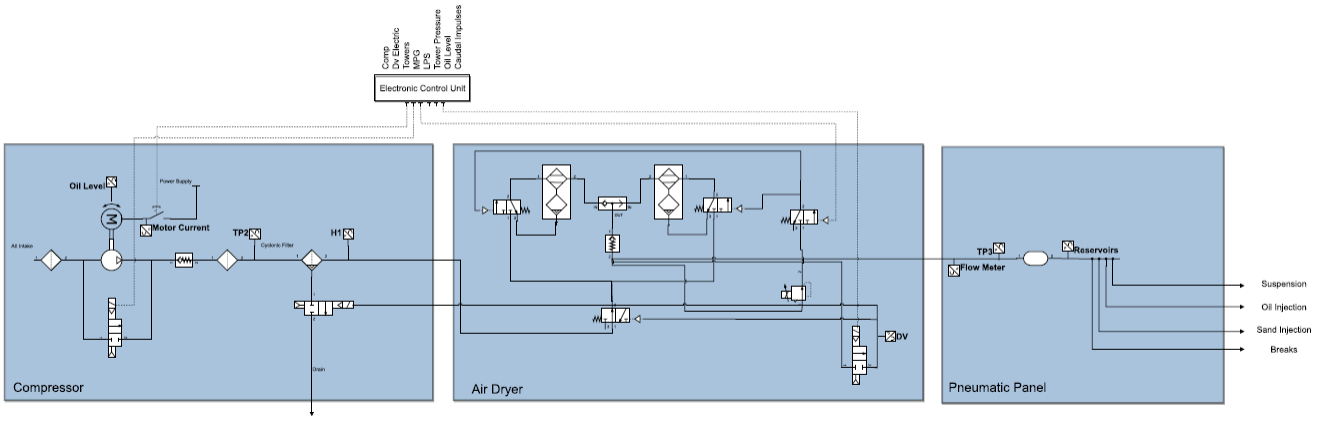


Figure 1: Air Producing Unit (APU)

The dataset consists of 1,516,948 instances, and It can be characterized as Tabular, Multivariate, and Time-Series. A description of the dataset, including feature names and types is shown below:

1. TP2 (bar) – the measure of the pressure on the compressor.

2. TP3 (bar) – the measure of the pressure generated at the pneumatic panel.

Dataset Description

3. H1 (bar) – the measure of the pressure generated due to pressure drop when the discharge of the cyclonic separator filter occurs.

4. DV pressure (bar) – the measure of the pressure drop generated when the towers discharge air dryers; a zero reading indicates that the compressor is operating under load.

5. Reservoirs (bar) – the measure of the downstream pressure of the reservoirs, which should be close to the pneumatic panel pressure (TP3).

6. Motor Current (A) – the measure of the current of one phase of the three-phase motor; it presents values close to

0A - when it turns off, 4A - when working offloaded, 7A - when working under load, and 9A - when it starts working.

7. Oil Temperature (ºC) – the measure of the oil temperature on the compressor.

8. COMP - the electrical signal of the air intake valve on the compressor; it is active when there is no air intake, indicating that the compressor is either turned off or operating in an offloaded state.

9. DV electric – the electrical signal that controls the compressor outlet valve; it is active when the compressor is functioning under load and inactive when the compressor is either off or operating in an offloaded state.

10. TOWERS – the electrical signal that defines the tower responsible for drying the air and the tower responsible for draining the humidity removed from the air; when not active, it indicates that tower one is functioning; when active, it indicates that tower two is in operation.

11. MPG – the electrical signal responsible for starting the compressor under load by activating the intake valve when the pressure in the air production unit (APU) falls below 8.2 bar; it activates the COMP sensor, which assumes the same behavior as the MPG sensor.

Dataset Description

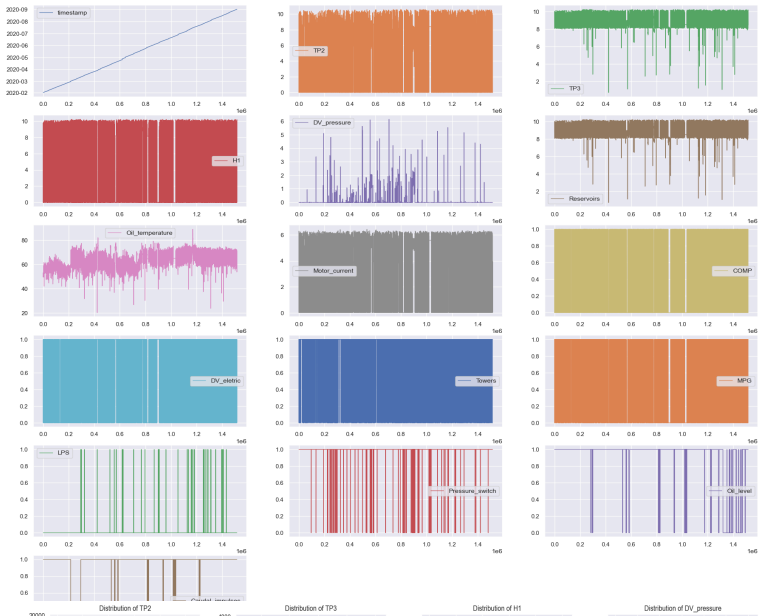
12. LPS – the electrical signal that detects and activates when the pressure drops below 7 bars.

13. Pressure Switch - the electrical signal that detects the discharge in the air-drying towers.

14. Oil Level – the electrical signal that detects the oil level on the compressor; it is active when the oil is below the expected values.

