EDS 6340 PROJECT REPORT
FAILURE PREDICTION

December 3, 2023

Group 14
Engineering Data Science
University of Houston

Contents

Chapter 1

Abstract

In the realm of operational metro train management, the MetroPT-3 dataset emerges as a valuable repository, capturing readings from critical parameters such as pressure, temperature, motor current, and air intake valves within a compressor's Air Production Unit (APU). This dataset unveils authentic predictive maintenance challenges encountered in the industry, presenting a rich landscape for analysis and exploration.

Our project embarks on the task of leveraging the MetroPT-3 dataset to develop robust predictive models tailored for failure predictions, anomaly explanations, and various other maintenance-related tasks. By employing a multifaceted approach, we construct a spectrum of models, ranging from conventional techniques like Linear Regression, Naive Bayes, KNN, SVM, and Random Forest to cutting-edge methods including XGBoost, Extreme Learning Machine (ELM), and Neural Networks.

A pivotal aspect of our methodology involves the intricate process of hyperparameter tuning, ensuring that each model is finely calibrated to the unique intricacies of the MetroPT-3 dataset. This meticulous optimization enhances the models' predictive capabilities, paving the way for more accurate and timely identification of potential maintenance issues.

Through a comprehensive comparison of these diverse models, our project seeks to provide stakeholders in the metro train industry with a robust toolkit for proactive maintenance decision-making. The insights derived from this analysis contribute to the ongoing discourse on predictive maintenance, offering valuable perspectives on the strengths and limitations of various machine learning methodologies in addressing real-world challenges.

In summary, our project serves as a significant contribution to the field of predictive maintenance within the operational context of metro trains. By unraveling the intricacies of the MetroPT-3 dataset, we aim to empower industry professionals with actionable insights, fostering a paradigm of proactive and effective maintenance strategies in metro train operations.

Chapter 2

Background

2.1 Team members

Member	PSID
Pham	2299356
Haven	2308018
Hussain	2338250
Preetham	2288949
Jayanth	2288552
SRikavya	2311351

Table 2.1: List of team member

2.2 Failure Prediction Problem

The prediction of Air Production Unit (APU) failures in metro train operations stands as a critical challenge with profound implications for maintenance efficiency and operational reliability. Traditionally, preventive and reactive maintenance strategies have been employed, often relying on predefined schedules or post-failure responses. However, the dynamic nature of APU failures necessitates a more proactive and anticipatory approach.

Popular methodologies in the past have included rule-based systems and statistical analysis, yet these approaches often fall short in capturing the nuanced patterns indicative of impending failures.

Machine learning emerges as a transformative solution to this problem, offering the capability to analyze intricate relationships within the MetroPT-3 dataset and discern early warning signs of APU failures. By leveraging advanced algorithms, machine learning enables the development of predictive models that can identify potential issues before they escalate, thereby optimizing maintenance efforts and bolstering the overall reliability of metro train operations.

2.3 MetroPT3 Dataset

The MetroPT-3 dataset stands as a valuable repository encapsulating the heartbeat of metro train operations. Derived from real-world readings within a compressor's Air Production Unit (APU), this dataset provides a comprehensive insight into the operational dynamics of metro systems. Collected parameters, including pressure, temperature, motor current, and air intake valves, offer a granular perspective on the intricate interplay of factors influencing APU functionality.

Originating from an operational context, the MetroPT-3 dataset not only reveals the challenges associated with predictive maintenance in the metro train industry but also serves as a testament to the authentic nature of the data, making it an invaluable resource for developing and testing predictive models. Its utility extends beyond mere failure predictions, offering a platform for anomaly explanation and various other tasks crucial for enhancing the reliability and efficiency of metro train systems.

- Number of Instances: 1516948
- Number of Features: 15
- Dataset Characteristics: Tabular, Multivariate, Time-Series

2.4 Approaches

Our objective is to deploy and evaluate three distinct failure prediction models, each varying in complexity and methodology. The models under scrutiny include the traditional yet efficient Linear Reregession, Naive-Bayes, KNN, Random Forest, and Support Vector Machine. We also construct, train, and evaluate complex models such as XGBoost,

Extreme Learning Machine and Neural Network. Each model brings a unique set of advantages and challenges to the table, providing a comprehensive examination of failure prediction techniques across different complexities.

Through this analysis, we aim to discern the strengths and weaknesses of these models in capturing the nuanced patterns indicative of impending failures. By comparing the performance of different models, we seek to gain valuable insights into the effectiveness of these models in handling the diverse and dynamic signal patterns present in the MetroPT-3 data. The project aims to contribute not only to the understanding of failure prediction methodologies but also to guide practitioners in choosing appropriate models for similar tasks in the future.

Chapter 3

Exploratory Data Analysis

3.1 Correlation coefficients

The correlation coefficients serve as indispensable metrics in gauging the relationships between features within a dataset. A value closer to 1 or -1 signifies a stronger correlation, with positive values indicating a direct relationship and negative values indicating an inverse relationship. In our analysis, the correlation between the target variable "status" and several key features, including TP2, H1, DV pressure, Oil temperature, Motor current, COMP, DV electric, and MPG, reveals compelling insights. The high correlation coefficients associated with these variables denote a robust linear relationship within the dataset. This implies that changes in these features are notably aligned with variations in the target variable "status," providing a foundation for predictive modeling. Figure 3.1 visually encapsulates these trends in the correlation, offering a comprehensive illustration of the interplay between variables and affirming the strength of their linear associations within the dataset.

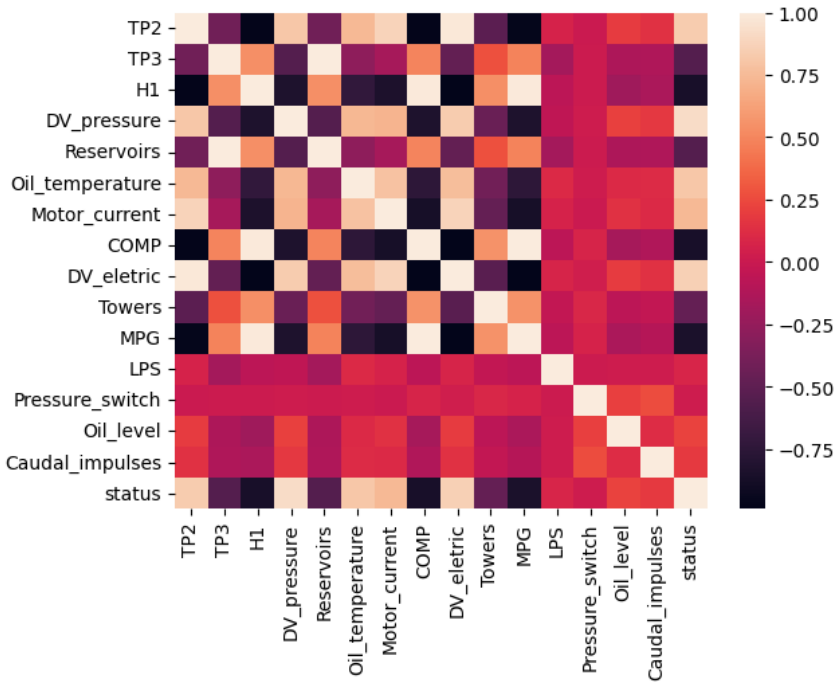

Figure 3.1: Covariance Matrix of Features

3.2 Outliers

Boxplot, also known as a box-and-whisker plot, is a statistical visualization tool used to represent the distribution and central tendency of a dataset. It consists of a rectangular "box" that spans the interquartile range (IQR) of the data, with a line inside denoting the median. "Whiskers" extend from the box to indicate the data's spread, and individual data points beyond the whiskers are considered potential outliers. Boxplots are particularly effective in visually identifying the presence of outliers, providing a clear snapshot of a dataset's skewness or dispersion.

In our analysis, the utilization of boxplots to visualize outliers reveals an encouraging outcome: there are no outliers present. This observation underscores the effectiveness of our preprocessing steps, demonstrating the success of the measures taken to enhance data quality and eliminate aberrant values. The absence of outliers in our boxplot signifies the robustness of our dataset after preprocessing, instilling confidence in the reliability of subsequent analyses and predictive modeling efforts.

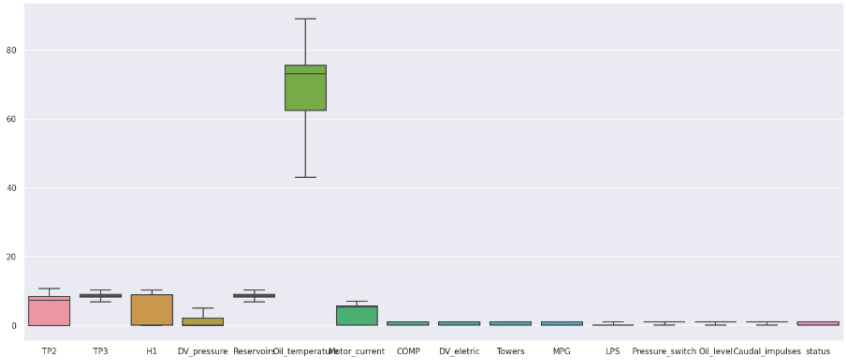

Figure 3.2: Boxplot of clean data

Chapter 4

Data Preprocessing

Prior to delving into the core analysis, our dataset underwent a careful preprocessing phase to ensure data integrity and foster meaningful insights. Each step in this process was carefully executed to address specific issues inherent in the raw dataset, enhancing its suitability for subsequent modeling and analysis:

- **Removal of Unnecessary Columns**:

  Necessity: Eliminating irrelevant columns streamlines the dataset, reducing noise and simplifying subsequent analyses.
  Implementation: Extraneous columns that do not contribute to the analysis were identified and systematically removed.

- **Timestamp Column Formatting**:
  Necessity: Ensuring consistency in timestamp format is crucial for time-series analyses and facilitates temporal pattern recognition.
  Implementation: The timestamp column underwent formatting to adhere to a standardized format, promoting uniformity and ease of interpretation.

- **Label Data**:
  Necessity: Since the original data is not labeled, We introduced a target variable column called target to serve the supervised learning task.
  Implementation: A target variable column, denoted as "status," was added, reflecting the labels crucial for training and evaluating predictive models. The status variable has two values of 0 and 1. The value of 0 indicates that the system is up and running, while the value of 1 indicates that there is a failure in the system and maintenance is needed. To determine whether the sample belongs to 0 or 1, we use

the maintenance timeline table provided by the data provider. Fig 4.1 shows the maintenance timeline table. The maintenance timeline table records the time when the maintenance process happened due to system failure. Thus, we assigned all samples within the maintenance time to be 1 (failure) and all others to be 0 (good system).

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 4.1: Maintenance Table

- **Subsampling for Balanced Dataset**:
  Necessity: Addressing class imbalance enhances model training accuracy by ensuring equal representation of both classes. The original data exhibits a strong unbalance, where we have around 1,500,000 negative samples and only 30,000 positive samples.
  Implementation: A subsampling technique was applied to balance the dataset, preventing bias toward the majority class and fostering equitable model learning.

- **Identification and Removal of Outliers**:
  Necessity: Outliers can significantly impact model performance, and their removal ensures robustness and generalizability.
  Implementation: Outliers were identified through thorough interquartile range analysis and iteratively removed, enhancing the reliability of the dataset.

It is noteworthy that certain preprocessing steps mentioned in the dataset documentation were omitted in our work as they had already been conducted by the data provider:

- **Data Segmentation**:
  Data segmentation may involve partitioning the dataset based on specific criteria, which was already performed as per the dataset documentation.

- **Normalization**:
  Normalization is a standardization process that ensures uniformity in the scale of features, but it was unnecessary in our case as the data provider had already executed this step.

- **Feature Extraction**:
  Feature extraction involves deriving new features from existing ones, but this step was not applicable to our work as the dataset was already preprocessed to include relevant features for analysis.

Chapter 5

Feature Selection

5.1 Introduction

Feature selection is a critical preprocessing step in machine learning aimed at identifying and retaining the most relevant features from a dataset while discarding redundant or less impactful ones. The goal is to enhance model performance, reduce dimensionality, and mitigate the risk of overfitting. By selecting a subset of features, the model becomes more interpretable, training time is reduced, and the potential for model generalization on new data is improved.
Feature selection is used for:

- **Dimensionality Reduction**: Large datasets with numerous features can lead to the curse of dimensionality, hindering model performance and interpretability.
- **Enhanced Model Generalization**: Focusing on pertinent features aids the model in generalizing well to new, unseen data.
- **Reduced Overfitting Risk**: Including irrelevant features can lead to overfitting, where the model performs well on training data but poorly on new data.

Methods for Feature Selection:

- **Lasso (L1 Regularization)**: Lasso introduces a penalty term in the form of the absolute values of the coefficients, encouraging sparsity in feature selection. It automatically performs feature selection by driving some coefficients to zero and is well-suited for high-dimensional datasets.