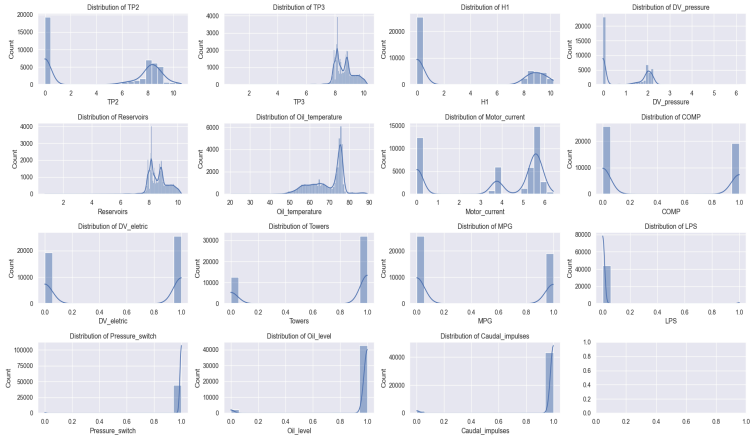
15. Caudal Impulse – the electrical signal that counts the pulse outputs generated by the absolute amount of air flowing from the APU to the reservoirs.

Figure 2: Visualization of the time series



We visualized sensor data (February 2020 to August 2020) over time to identify trends. Most features exhibited significant fluctuations, making it difficult to discern clear patterns visually.

Figure 3: Visualization of the distribution plots



We created histograms to understand the distribution of each sensor value. This helped identify:

- Binary features (8 out of 15) with distinct peaks at 0 and 1.

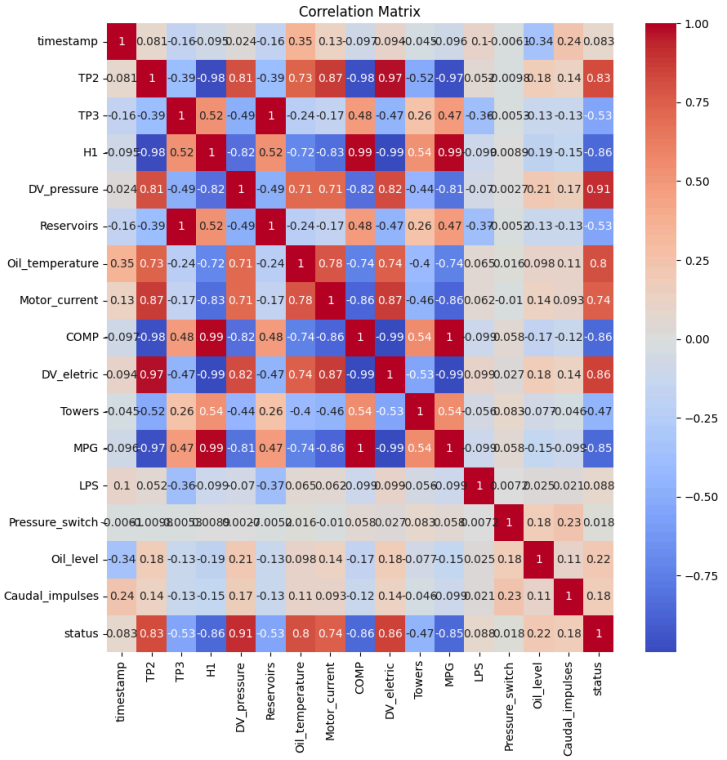
Dataset Description

- Skewed distributions for features like TP3, H1, and Reservoirs.

- A normal distribution for Oil_temperature.

- Identical distributions for TP3 and Reservoirs, likely due to their inherent relationship (as explained in the data description).

Figure 4: Correlation matrix



We calculated correlations between sensor readings to identify potential relationships. Notably:

- TP3 and Reservoirs, and COMP and MPG displayed very high positive correlations (1 and 0.98), suggesting a potential link between these features. However, we decided to retain both features due to their importance to the system (potentially indicating air leaks).

- DV Electric exhibited a strong positive correlation (0.67) with the "Air Leak" label, suggesting a higher likelihood of air leaks when the compressor operates under load.

These analyses provided valuable insights into the sensor data, laying the groundwork for the next steps: implementing methods for effective air leak identification and mitigation (as described in subsequent sections).

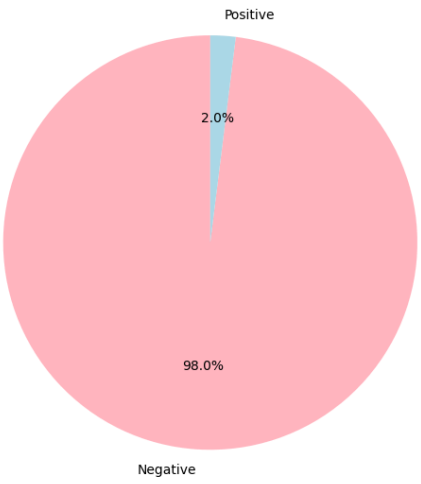
Methodology

Feature Engineering

It is important to note that our initial dataset was unlabelled which means there was no attribute to describe the status of the APU failure. but the failure reports provided by the company are available in the following table:

Nr.	Start Time	End Time	Failure	Severity	Report
#1	4/18/2020 0:00	4/18/2020 23:59	**Air leak**	High stress	
#1	5/29/2020 23:30	5/30/2020 6:00	**Air Leak**	High stress	Maintenance on 30Apr at 12:00
#3	6/5/2020 10:00	6/7/2020 14:30	**Air Leak**	High stress	Maintenance on 8Jun at 16:00
#4	7/15/2020 14:30	7/15/2020 19:00	**Air Leak**	High stress	Maintenance on 16Jul at 00:00

Figure 5: Failure Information report



This shows that the Air Leaks happened only 4 times. According to this, all the data points that fall within the stated timestamps have an air leak, thus have a value = 1 in the Status column. Now that the timestamp is no longer necessary it has been dropped from the dataset.

Instances with Air Leaks constituted of approximately 2% only of the dataset

Figure 6: Pie chart of the imbalanced class distribution

Methodology

Balancing Data

As the task for this study is to predict failures and the need for maintenance, we know that failures usually occur rarely. Thus, we face an imbalanced dataset where the negative label **(0)** indicating no failures outweighs the positive label **(1)** by around 50 times. For unbiased and accurate models, we ought to balance out training data with one of these techniques:

Undersampling: reduces the number of instances in the majority class to match the number of minority class instances by randomly selecting them.

Oversampling: increases the number of instances of the minority class by replicating or generating new instances.

SMOTE (Synthetic Minority Oversampling Technique): generates synthetic instances by interpolating between existing instances in the minority class.

The choice of undersampling in our project is due to the fact that it is fast and intuitive. Also, we already have a large dataset. Undersampling avoids further size increase.

Outliers

Outliers can sometimes point towards unexpected events or equipment malfunctions not captured by the "normal" data patterns. These deviations can trigger further investigation, potentially revealing underlying issues that contribute to air leaks. If we remove them during data exploration, we might miss crucial signs of a developing problem. This could lead to delayed detection and potentially larger issues down the line. Furthermore, Machine learning models for anomaly detection learn from the data they are trained on. If outliers, which represent potential air leaks, are removed from the training data, the model might not be able to recognize them as anomalies in unseen data. This could lead to the model missing actual air leaks during operation.

Methodology

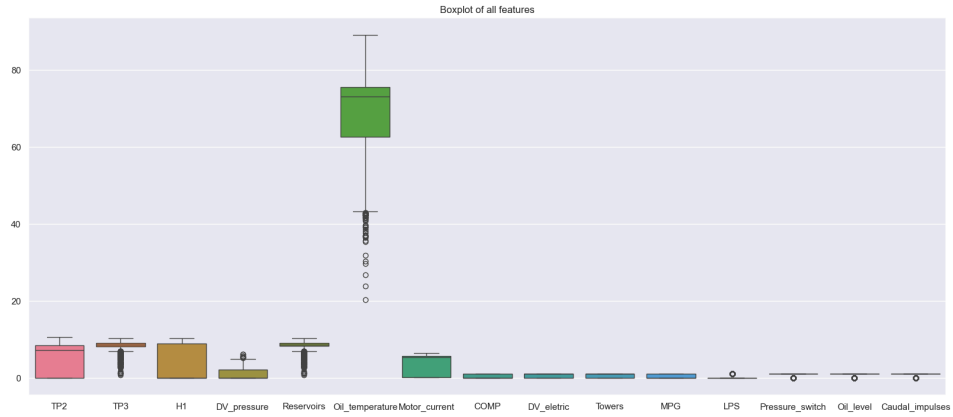


Figure 7: Box-plots of the Outliers

Features Selection

To select the most appropriate features for our models. We will take into consideration two specific criteria. The first one is using **F-test** to rank the features based on their importance in relation to the **status** and plot them ordered. The second aspect is taking into account the correlation between features to drop the highly correlated ones.

Based on the first criterion, we select **{ DV_pressure, DV_eletric, COMP, H1, MPG, TP2, Oil_temperature, Motor_current, TP3, Reservoirs, Towers, Caudal_impulses }**.

Adding to that, the second criterion which shows via the correlation heatmap that **TP2**, **TP3** and **MPG** need to be removed due to their high correlations with other more important features. Thus, the final set of descriptive features is **{ DV_pressure, DV_eletric, COMP, H1, Oil_temperature, Motor_current, Reservoirs, Towers, Caudal_impulses }**.

Methodology

Machine Learning Algorithms

Several machine learning models were trained and assessed:

1. Decision Trees: these were implemented to understand the importance of different features and their interactions.

2. Random Forests: by averaging the results of multiple decision trees, this ensemble method aimed to improve prediction accuracy and robustness.

3. K-Nearest Neighbors: KNN was utilized as a non-parametric algorithm to determine the class of a sample by considering the majority class among its k nearest neighbors in the feature space. The value of k was selected through hyperparameter tuning to strike a balance between underfitting and overfitting.

4. Naive Bayes: Gaussian Naive Bayes was used due to the numerical nature of our features, with hyperparameter tuning applied to enhance performance. Complement Naive Bayes was also tested to handle data imbalance.

5. Support Vector Machines (SVM): SVMs were employed to find the optimal hyperplane for classification. We also performed hyperparameter tuning to optimize their performance.

6. Artificial Neural Networks (ANN): ANNs were explored to capture complex patterns and non-linear relationships in the data.

Hyperparameter Tuning: each model has specific hyperparameters that significantly impact its performance. These parameters were optimized using techniques like grid search and random search:

- **Grid Search**: Systematically exploring a predefined set of hyperparameters.
- **Random Search**: Randomly sampling hyperparameters from a defined distribution.

Methodology

Cross-Validation: To ensure the robustness of the models, k-fold cross-validation was employed. This involved dividing the training data into k subsets, training the model on k-1 subsets, and validating it on the remaining subset. This process was repeated k times, with each subset used exactly once for validation:

- **Stratified K-Fold**: A variant of k-fold cross-validation that ensures each fold is representative of the class distribution.

Model Training: Each algorithm was trained on the balanced and preprocessed training dataset.