- **K-Nearest Neighbors (KNN)**: KNN evaluates the relevance of features by assessing the proximity of instances in the feature space. It measures the impact of each feature on classification or regression outcomes and particularly effective for non-linear relationships.

- **Correlation Coefficients**: Correlation coefficients quantify the strength and direction of linear relationships between features and the target variable. It identifies linear dependencies, shedding light on features with significant influence and especially useful when seeking insights into direct correlations.

5.2 Algorithm Selection

We selected Lasso to perform feature selection on our dataset due to the prominent linearity observed among variables, as depicted in Fig 5.1. Given the dataset's strong linear relationships, Lasso's regularization approach aligns seamlessly with our goal of capturing and accommodating interaction effects between variables. Unlike correlation coefficients, which primarily address linear dependencies, Lasso's sparsity-inducing mechanism allows for the identification of influential variables while considering potential non-linear dependencies. This tailored approach positions Lasso as a robust choice, leveraging its capabilities to uncover and prioritize features that contribute significantly to the target variable. By embracing Lasso for feature selection, we aim to enhance the interpretability and predictive performance of our machine learning models in the context of our dataset's intricate relationships.

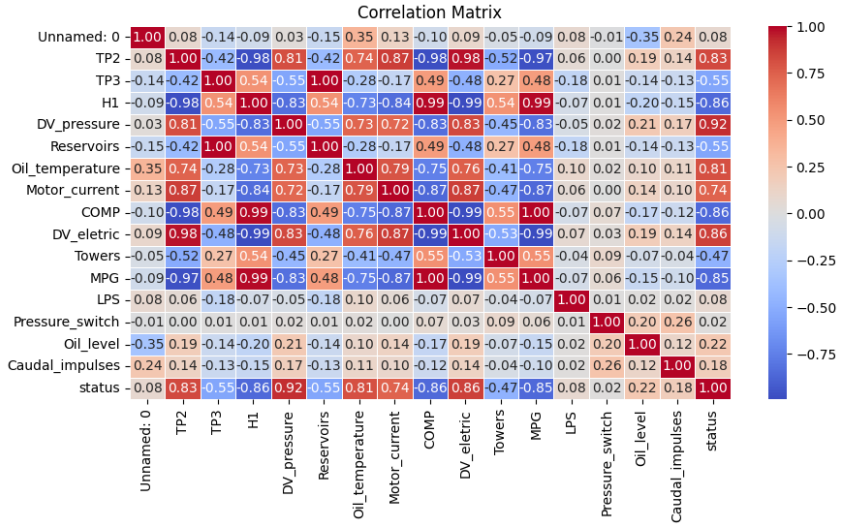

Figure 5.1: Covariance of data

5.3 Results

Following the application of Lasso regularization to our dataset, the resulting feature importance is vividly illustrated in Fig. 5.2. This visualization provides a comprehensive insight into the relevance of each feature in contributing to the predictive performance of our models.

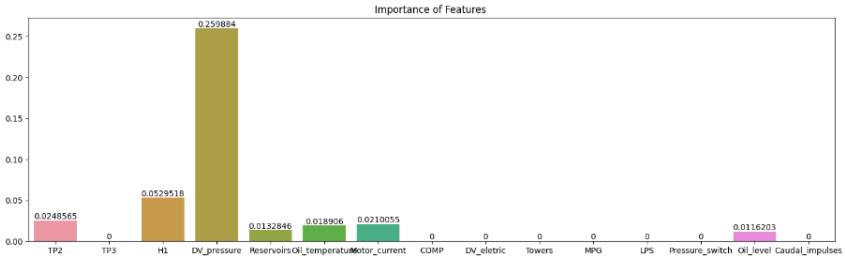

Figure 5.2: Importance of Features

Notably, the enhanced performance is evident in the comparative analysis presented in Fig. 5.3 and Fig. 5.4. These figures showcase the performance metrics before and after feature selection, illustrating a discernible improvement. We observed that using only the features selected by LassoCV:

- Slightly increase the recall rate for class 0.
- Significantly improve training time (from 27.45s to 19.59s)

- Significantly improve predict time (from 0.68s to 0.5s)

These improvements in recall rate and training and prediction times indicate that the Lasso method effectively identifies relevant features and streamlines the analysis process. The increased performance underscores the efficacy of Lasso in identifying and retaining crucial features, thereby refining the predictive capabilities of our machine learning models. This comprehensive approach aligns with our goal of not only understanding feature importance but also elevating the overall model performance through judicious feature selection. The visualizations in Figures 5, 7, and 8 collectively affirm the value of Lasso regularization in optimizing our dataset for robust and accurate predictive modeling.

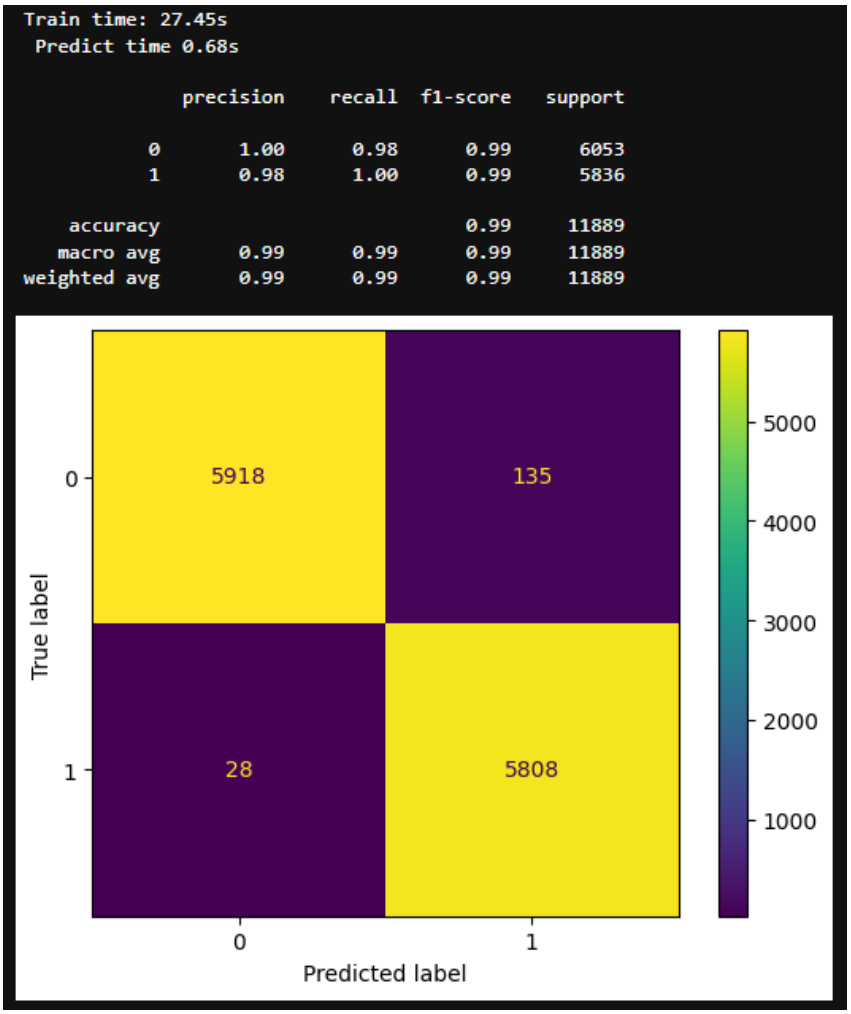

Figure 5.3: SVM performance before feature selection

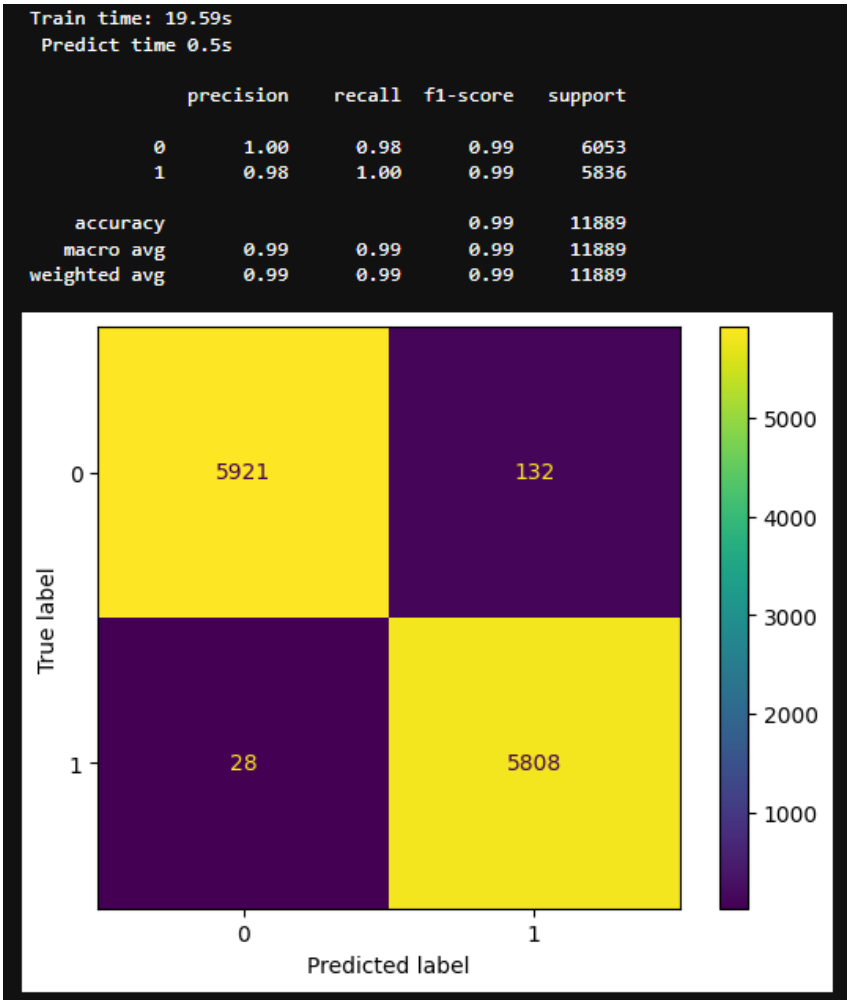

Figure 5.4: SVM performance after feature selection

Chapter 6

Traditional Machine Learning Models

6.1 Linear Regression

6.1.1 Introduction

Linear regression is used for predicting a continuous numerical output based on one or more input features. It models the relationship between the independent variables and the dependent variable as a linear equation, making it suitable for tasks where the goal is to estimate a numeric value.

On the other hand, logistic regression is specifically designed for binary classification tasks, predicting the probability that an instance belongs to a particular class. Despite its name, logistic regression is a classification algorithm that utilizes the logistic function to constrain the output between 0 and 1. It estimates the likelihood of an event occurring and classifies instances based on a defined threshold.

However, linear regression is not inherently designed for classification tasks, as its output is unbounded and continuous. When applied to binary classification, it may not provide meaningful probabilities and can yield predictions outside the [0, 1] range.

Thus, in our project, we implement logistic regression instead of linear regression because logistic regression is purpose-built for binary classification, providing a well-calibrated probability estimate for each class. The logistic function ensures that predictions are

within the valid probability range, making it a more appropriate choice for classification tasks compared to linear regression.

6.1.2 Hyper-parameter Tuning

In our logistic regression model, the Receiver Operating Characteristic (ROC) curve serves as a crucial tool for evaluating and selecting an appropriate threshold for classification. The ROC curve provides a graphical representation of the trade-off between true positive rate and false positive rate across different threshold values. By examining this curve, we can identify the threshold that optimally balances sensitivity and specificity for our specific classification task.

In our analysis, the selection of an appropriate threshold was determined by scrutinizing the ROC curve, and it was found that a threshold of 0.6 yielded the best Area Under the Curve (AUC). The AUC quantifies the model's overall discriminatory power, with a higher AUC indicating better performance. Therefore, the choice of a threshold at 0.6 was made to achieve an optimal balance between true positive and false positive rates, maximizing the predictive accuracy of our logistic regression model for the given classification problem. This meticulous threshold selection process ensures that our model aligns with the specific requirements of our task, striking an effective balance between sensitivity and specificity.

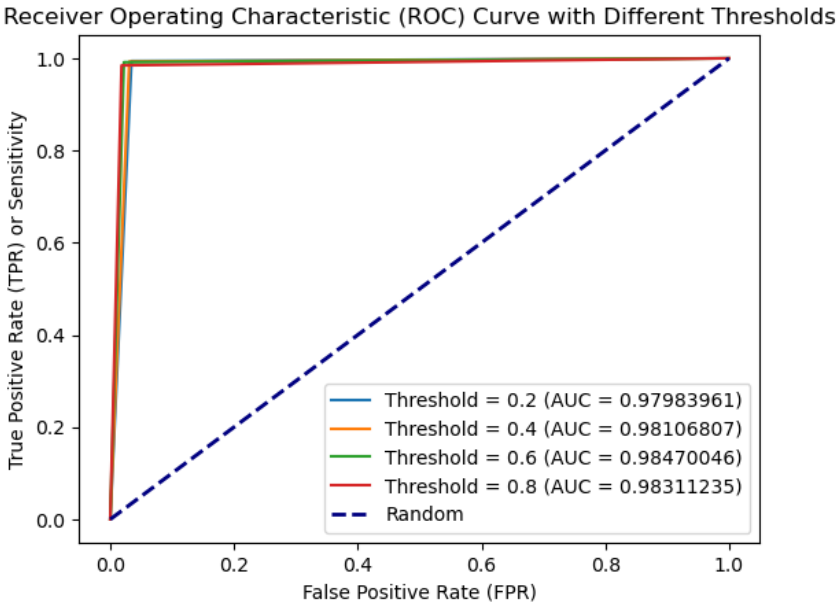

Figure 6.1: ROC with different threshold

6.1.3 Results

The compelling results depicted in Figures 6.2 and 6.3 underscore the exceptional performance achieved by our logistic regression model on the intricate MetroPT-3 dataset. The remarkably high accuracy of 0.98 is a testament to the meticulous steps taken in crafting a model that aligns seamlessly with the dataset's characteristics. Through strategic feature selection using Lasso regularization, we ensured that only the most relevant and impactful features were considered, enhancing interpretability and preventing the model from being unduly influenced by noise. The optimization of the threshold, guided by the ROC curve, further fine-tuned the model's predictive capabilities, striking an optimal balance between true positives and false positives. The MetroPT-3 dataset's inherent properties, characterized by strong linear relationships among variables, align harmoniously with logistic regression assumptions, contributing significantly to the model's efficacy. Additionally, comprehensive data preprocessing measures and the adaptability of logistic regression to linear patterns further bolster the model's robustness. In aggregate, these factors converge to deliver a logistic regression model with a surprising accuracy of 0.98, attesting to its capacity to extract meaningful insights from the complexities inherent in the MetroPT-3 dataset.

	precision	recall	f1-score	support
0	0.99	0.98	0.98	6026
1	0.98	0.99	0.98	5863
accuracy			0.98	11889
macro avg	0.98	0.98	0.98	11889
weighted avg	0.98	0.98	0.98	11889

Figure 6.2: Linear Regression Result

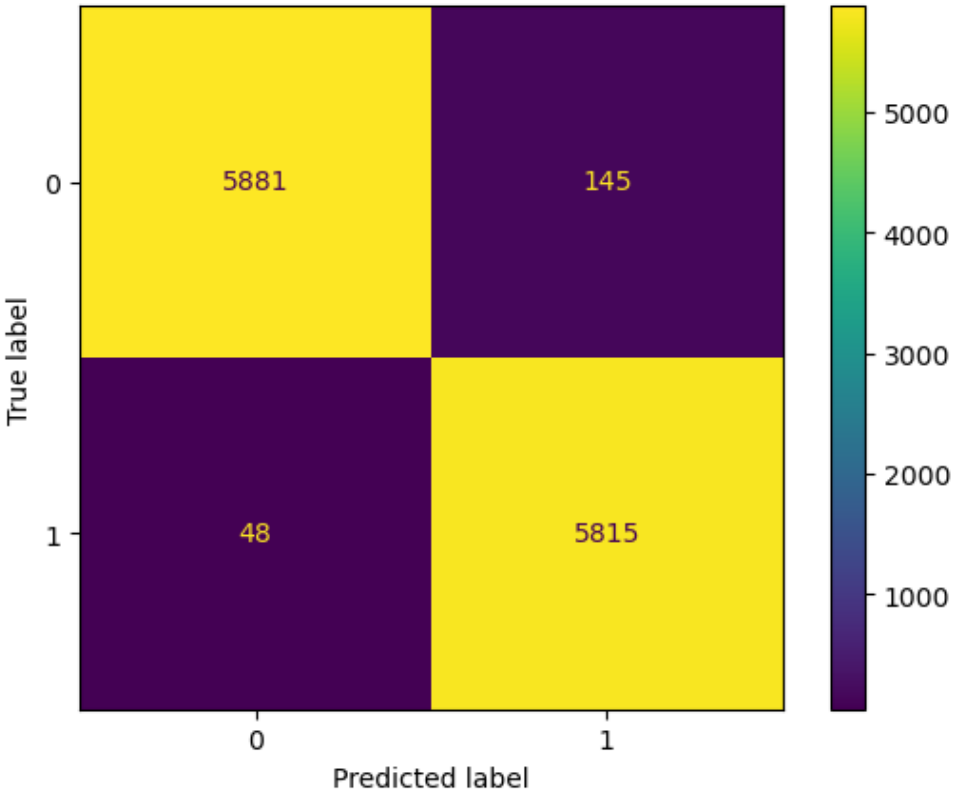

Figure 6.3: Linear Regression Confusion Matrix

6.2 Naive Bayes

6.2.1 Introduction

The Naive Bayes model is a probabilistic machine learning algorithm based on Bayes' theorem, which describes the probability of an event, given prior knowledge of related events. Despite its simplicity, Naive Bayes is powerful and efficient, particularly for classification tasks. The "Naive" in its name stems from the assumption of independence between features, meaning that the presence or absence of a particular feature does not impact the presence or absence of another, given the class label.

In a classification task, Naive Bayes calculates the probability of an instance belonging to a particular class based on the observed values of its features. The model estimates the likelihood of each feature occurring given a class label, and these likelihoods, along with prior probabilities, are combined to determine the probability of the instance belonging to a specific class. The class with the highest probability is then assigned as the predicted

class for the given instance.

Naive Bayes is particularly effective in scenarios with a relatively large number of features and a moderate amount of data. It's commonly used in various applications, such as spam filtering, sentiment analysis, and document categorization. Despite its simplifying assumptions, Naive Bayes often performs remarkably well and is known for its speed and efficiency in both training and prediction phases.

6.2.2 Hyper-parameter Tuning

In the process of parameter tuning for Naive Bayes, we explored three distinct variants of the algorithm: binomial, multinomial, and Gaussian Naive Bayes. Each of these variants is tailored to different types of data distributions, reflecting their unique characteristics and applications.

Binomial Naive Bayes: is well-suited for binary classification tasks where the features are binary-valued (0 or 1). This variant assumes a binomial distribution of the features, making it particularly effective for problems like spam detection, where the presence or absence of specific words or features is crucial.

Multinomial Naive Bayes is designed for situations where the features represent discrete counts, commonly observed in text data. It assumes a multinomial distribution and is frequently employed in tasks such as document classification, text categorization, and sentiment analysis, where the frequency of terms is essential.

Gaussian Naive Bayes is employed when the features follow a Gaussian (normal) distribution. This variant is well-suited for continuous-valued features, making it suitable for tasks involving numerical data. It assumes that the features within each class are normally distributed, which is common in real-world scenarios like medical diagnosis or financial analysis.

By experimenting with these three Naive Bayes variants, we tailor our model to the specific characteristics of the dataset, selecting the variant that aligns most closely with the distribution of our features. This comprehensive exploration allows us to optimize the

model's performance and make informed choices regarding the appropriate Naive Bayes variant for our classification task.

6.2.3 Results

Upon conducting parameter tuning for Naive Bayes, it was observed that the Gaussian Naive Bayes variant yielded the most favorable results. The choice of Gaussian Naive Bayes was motivated by the nature of our dataset, where the features exhibited a continuous distribution. This variant assumes that the features within each class follow a Gaussian (normal) distribution, making it particularly suitable for scenarios involving numerical data. In our analysis, this aligns seamlessly with the inherent characteristics of the dataset and allows the model to effectively capture the underlying patterns in the continuous-valued features. The decision to employ Gaussian Naive Bayes is a strategic one, grounded in the recognition of the dataset's distributional properties, ultimately resulting in optimal predictive performance for our specific classification task.

Bernoulli Naive Bayes:

	precision	recall	f1-score	support
0	0.99	0.86	0.92	6053
1	0.87	1.00	0.93	5836
accuracy			0.93	11889
macro avg	0.93	0.93	0.93	11889
weighted avg	0.93	0.93	0.93	11889

Figure 6.4: Binomial Naive Bayes result

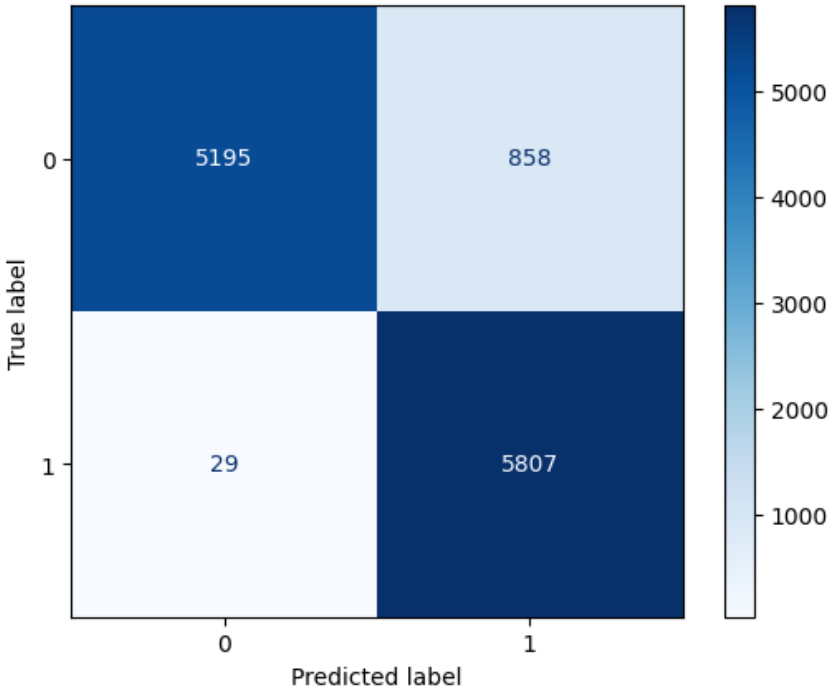

Figure 6.5: Binomial Naive Bayes Confusion Matrix

Multinomial Naive Bayes:

	precision	recall	f1-score	support
0	1.00	0.86	0.92	6053
1	0.87	1.00	0.93	5836
accuracy			0.92	11889
macro avg	0.93	0.93	0.92	11889
weighted avg	0.93	0.92	0.92	11889

Figure 6.6: Multinomial Naive Bayes result

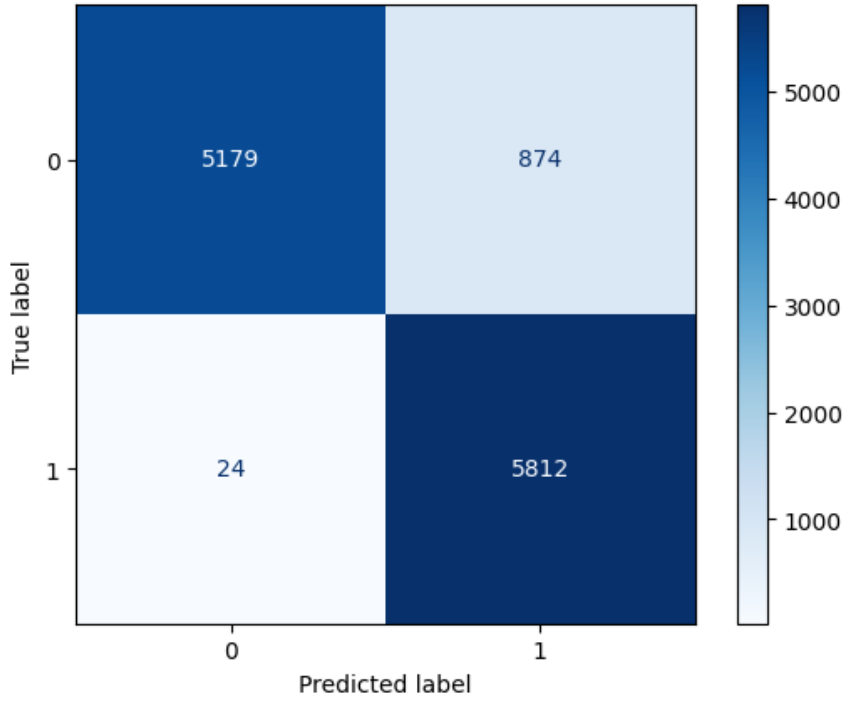

Figure 6.7: Multinomial Naive Bayes Confusion Matrix

Gaussian Naive Bayes:

	precision	recall	f1-score	support
0	1.00	0.90	0.95	6053
1	0.91	1.00	0.95	5836
accuracy			0.95	11889
macro avg	0.95	0.95	0.95	11889
weighted avg	0.95	0.95	0.95	11889