Evaluation

To assess the fit and how good each model performed in this **anomaly detection** study, we will evaluate it by using the following metrics:

Precision: to maximize the positive (failure) predictions that were actually failures (TP) and minimize false calls (FP) which are non-failure values predicted as failure:

$$Precision = \frac{TP}{TP + FP}$$

F1 Score: harmonic mean of Precision and Recall:

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

Accuracy: measures the overall correctness of the model:

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

We will mostly focus on the F1 and Precision scores.

Results and Analysis

Performance Evaluation

For each algorithm we used in this study, we will declare its evaluation scores and state the causes of the results:

Decision Tree: Despite its simplicity, it performed well after simple parameters tuning using Random Search. Here are the results:

Data	Precision	F1	Accuracy
Train	0.9967	0.9961	0.9962
Test	0.8208	0.9004	0.9956

This is because tree classifiers are less sensitive to outliers and the fact that they can capture nonlinear relationships between features and the target.

Random Forest: combined multiple decision trees to improve classification accuracy and reduce overfitting. Here are the results: .

Data	Precision	F1	Accuracy
Train	0.9995	0.9985	0.9985
Test	0.832562	0.9077	0.9959

A small improvement than the Decision Tree.

Naive Bayes: although its accuracy is good, its main f1 and precision performance showed a very bad overfitting even after applying some techniques to improve results like: tuning parameters (variance smoothing) with stratified K-Fold cross validation and using Complement Naive Bayes which is supposed to handle the imbalance in the test data. And this is the best we could achieve:

Data	Precision	F1	Accuracy
Train	0.9153	0.9529	0.9509
Test	0.1804	0.3055	0.9097

Results and Analysis

Two main reasons resulted in the horrible performance: the huge number of outliers that are among our main goals of this case study. One outlier affects the mean and variance of the features. Thus, all likelihood probabilities will be affected. Additionally, the imbalanced test set that we better not balance because we want it to fully reflect real life scenarios.

Support Vector Machine: same as Naive Bayes, it performed badly. The best results we could get after hyperparameters tuning on both the kernel and the degree where the best kernel is linear and best degree is 5:

Data	Precision	F1	Accuracy
Train	0.9780	0.9856	0.9855
Test	0.4613	0.6302	0.9767

Sensitivity of the SVM to class imbalance is one major cause for these poor performance results where it tends to be biased towards the majority class. Another aspect is the fact that SVM is sensitive to outliers that can heavily influence the location and orientation of the decision boundary.