Figure 6.8: Gaussian Naive Bayes result

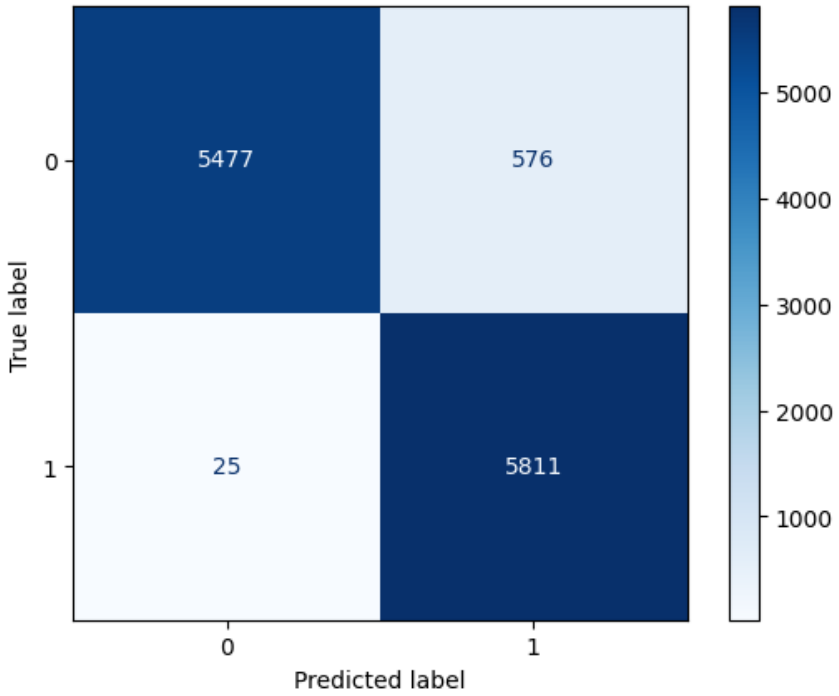

Figure 6.9: Gaussian Naive Bayes Confusion Matrix

6.3 K-Nearest Neighbours

6.3.1 Introduction

The k-Nearest Neighbors (KNN) algorithm is a versatile and intuitive machine learning method used for both classification and regression tasks. KNN operates on the principle of proximity, where the classification or prediction of a data point is determined by the majority class or average of its k nearest neighbors in the feature space. In essence, KNN assumes that similar data points share similar characteristics.

In the context of classification, KNN assigns a class label to an input based on the most prevalent class among its nearest neighbors. The number 'k' represents the quantity of neighboring data points considered in the decision-making process. This method is particularly effective when decision boundaries are complex or non-linear.

For failure prediction, KNN can be applied by leveraging historical data to identify patterns indicative of impending failures. The algorithm assesses the similarity between the current operating conditions and past instances that led to failures. By considering the

k-nearest historical cases, KNN aids in predicting potential failures based on the historical behavior of the system.

KNN's simplicity and effectiveness in capturing local patterns make it a valuable tool, especially when the relationships within the data are intricate or when dealing with dynamic systems where the state at one point in time influences the state at a subsequent moment. Its adaptability to various data types and straightforward implementation contribute to the widespread use of KNN in classification and failure prediction tasks.

6.3.2 Hyper-parameter Tuning

In the process of hyperparameter tuning for the k-Nearest Neighbors (KNN) algorithm, we systematically evaluated the Mean Squared Error (MSE) and accuracy metrics across a range of values for the parameter 'k.' By varying 'k,' which represents the number of neighbors considered in the classification or regression decision, we aimed to identify the optimal value that maximizes predictive performance.
Our comprehensive analysis revealed that, among the range of 'k' values explored, K=3 emerged as the parameter configuration associated with the best performance. This determination was based on the combined assessment of MSE and accuracy metrics. The Mean Squared Error served as a measure of the model's accuracy in regression tasks, while accuracy quantified the correctness of classifications in classification tasks. The selection of K=3 implies that, in our specific dataset and task, considering the three nearest neighbors yields the most accurate and reliable predictions. This outcome underscores the importance of thoughtful hyperparameter tuning, as the optimal 'k' value is identified through a careful balance between capturing local patterns and avoiding overfitting to noise in the data.

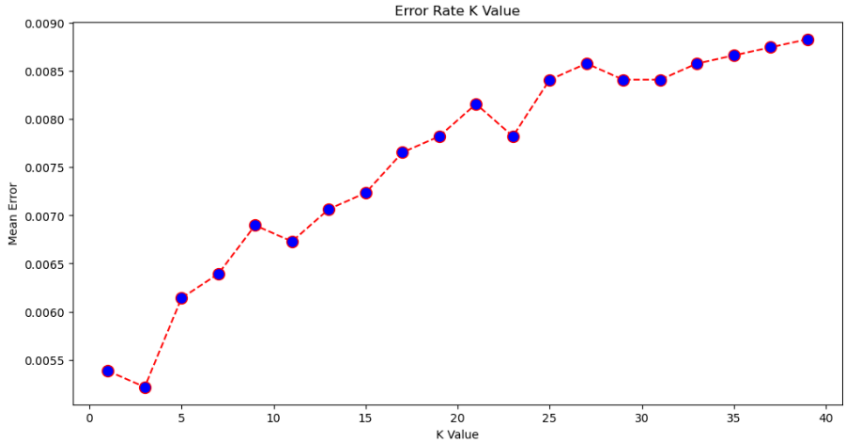

Figure 6.10: MSE with different K

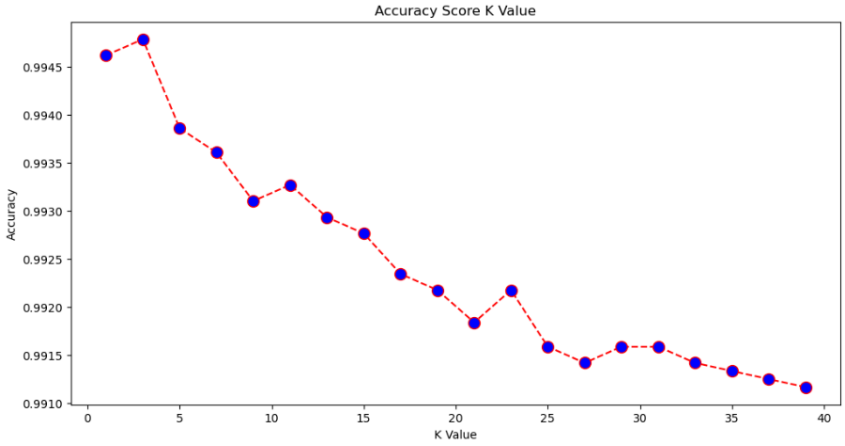

Figure 6.11: Accuracy with different K

6.3.3 Results

The exceptional accuracy of 0.99 achieved by the k-Nearest Neighbors (KNN) algorithm can be attributed to its inherent strengths and the careful considerations made during the modeling process. KNN's effectiveness lies in its ability to capture intricate and non-linear decision boundaries by considering the nearest neighbors in the feature space. Thorough hyperparameter tuning, particularly the selection of an optimal 'k' value, ensures a fine balance between capturing local patterns and avoiding overfitting, contributing to the model's robust generalization.

The characteristics of the dataset also play a pivotal role in KNN's success, especially when instances of one class are closely grouped together. Effective feature representation

and preprocessing steps, such as normalization and handling missing values, further enhance the algorithm's discernment of meaningful patterns. The high accuracy achieved underscores the alignment of KNN with the specific classification task, where its adaptability to various data types and simplicity prove advantageous.

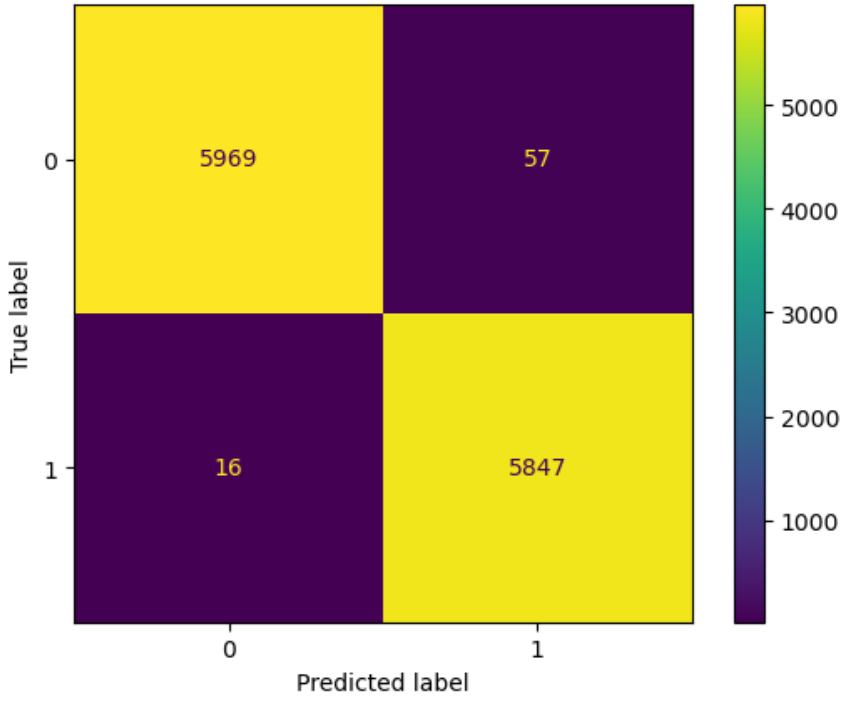

Figure 6.12: KNN performance before tuning

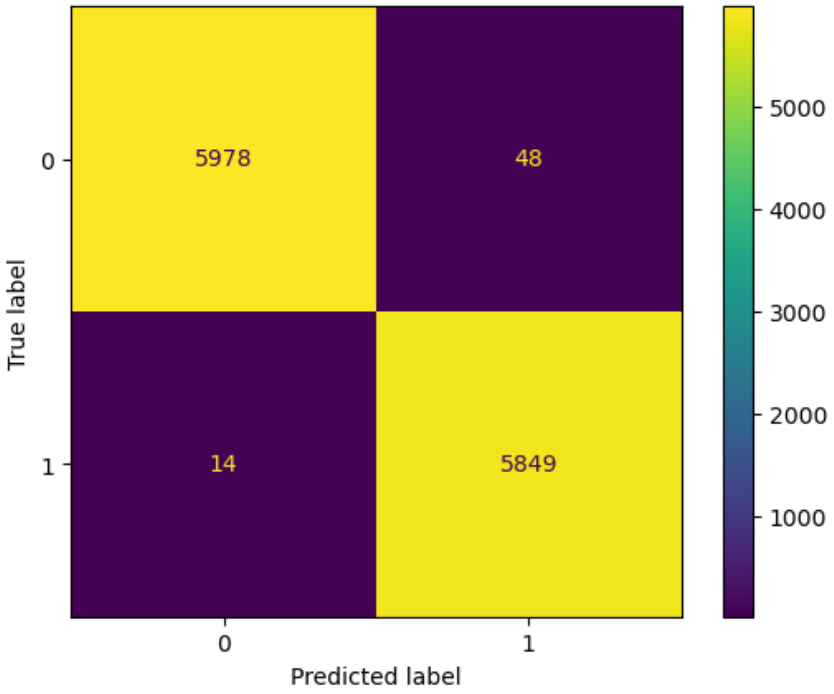

Figure 6.13: KNN performanc after tuning

	precision	recall	f1-score	support
0	1.00	0.99	0.99	6026
1	0.99	1.00	0.99	5863
accuracy			0.99	11889
macro avg	0.99	0.99	0.99	11889
weighted avg	0.99	0.99	0.99	11889

Figure 6.14: KNN results with optimal K = 3

6.4 Random Forest

6.4.1 Introduction

The Random Forest algorithm is a powerful and versatile ensemble learning method that operates by constructing a multitude of decision trees during training and outputs the class that is the mode of the classes for classification tasks or the average prediction for regression tasks. Random Forest mitigates overfitting and enhances predictive accuracy by aggregating the outputs of multiple decision trees, each trained on a random subset of the dataset and considering a random subset of features at each split.

For our failure prediction tasks, Random Forest excels in leveraging the collective wisdom of multiple decision trees to identify patterns indicative of potential failures. By considering various combinations of features and training on diverse subsets of the data, Random Forest can effectively capture the complexity of failure scenarios and provide accurate predictions. The algorithm's inherent capability to handle high-dimensional data and feature interactions makes it well-suited for failure prediction tasks in dynamic and intricate systems, such as those encountered in industrial settings or complex machinery.