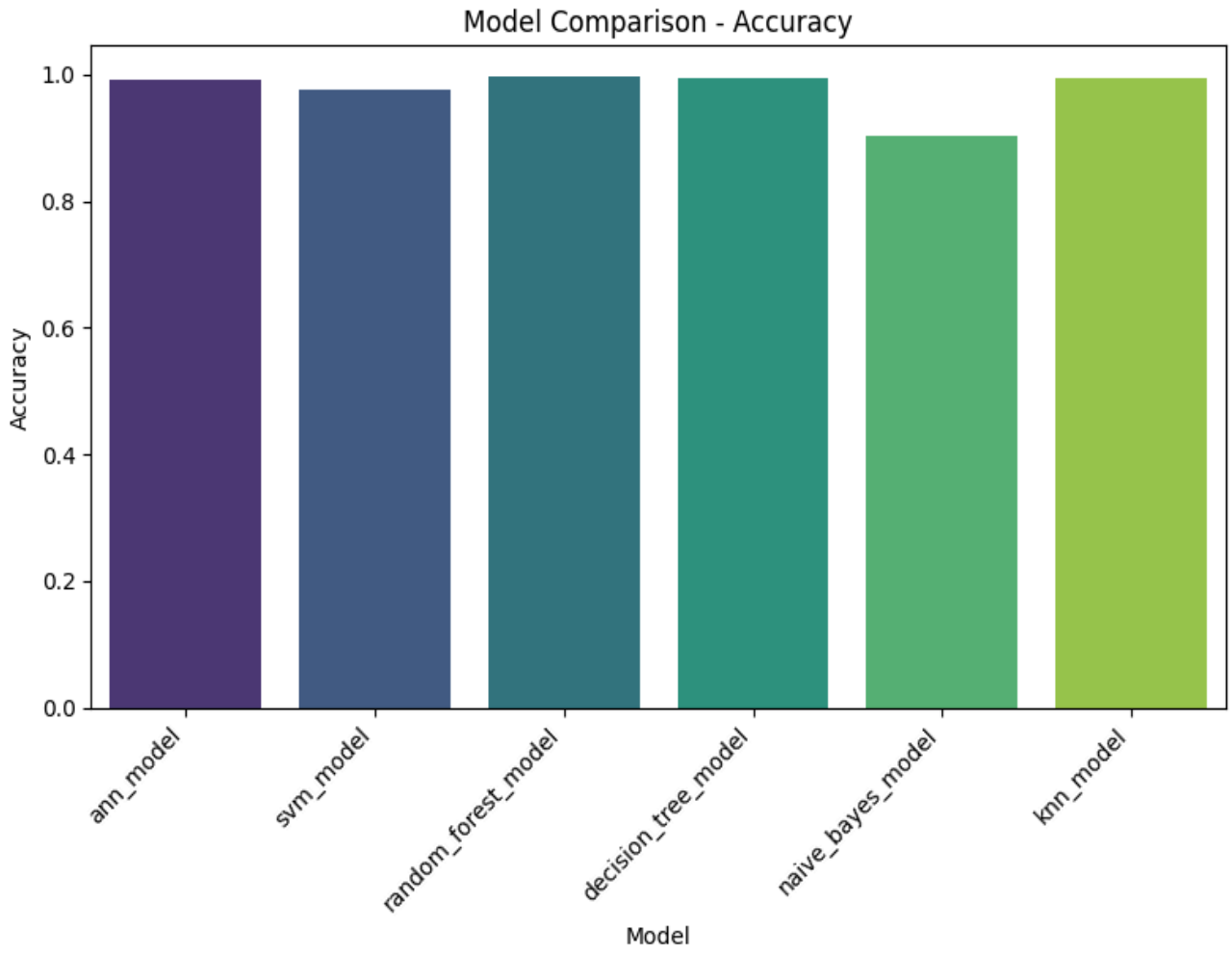
Comparative Analysis

In this section, we delve into a comprehensive analysis of the performance metrics exhibited by all machine learning models that we used in order to detect anomaly cases. Through a detailed examination of Accuracy, Precision, Recall, and F1 Score, we gain valuable insights into the effectiveness of each model in identifying anomalies within the dataset.

Accuracy

The following bar chart comparing model accuracies shows that the Random Forest and Decision Tree models have the highest accuracy, closely followed by the Artificial Neural Network (ANN) and K-Nearest Neighbors (KNN).

Results and Analysis

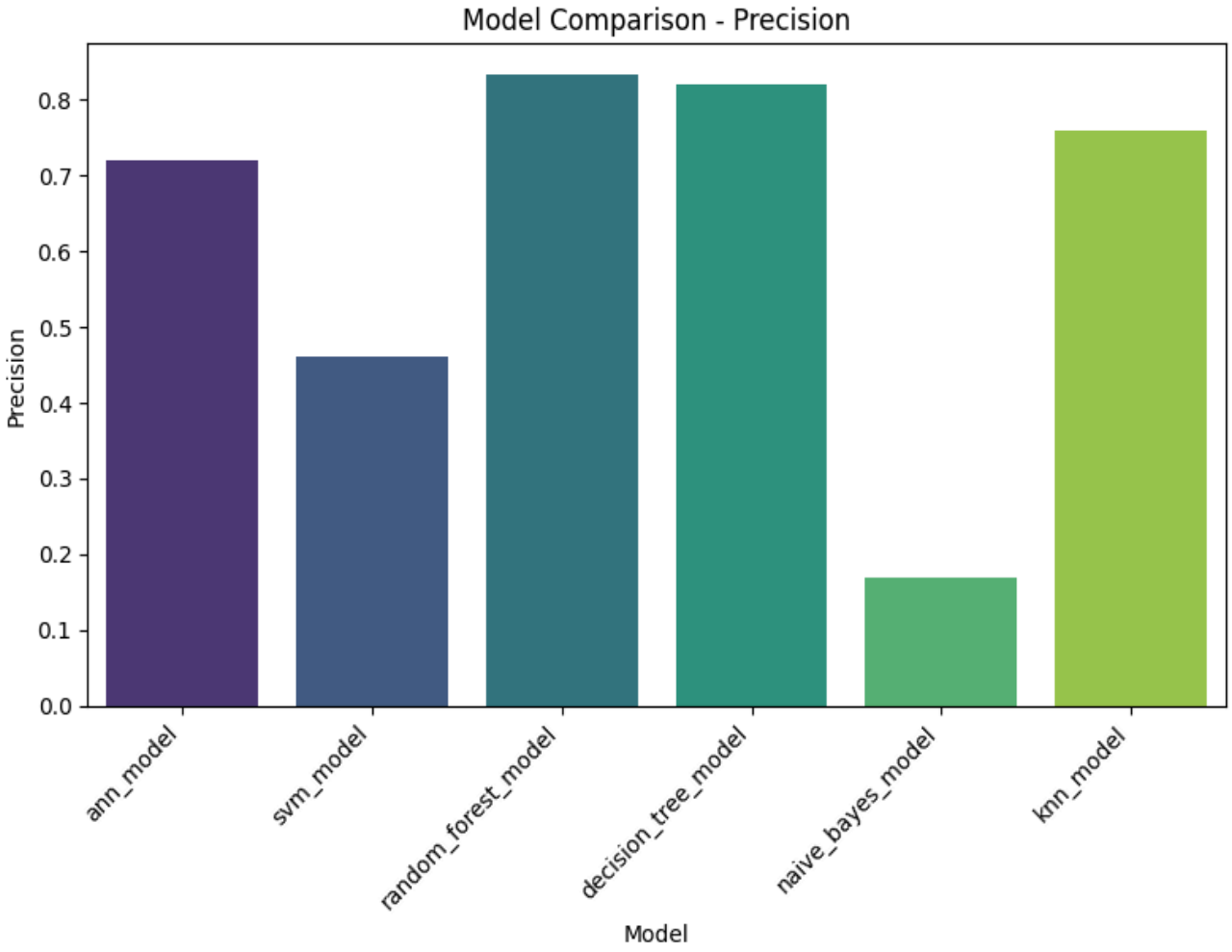


But the accuracy alone is not a good indicator of the best model in the case of anomaly detection, this is because we have too few abnormal cases in our test data, so even a model that assigns all cases to ‘normal’ will have a good accuracy as well.

Precision

The following bar chart compares model precisions and highlights how well each model performs in terms of minimizing false positives, we see that random forest, decision tree and KNN models are the best in preventing false positives, then followed by ANN model with not so satisfying results, this later is followed by SVM and naive bayes models which didn’t perform well.