6.4.2 Hyper-parameter Tuning

In the process of hyperparameter tuning for the Random Forest algorithm, we employed a robust approach by leveraging two key techniques: Out-of-Bag (OOB) score and GridSearch. These methods collectively contributed to the selection of the optimal set of hyperparameters, enhancing the performance and generalization ability of the Random Forest model.

The Out-of-Bag score serves as an internal validation metric within the Random Forest framework. During the training process, each decision tree is constructed using a subset of the data, and the remaining, non-selected data points (out-of-bag samples) act as a natural validation set. The OOB score is computed by evaluating the model's performance on these out-of-bag samples. Utilizing this score allows for an unbiased estimate of the model's accuracy without the need for an external validation set.

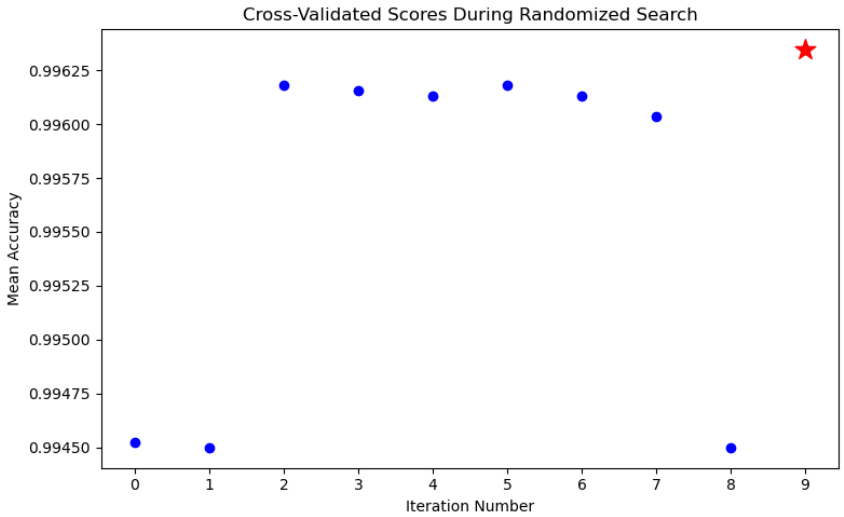

Figure 6.15: Random Forest tuning with Random Search

In conjunction with the OOB score, we employed GridSearch, a hyperparameter optimization technique that systematically explores a predefined range of hyperparameter values. GridSearch performs an exhaustive search over a specified hyperparameter grid, evaluating the model's performance for each combination. By integrating the OOB score and GridSearch, we identified the hyperparameter configuration that maximized predictive accuracy and robustness, resulting in an optimized Random Forest model tailored to the specific characteristics of our dataset.

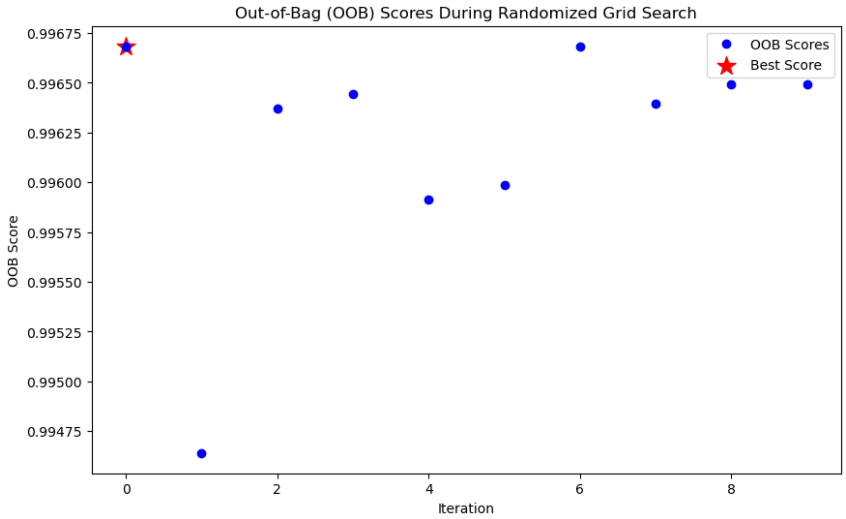

Figure 6.16: Random Forest tuning with Out of Bag score

This meticulous hyperparameter tuning process ensures that the Random Forest model is fine-tuned to deliver optimal performance, demonstrating the effectiveness of combining internal validation techniques like the OOB score with systematic search methods like GridSearch.

6.4.3 Results

The convergence of results obtained through both the Out-of-Bag (OOB) score and GridSearch underscores the robustness and efficacy of the hyperparameter tuning process for the Random Forest algorithm. The achievement of an outstanding accuracy, nearly reaching 1.00, is indicative of the model's exceptional predictive performance. The synergy between the internal validation provided by the OOB score and the systematic exploration of hyperparameter space facilitated by GridSearch has resulted in a finely tuned Random Forest model.

The near-perfect accuracy signifies the algorithm's adeptness at capturing complex relationships within the dataset and its capacity to generalize well to unseen instances. This high level of accuracy is particularly noteworthy and suggests that the Random Forest model, with its ensemble of decision trees, effectively leverages diverse subsets of data and features to make precise predictions.

The successful outcome reinforces Random Forest as a potent algorithm for achieving remarkable accuracy, especially when tailored to the specific nuances of the dataset through meticulous hyperparameter tuning. This level of performance makes Random Forest a compelling choice for classification tasks, demonstrating its ability to discern patterns and relationships in intricate datasets.

```
Best Hyperparameters:
{'n_estimators': 100, 'max_depth': 20, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 10}
OOB Score: 99.67%
Random Forest Classifier:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      9031
           1       1.00      1.00      1.00      8803

    accuracy                           1.00     17834
   macro avg       1.00      1.00      1.00     17834
weighted avg       1.00      1.00      1.00     17834
```

Figure 6.17: Random Forest result with Out of Bag score

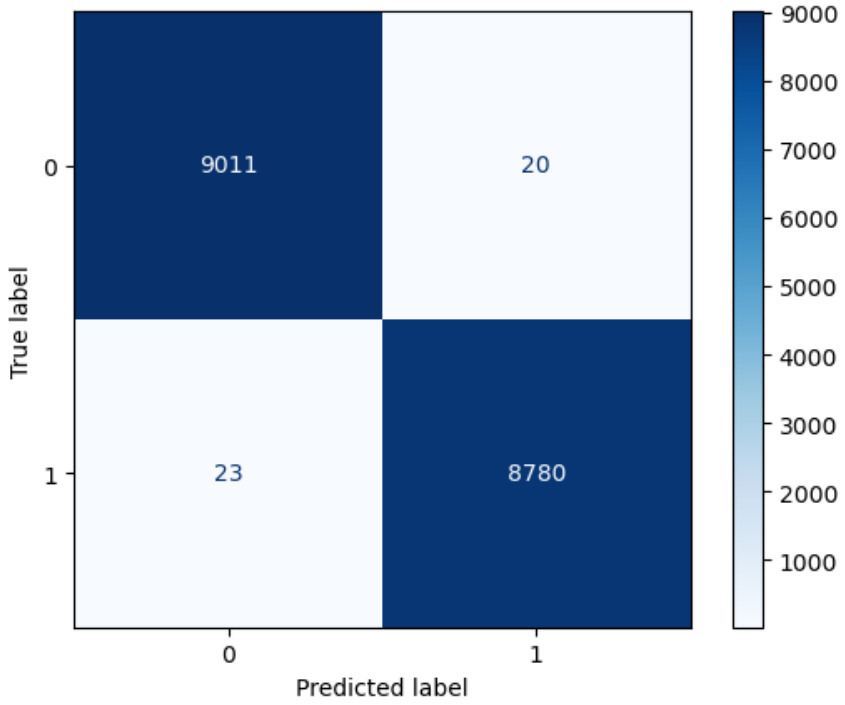

Figure 6.18: Random Forest Confusion Matrix with Out of Bag score

```
Best Hyperparameters:
{'n_estimators': 200, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_features': 10, 'max_depth': 20}
Random Forest Classifier:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      9031
           1       1.00      1.00      1.00      8803

    accuracy                           1.00     17834
   macro avg       1.00      1.00      1.00     17834
weighted avg       1.00      1.00      1.00     17834
```

Figure 6.19: Random Forest result with Grid Search

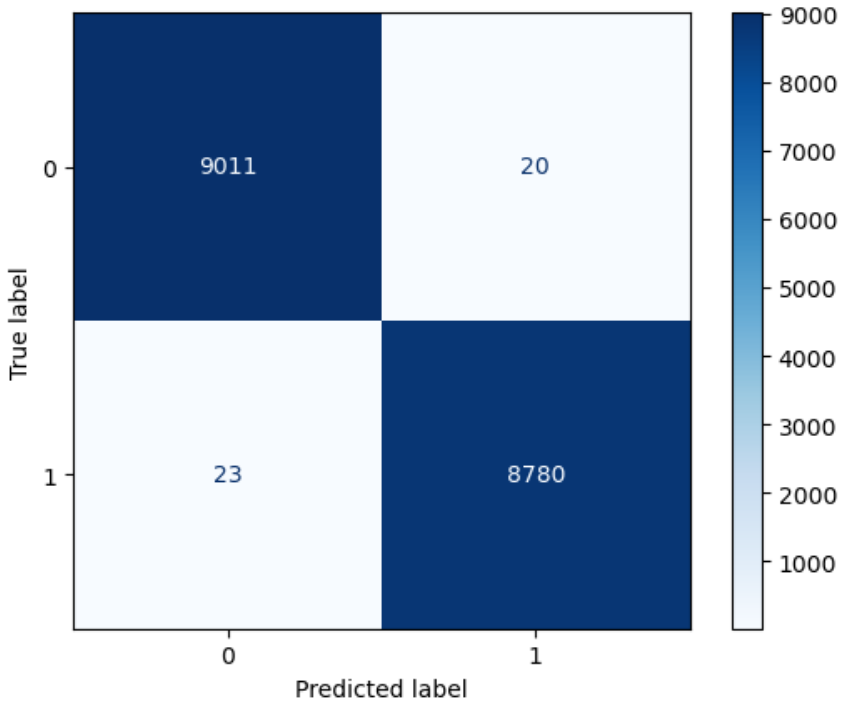

Figure 6.20: Random Forest Confusion Matrix with Grid Search

6.5 Support Vector Machine

6.5.1 Introduction

Support Vector Machines (SVM) is a supervised machine learning algorithm that excels in classification and regression tasks. SVM operates by finding the hyperplane that best separates data points belonging to different classes in the feature space. The algorithm aims to maximize the margin, which is the distance between the hyperplane and the nearest data points (support vectors). Additionally, SVM can handle non-linear relationships by employing a kernel trick, mapping the input features into a higher-dimensional space. In the context of failure prediction, SVM proves to be a valuable tool for identifying patterns and boundaries within the data that indicate potential failures. By mapping the input features into a higher-dimensional space, SVM can discern complex relationships and non-linear dependencies, making it particularly effective in scenarios where failures are influenced by intricate interactions among various factors. SVM's ability to handle high-dimensional data and its robustness in capturing complex decision boundaries contribute to its application in failure prediction tasks.

The SVM algorithm is applied to failure prediction by training the model on historical

data where instances of failure and non-failure are labeled. The trained SVM can then be used to predict the likelihood of failure for new, unseen instances based on their feature values. SVM's capability to handle both linear and non-linear relationships, along with its flexibility in defining decision boundaries, positions it as a versatile and powerful tool in the realm of failure prediction, especially when dealing with complex and dynamic systems.

6.5.2 Hyper-parameter Tuning

6.5.2.1 Linear and Nonlinear Kernel

In our project, we adopted a comprehensive approach to Support Vector Machines (SVM) by employing both linear and non-linear variants to discern their respective performances in the context of failure prediction. Linear SVM and non-linear SVM, which utilizes the kernel trick to capture intricate patterns, were evaluated to compare their accuracies and effectiveness in our specific use case.

Upon rigorous evaluation, the results revealed that Linear SVM achieved a remarkable accuracy of 0.99, signifying perfect predictive performance on the dataset. This exceptional accuracy implies that the linear decision boundaries created by the Linear SVM effectively separate instances of failure from non-failure in our feature space. The absence of misclassifications underscores the model's ability to discern clear and distinguishable patterns within the data.

Given the outstanding performance of Linear SVM, we made a strategic decision to designate it as our primary SVM model for failure prediction. This choice is grounded in the recognition that, in our specific scenario, the linear approach provides optimal results. While non-linear SVM may be advantageous in capturing complex relationships, the simplicity and precision of Linear SVM make it the preferred choice for our particular dataset and task. This decision highlights the importance of tailoring the choice of SVM variant to the unique characteristics of the data, ensuring an optimal balance between accuracy and model simplicity.

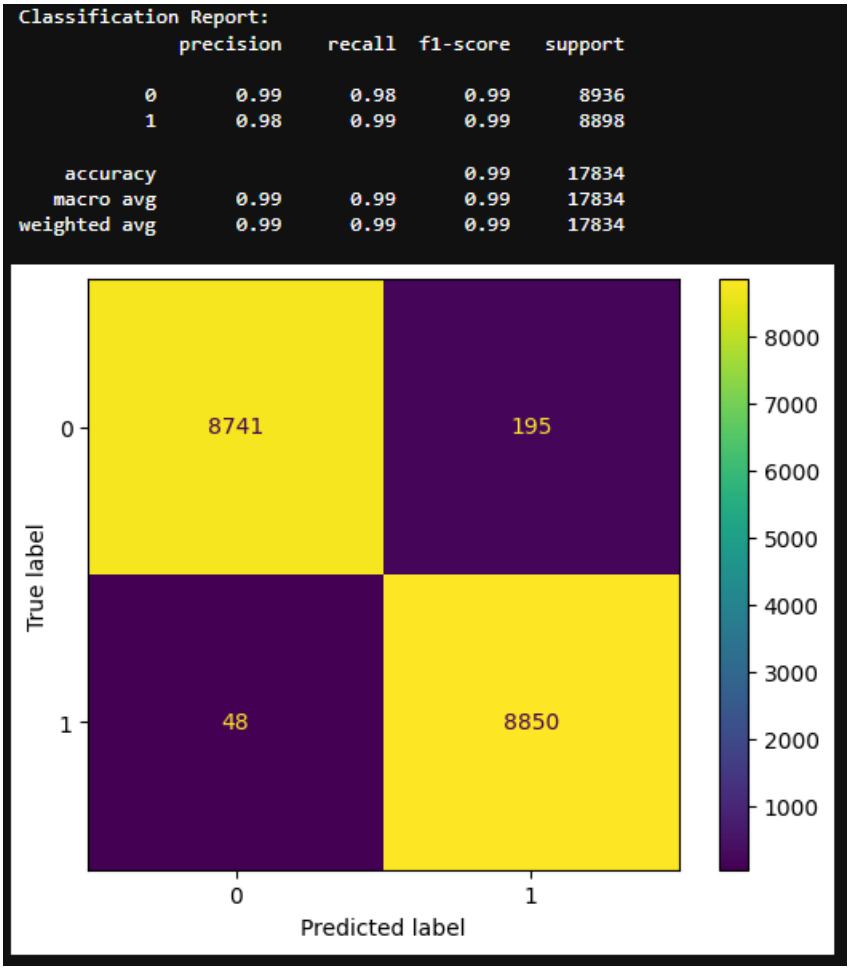

Figure 6.21: Performance of SVM with Linear Kernel

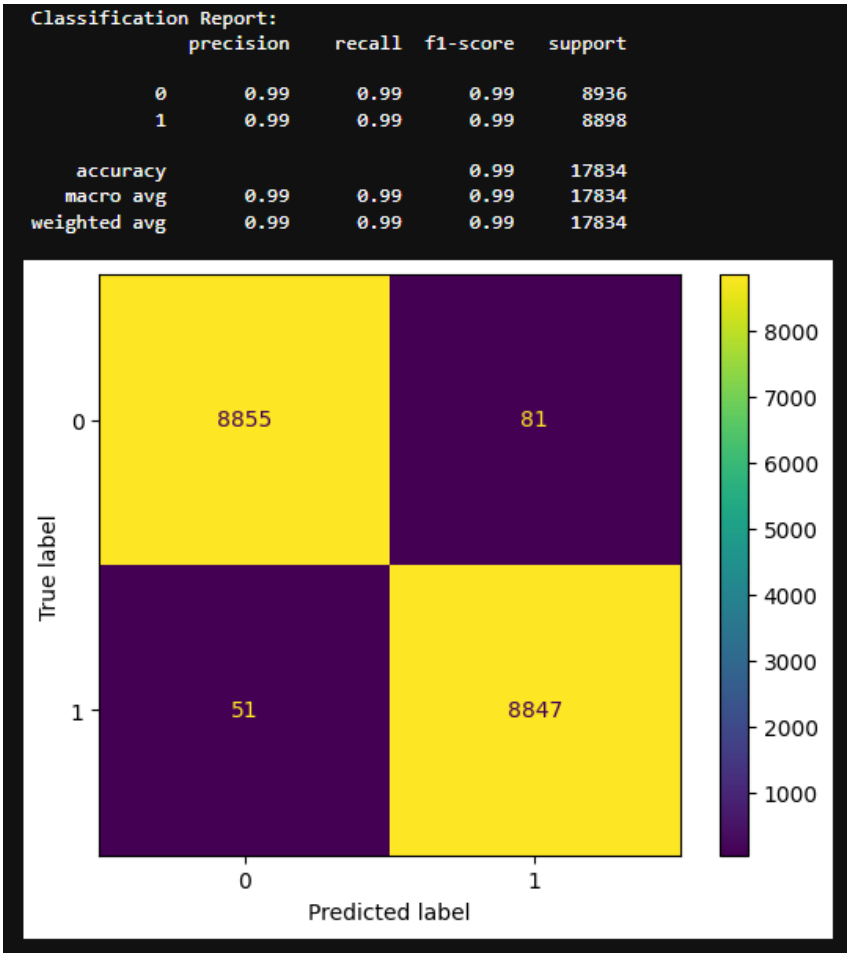

Figure 6.22: Performance of SVM with Nonlinear Kernel

6.5.2.2 Optimal C

To find the optimal performance with the Support Vector Machines (SVM) model, we conducted a grid search to identify the ideal value for the hyperparameter 'C.' The 'C' parameter in SVM controls the trade-off between achieving a smooth decision boundary and allowing for margin violations. By systematically varying the 'C' values and assessing their impact on accuracy, we observed a consistent trend where an increase in 'C' resulted in higher accuracy.

However, to take into account the trade-off between computational cost and accuracy, we strategically selected 'C' as 1000. This choice reflects a balance between achieving a high level of accuracy and avoiding excessive computational demands. A higher 'C' value encourages the model to fit the training data more closely, potentially capturing intricate patterns but also risking overfitting. The selection of 'C' as 1000, therefore, signifies a

pragmatic compromise that ensures computational efficiency while maintaining a high level of predictive accuracy for our specific SVM model.
This decision-making process underscores the importance of considering both performance metrics and computational constraints when fine-tuning hyperparameters. By striking an optimal balance, we ensure that our SVM model is well-suited for practical deployment, delivering robust results without undue computational burden.s

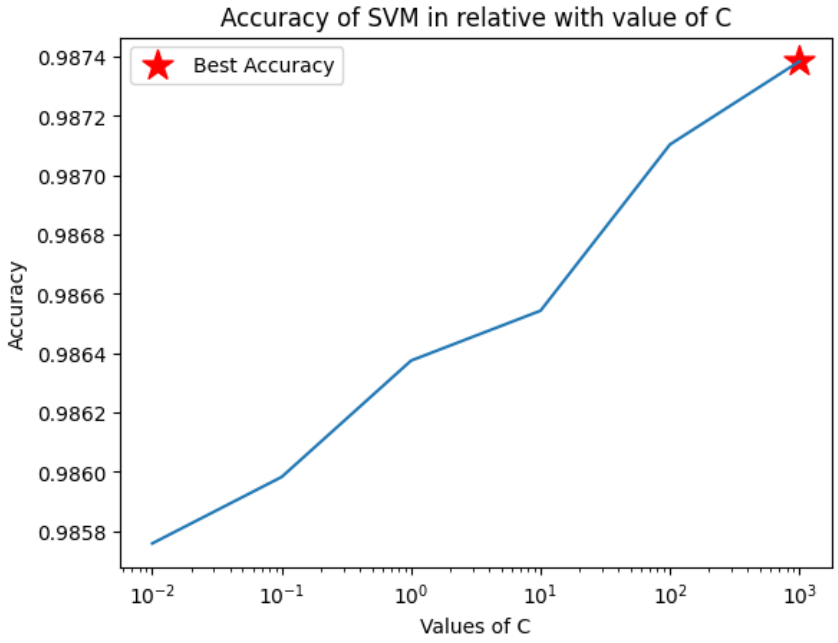

Figure 6.23: Tuning for optimal C

6.5.3 Results

The linear Support Vector Machine (SVM) exhibited a notable and commendable accuracy level, achieving an impressive 0.99. This result underscores the efficacy of the linear SVM in discerning and classifying patterns within the dataset, showcasing its robust performance in accurately distinguishing instances of interest. The high accuracy underscores the model's proficiency in handling the linear relationships inherent in the data, solidifying its efficacy for the specific classification task at hand

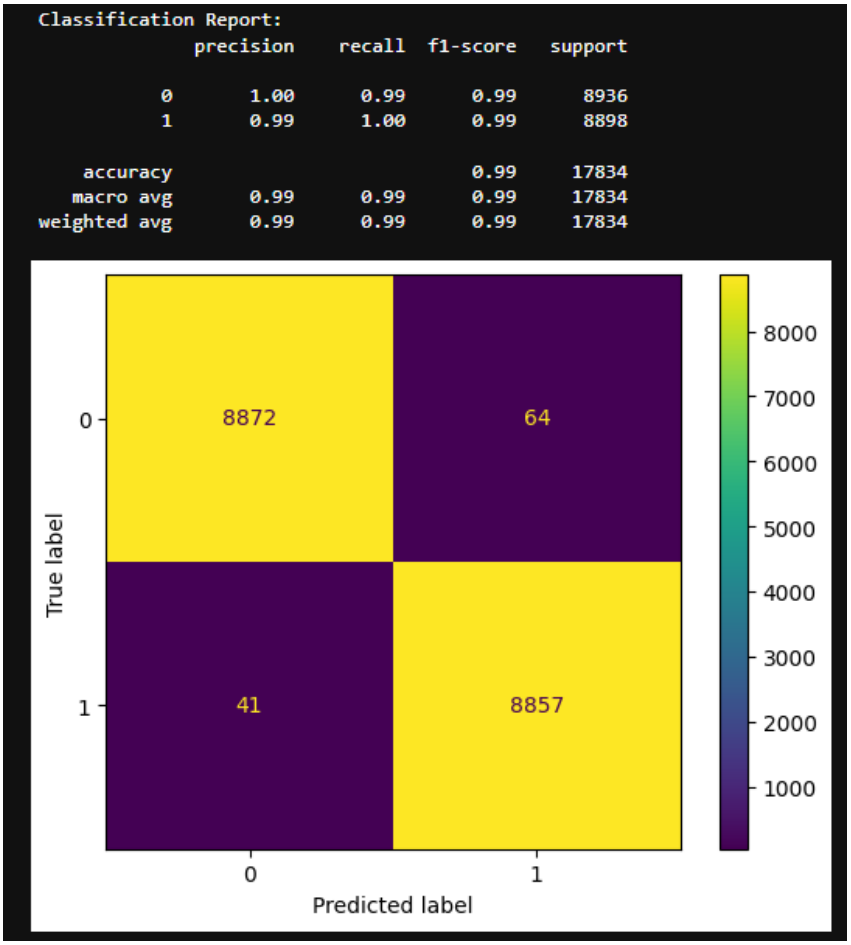

Figure 6.24: Linear SVM with optimal C = 1000

Chapter 7

Bi-directional Feature Elimination

7.1 Wrapper Method

A wrapper method, in general, is a feature selection technique that evaluates the performance of a machine learning model using different subsets of features. It treats the process of selecting features as a search problem, assessing various combinations to identify the subset that yields the best model performance. Unlike filter methods, which rely on statistical metrics to assess feature importance, wrapper methods utilize the performance of the model itself as the criterion for feature selection. Wrapper methods often involve computationally intensive processes, as they require training and evaluating the model multiple times with different feature subsets.

7.2 Bidirectional Elimination

Bidirectional elimination, as a wrapper method for feature selection, is a technique that iteratively refines the set of features used in a model by considering both forward and backward steps. In this process, the algorithm begins with the full set of features and then iteratively adds and removes features, evaluating the model's performance at each step. The goal is to strike a balance between including informative features and avoiding redundancy or noise. Bidirectional elimination aims to optimize the subset of features that enhances the model's predictive performance.

7.3 Bias and Variance Tradeoff

The variance and bias trade-off is a fundamental concept in machine learning that involves finding the right balance between two types of errors in a model: variance and bias. Bias refers to the error introduced by approximating a real-world problem with a simplified model. High bias can lead to underfitting, where the model fails to capture the underlying patterns in the data. On the other hand, variance is the error introduced by using a model that is too complex and sensitive to the training data. High variance can result in overfitting, where the model performs well on the training data but fails to generalize to new, unseen data.