Results and Analysis

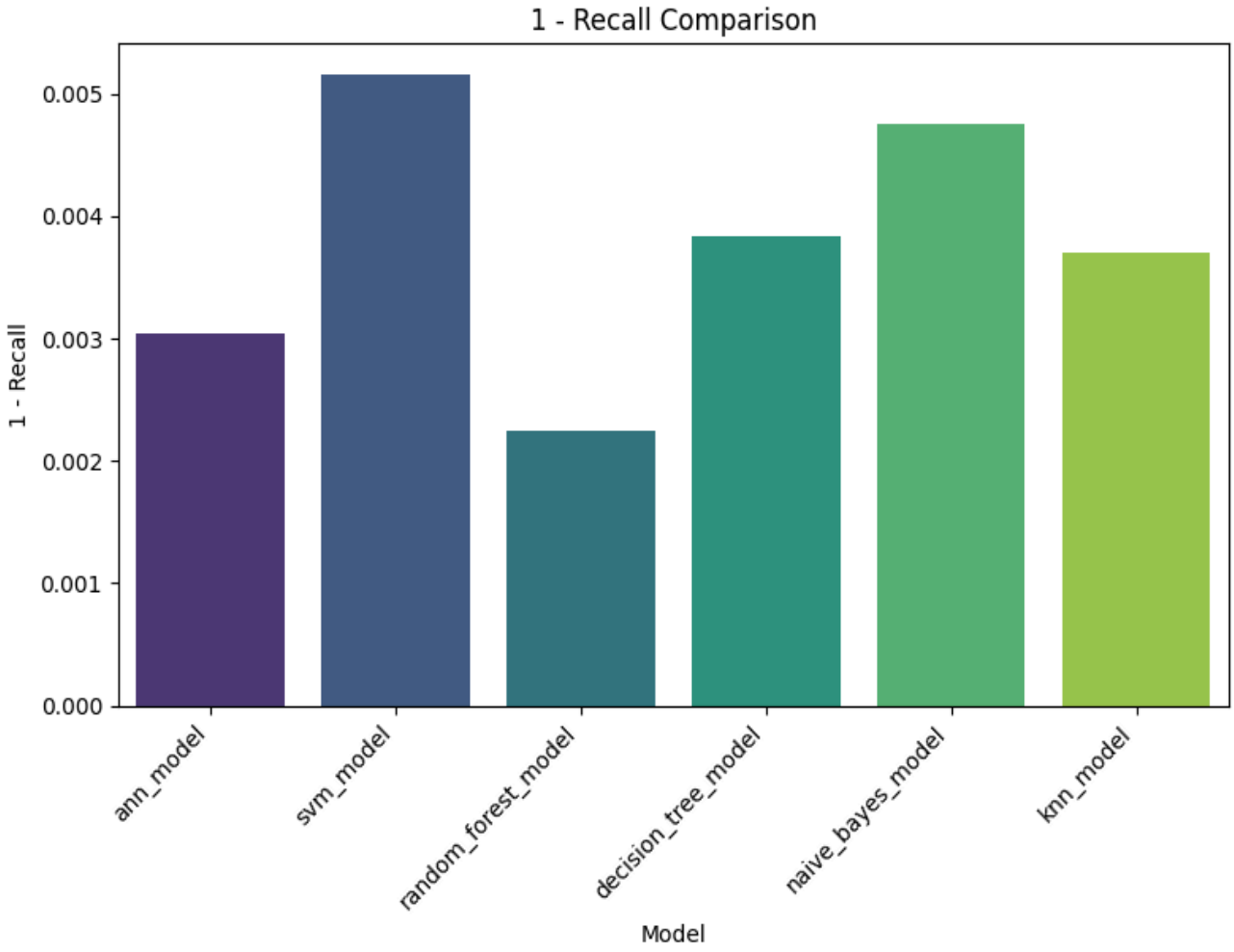


Here we also should state that the precision alone is not enough to decide what model we should choose for our anomaly detection task, false positive cases could be costly but in our case, we should also point our focus towards if our model could detect the majority of abnormal cases or not.

Recall

The following bar chart reflects the ability of each model to identify all actual anomalies. It calculates the value of '1-recall' for each model to identify how many abnormal cases the models didn't detect. We see that random forest and ANN are the best when considering this metric, followed by KNN and decision trees that also did a good job, then followed by naive bayes and SVM models which didn't perform quite well.