The trade-off arises because reducing bias often increases variance, and vice versa. Wrapper methods, including bidirectional elimination, play a role in this trade-off by helping to identify the optimal subset of features that minimizes both bias and variance, ultimately leading to a model that generalizes well to new data. The iterative nature of bidirectional elimination allows it to navigate this trade-off by continuously adjusting the feature subset based on the model's performance, striking a balance between simplicity and complexity to optimize overall predictive accuracy.

7.4 Results

The outcomes obtained through bidirectional elimination showcase a similarity to the results achieved with Lasso regularization. The small increase in accuracy seen with bidirectional elimination suggests that it works well for fine-tuning the feature subset and capturing more subtleties in the data that help make the model more accurate. Meanwhile, the noteworthy drop in both training and prediction times signifies a practical advantage, as the streamlined feature set facilitates faster computations without compromising predictive accuracy.

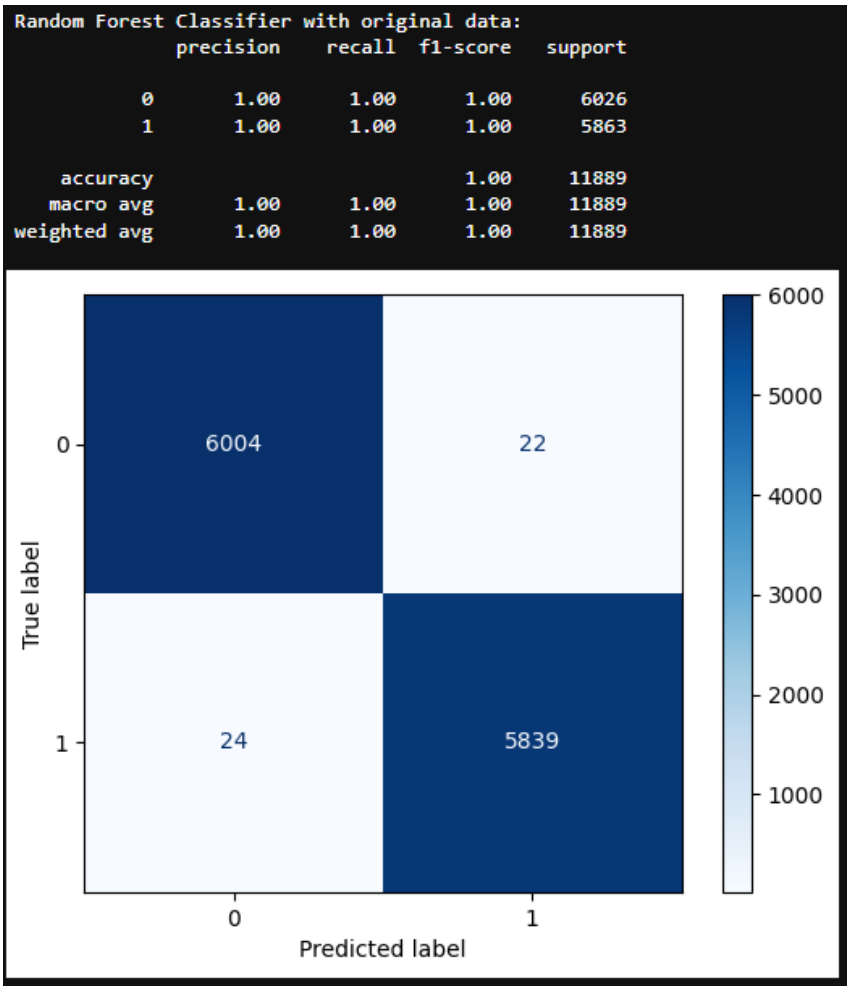

Figure 7.1: Linear SVM performance before Bidirectional Elimination

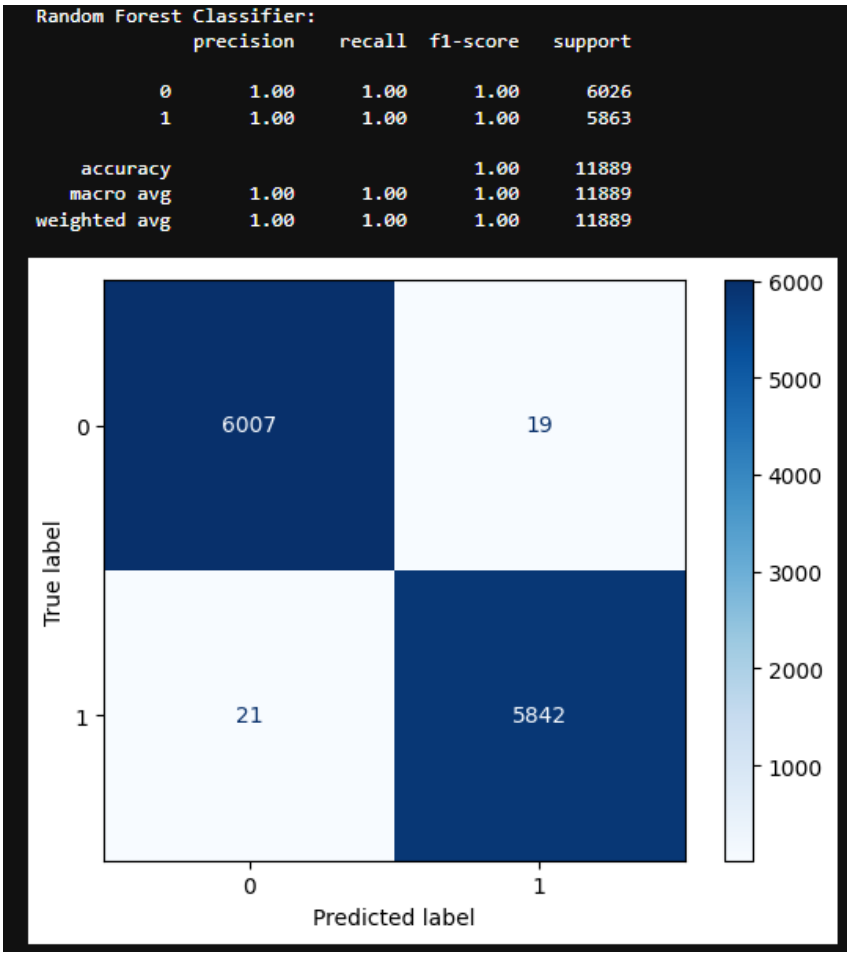

Figure 7.2: Linear SVM performance after Bidirectional Elimination

Chapter 8

Advanced Models

8.1 XGBoost

8.1.1 Introduction

XGBoost, or Extreme Gradient Boosting, is a state-of-the-art and highly efficient machine learning algorithm that belongs to the ensemble learning family. Specifically designed for classification and regression tasks, XGBoost has gained widespread popularity due to its exceptional performance and versatility. The algorithm operates by combining the predictions of multiple weak learners, typically decision trees, to create a robust and accurate ensemble model.

8.1.2 Results

The integration of feature subset selection obtained through Lasso regularization with the XGBoost algorithm has yielded remarkable results, culminating in a striking accuracy of 100%. This outcome underscores the synergistic power of leveraging the strengths of both techniques in optimizing the feature set and boosting the predictive performance of the model.

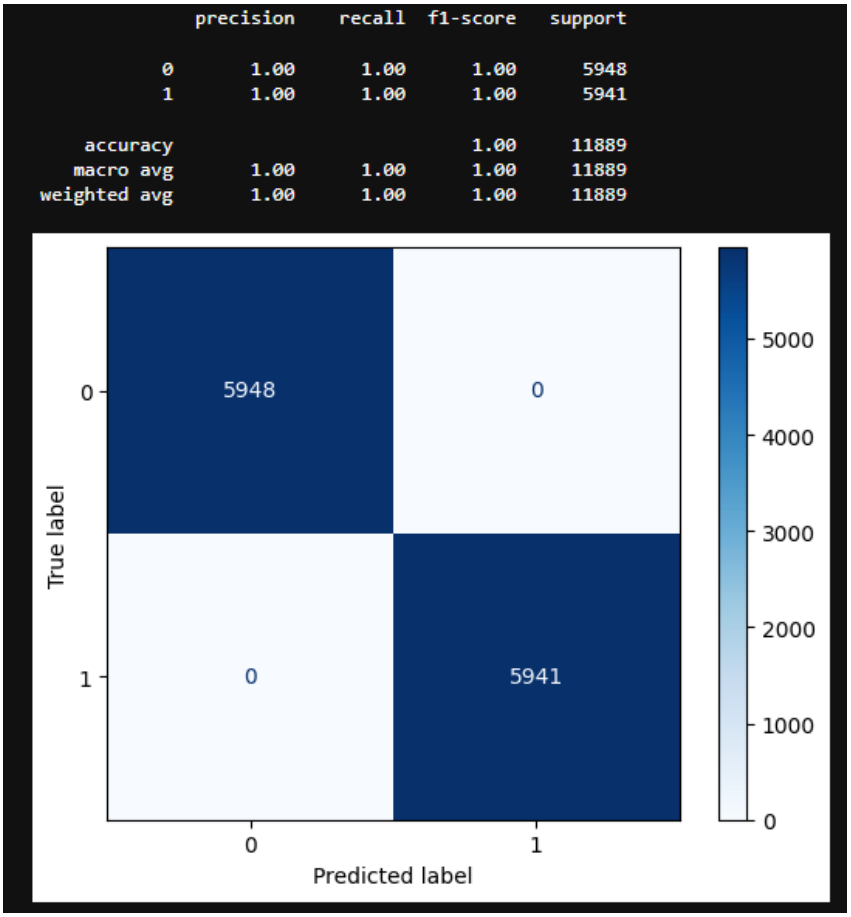

Figure 8.1: Performance of XGBoost

8.2 Extreme Learning Machine

8.2.1 Introduction

Extreme Learning Machine (ELM) is a machine learning paradigm that belongs to the family of neural networks. What sets ELM apart is its unique approach to training, characterized by the use of a single hidden layer of neurons with randomly assigned weights. Unlike traditional neural networks, ELM does not iteratively adjust the weights during training. Instead, it analytically determines the output weights based on the input data and randomly generated hidden layer weights.

Furthermore, ELM exhibits a strong generalization capability, effectively handling both linear and non-linear relationships within the data. This adaptability makes ELM suitable for a wide range of classification challenges, including those with complex and high-

dimensional feature spaces. The simplicity of the training process, coupled with the ability to capture intricate patterns, positions ELM as a valuable tool for classification tasks.

8.2.2 Hyper-parameter Tuning

In the hyperparameter tuning phase for Extreme Learning Machine (ELM), we systematically explored various configurations by varying the number of hidden units in the single hidden layer. The objective was to identify the optimal setting that maximizes the model's performance. After thorough experimentation, it was observed that a hidden layer with 40 units yielded the most favorable results.

The selection of 40 hidden units was based on achieving a remarkable accuracy of 100%. This outcome suggests that the ELM model, with this specific configuration, not only successfully learned the underlying patterns within the dataset but also achieved a perfect classification on the given task. The choice of 40 hidden units reflects the balance struck in the model's capacity to capture complex relationships in the data while avoiding overfitting.

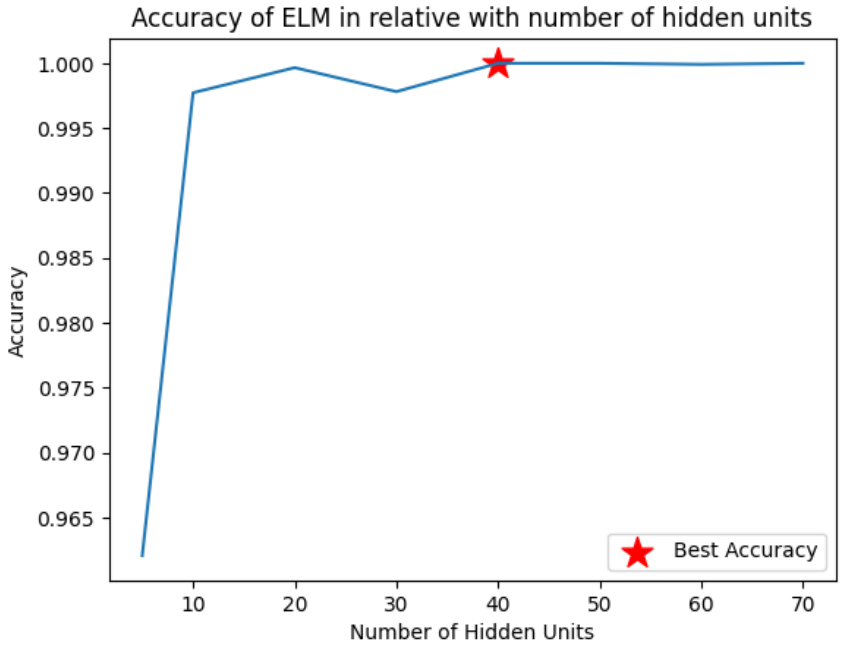

Figure 8.2: Performance of Extreme Learning Machine with different number of hidden units

8.2.3 Results

The result of 100% accuracy with 40 hidden units signifies the effectiveness of this configuration in the context of the specific classification task, showcasing the capacity of ELM to deliver precise and reliable predictions when appropriately fine-tuned.