Results and Analysis

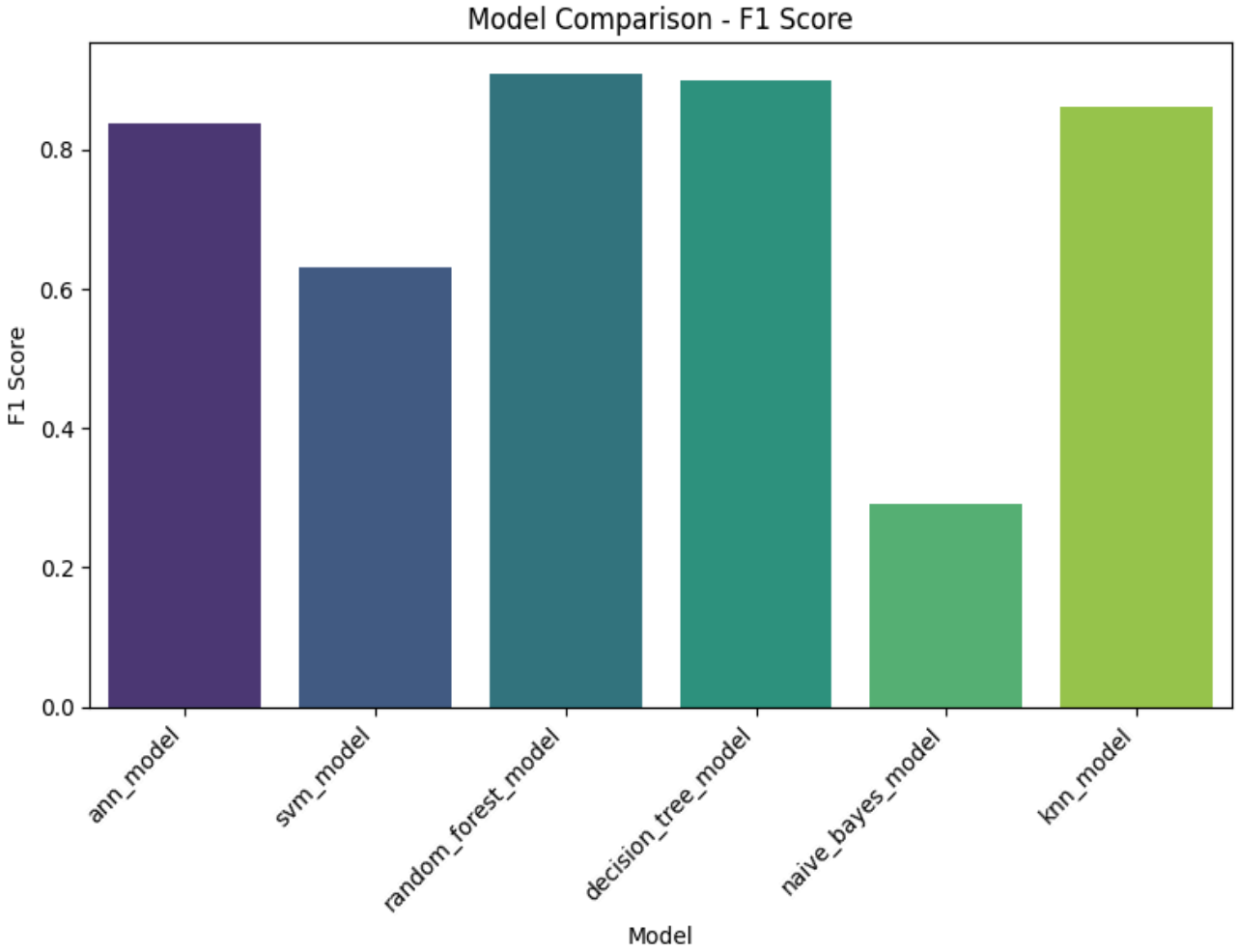


This is the most important metric in the comparison of our models, since false negative cases would be very costly in real life in case of anomaly detection problems, it's crucial to choose a model that can nearly identify all abnormal cases.

F1 Score

The following bar chart displays the F1 Score of each model, random forest and decision tree models has the highest score, indicating the best balance between precision and recall, making them the most reliable models overall, then we have also KNN with a good score, followed by ANN with slightly lower but still good F1 score, then we have SVM and naive bayes with a bad score.

Results and Analysis



The F1 score which represents a balance between the precision and recall metrics is the most reliable measure in indicating the best model in anomaly detection problems, especially when minimizing both false positive and false negative predictions is important.

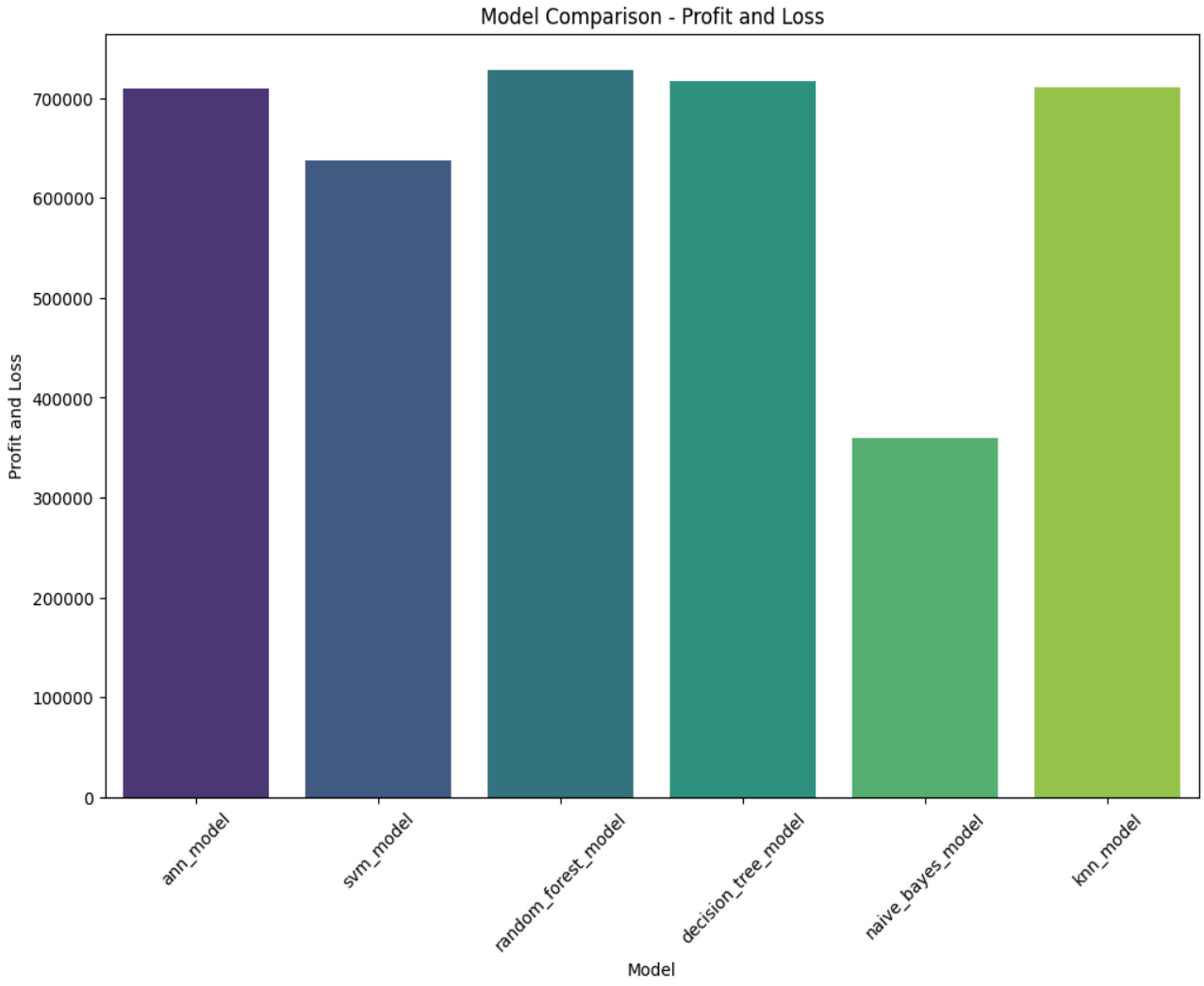
Profit and Loss

To further assess the models' effectiveness in a real-world context, we conducted a profit and loss analysis. We assigned different costs and benefits to the model's predictions to calculate the overall profit or loss.

Results and Analysis

The loss and profit assigned for the following barchart are:

- True positive profit: 100
- False positive loss: 10
- True negative profit: 0
- False negative loss: 700



Discussion

The comparative analysis reveals that the Random Forest model provides the best overall performance for anomaly detection in this study. It achieves the highest scores in accuracy (0.9959), precision (0.8326), and F1 score (0.9077), making it a robust and reliable choice. ANN and Decision Tree models also performed well, especially in terms of recall, ensuring that most anomalies were detected.

To further enhance the performance of these models and other algorithms, the following approaches can be considered:

Feature Engineering: Creating new features or transforming existing ones can help models better capture the underlying patterns in the data, improving their ability to detect anomalies. However, this step needs a lot of domain knowledge.

Ensemble Methods: Combining multiple models, such as using a voting classifier or stacking, can leverage the strengths of different models, potentially leading to improved performance.

Alternative Approaches

Apart from the techniques that we used in our project, there are other unsupervised methods that can be highly effective for anomaly detection:

Isolation Forest: An unsupervised learning algorithm that works by isolating anomalies through random partitioning. It is particularly effective for high-dimensional datasets and does not require labeled data.

Z-Score Separation: A statistical method where anomalies are detected based on their deviation from the mean. Instances with a Z-score above a certain threshold are flagged as anomalies.

Clustering-Based Methods: Algorithms such as DBSCAN or K-means can be used to detect anomalies by identifying points that do not fit well into any cluster.

Conclusion

In conclusion, the comparative analysis of various models for anomaly detection revealed that the Random Forest model provides the best overall performance, combining high accuracy, precision, recall, and F1 score. While other models like ANN and Decision Trees also performed well, the Random Forest's superior balance of all metrics makes it the most suitable choice for our anomaly detection needs.

Future work could explore further optimization of these models, including hyperparameter tuning and feature engineering, to enhance their performance even more. Additionally, investigating ensemble methods that combine multiple models might provide an even more robust solution for anomaly detection tasks. Furthermore, unsupervised methods like Isolation Forest and Z-score separation offer promising alternatives.

References

D. Narjes, V. Bruno, R. Rita, J. Gama. (2023). MetroPT-3 Dataset. UCI Machine Learning Repository. https://doi.org/10.24432/C5VW3R

A. A. Najjar, H. I Ashqar, A. Hasasneh; (2023); Predictive Maintenance of Urban Metro Vehicles: Classification of Air Production Unit Failures Using Machine Learning, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4403258

Veloso, B., Ribeiro, R. P., Gama, J., & Pereira, P. M. (2022). The MetroPT dataset for predictive maintenance. Scientific Data, https://www.nature.com/articles/s41597-022-01877-3

Victoria Shashkina, (2023), How companies can tap into machine learning predictive maintenance, and win big, https://itrexgroup.com/blog/machine-learning-predictive-maintenance/

A. B. Nassif, M. A. Talib, Q. Nasir and F. M. Dakalbab, (2021), Machine Learning for Anomaly Detection: A Systematic Review, https://ieeexplore.ieee.org/document/9439459?denied=

Sami Belkacem, Ranking Social Media New Feed Comparative Analysis, https://github.com/SamBelkacem/Ranking-social-media-news-feed/tree/main

imblearn library documentation, https://imbalanced-learn.org/stable/index.html

Who did what in the project?

Task	Who did the task?
Data Cleaning	Larbi SAIDCHIKH
Feature Engineering	Larbi SAIDCHIKH
Balancing Data	Mohanned KADACHE
Exploratory Data Analysis	Larbi SAIDCHIKH
Feature Importance	Mohanned KADACHE
Evaluation Protocol Definition	Mohanned KADACHE
Decision Tree	Mohanned KADACHE
Random Forest	Larbi SAIDCHIKH
K-Nearest Neighbors	Mohamed Nadjib BENTAYEB
Naive Bayes	Mohanned KADACHE
Support Vector Machines	Mohanned KADACHE
Artificial Neural Networks	Mohamed Nadjib BENTAYEB
Comparative Analysis	Mohamed Nadjib BENTAYEB
Report	Larbi SAIDCHIKH Mohamed Nadjib BENTAYEB Mohanned KADACHE