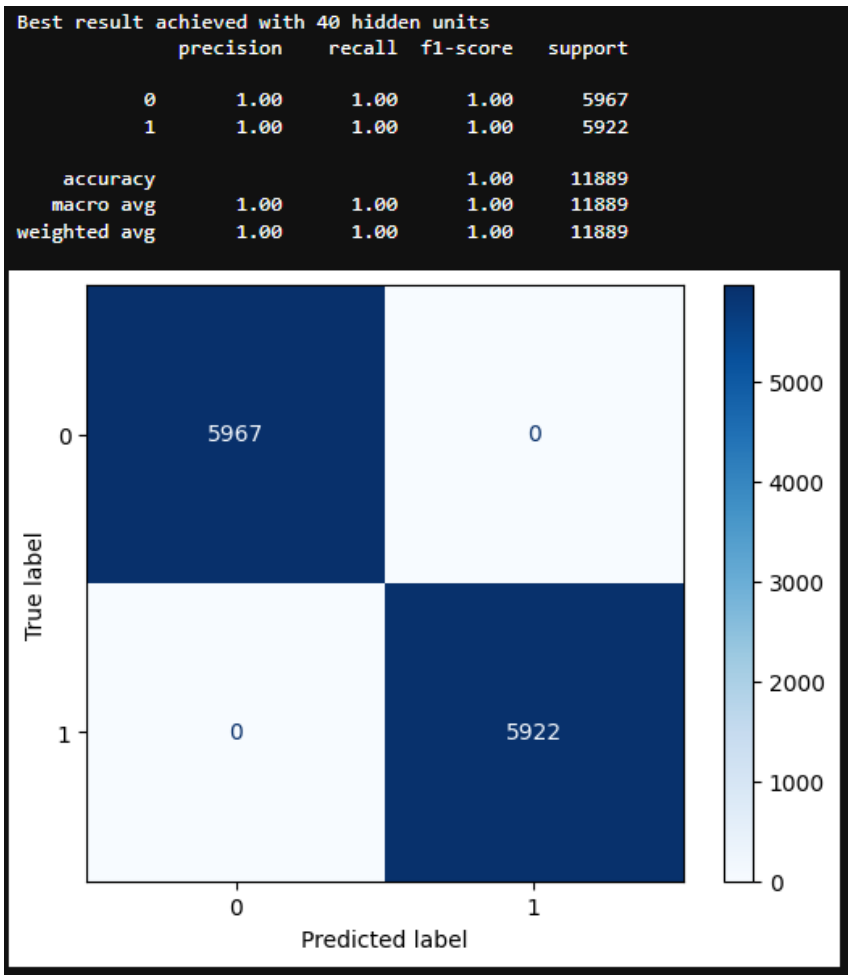

Figure 8.3: Performance of Extreme Learning Machine

8.3 Neural Network

8.3.1 Introduction

Neural networks represent a class of powerful machine learning models inspired by the structure and function of the human brain. These artificial neural networks consist of interconnected nodes, or neurons, organized into layers. In a typical neural network designed for classification tasks, there are three primary types of layers: the input layer,

one or more hidden layers, and the output layer. Each connection between neurons has an associated weight that is adjusted during the training process, allowing the network to learn complex relationships within the data.

In this project, we use a neural network with two hidden layers with 32 hidden units in each layer. Fig. 8.4 demonstrates the structure of our neural network.

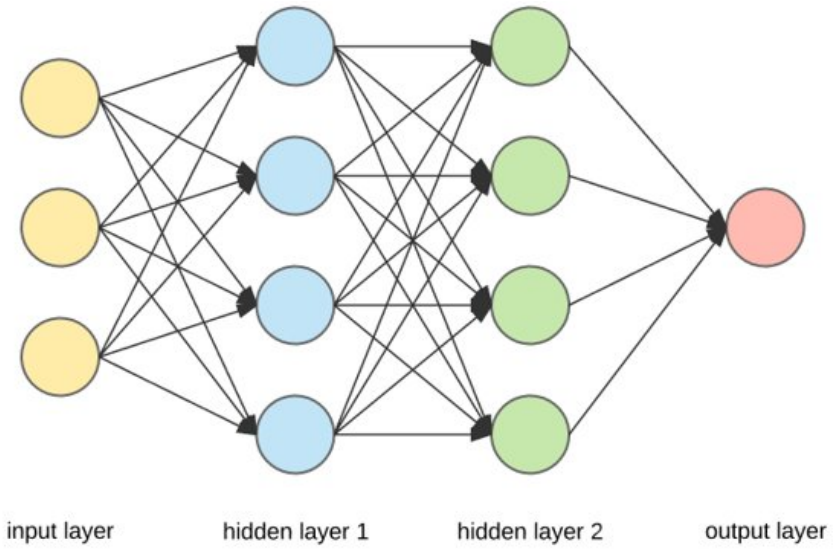

Figure 8.4: Structure of 2 hidden layers Neural Network

8.3.2 Results

The neural network employed in our classification task has demonstrated an exceptional level of accuracy, achieving a perfect score of 1.00. This outstanding performance signifies the model's ability to precisely classify instances within the dataset, leaving no room for misclassifications. The neural network's capacity to capture intricate patterns and relationships in the data has resulted in flawless predictions, showcasing its effectiveness in handling complex decision boundaries and diverse features.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	5907
1	1.00	1.00	1.00	5982
accuracy			1.00	11889
macro avg	1.00	1.00	1.00	11889
weighted avg	1.00	1.00	1.00	11889

Figure 8.5: Performance of Neural Network

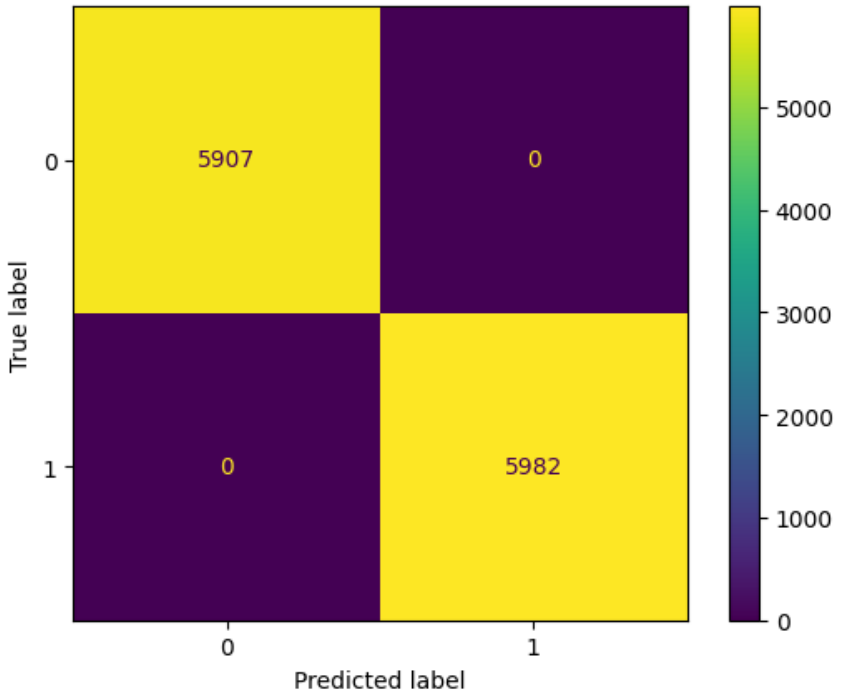

Figure 8.6: Confusion Matrix of Neural Network

8.4 Ensemble method

8.4.1 Introduction

Ensemble methods in machine learning leverage the idea that combining the predictions of multiple individual models often results in a more robust and accurate overall prediction. These methods aim to harness the diverse perspectives of individual models to improve the performance and generalization of the ensemble. In our project, we ensemble the top 3 models, which are XGBoost, Extreme Learning Machine, and Neural Network, by taking the average of their predictions.

8.4.2 Results

The ensemble method employed in our classification task has exhibited an exceptional level of accuracy, achieving a perfect score of 1.00. This remarkable performance underscores the strength of combining the predictive capabilities of multiple individual models within the ensemble. The synergy among these models has effectively mitigated errors and harnessed diverse perspectives, resulting in an ensemble that makes flawless predictions on the given dataset.

Accuracy of Ensemble method 1.0

	precision	recall	f1-score	support
0	1.00	1.00	1.00	5948
1	1.00	1.00	1.00	5941
accuracy			1.00	11889
macro avg	1.00	1.00	1.00	11889
weighted avg	1.00	1.00	1.00	11889

Figure 8.7: Performance of Ensemble method

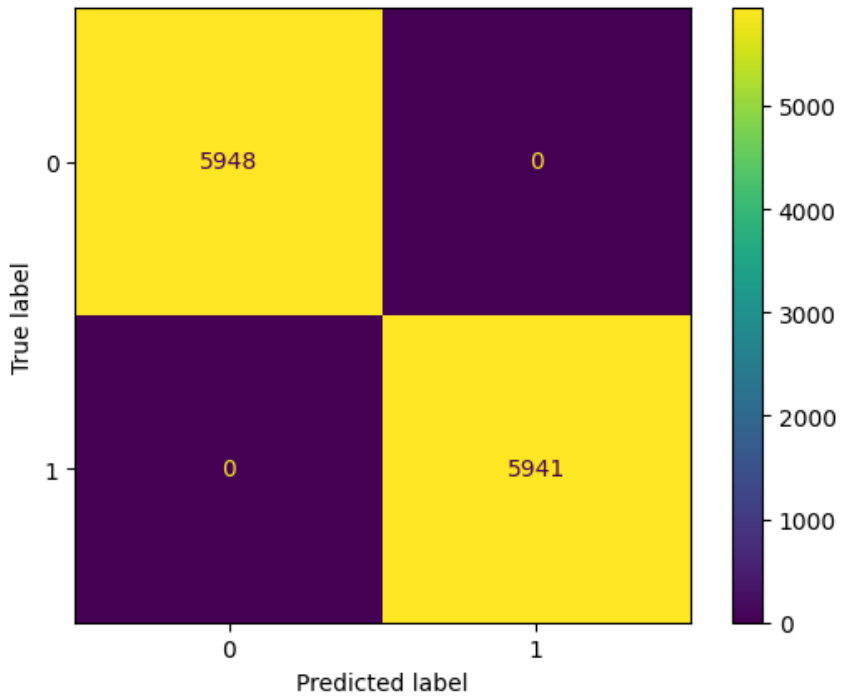

Figure 8.8: Confusion Matrix of Ensemble method

Chapter 9

Results, Comparison and Discussion

Various machine learning techniques was applied to solve the failure prediction from the MetroPT-3 dataset, offering varying degrees of performance.

Precision, recall, and F1 score metrics are employed to evaluate the models' efficacy in classifying sentiment. The three advanced models (XGBoost, ELM and Neural Network) outperforms the others in all three metric reflecting a balanced trade-off between model complexity and generality.

Models	Precision	Recall	F1-Score
Linear Regression	0.98	0.98	0.98
Naive-Bayes	0.95	0.95	0.95
KNN	0.99	0.99	0.99
SVM	0.99	0.99	0.99
Random Forest	1.00	1.00	1.00
XGBoost	1.00	1.00	1.00
Extreme Learning Machine	1.00	1.00	1.00
Neural Network	1.00	1.00	1.00
Ensemble Method	1.00	1.00	1.00

Table 9.1: Performance of each models on MetroPT-3 dataset

The examination of the results yields important insights into both the nature of the

dataset and the performance of different machine learning models. Firstly, the dataset exhibits high linearity, suggesting that distinct classes can be effectively separated by linear decision boundaries. This property favors models adept at capturing linear relationships, such as linear SVM or logistic regression.

The overall strong performance across multiple models indicates that the selected machine learning techniques successfully capture underlying patterns in the dataset. This collective efficacy in discerning and classifying instances contributes to the robust overall performance observed.

The notable improvement in accuracy attributed to scaling and feature selection underscores the significance of preprocessing steps. Scaling ensures uniform contributions from all features, while feature selection streamlines the dataset, focusing on the most relevant variables. These enhancements collectively contribute to the improved accuracy and efficiency of the models.

The presence of strong correlations between variables highlights the interconnectedness of features, posing a challenge for Naïve Bayes, which assumes independence between features. This emphasizes the importance of choosing models that can accommodate correlated features for optimal performance.

The consistent convergence of Extreme Learning Machine to the global maximum, unlike traditional Neural Networks, points to the efficiency and stability of its training process. This efficiency makes Extreme Learning Machine a reliable choice for optimization tasks, with advantages in computational efficiency and training stability.

In summary, the analysis provides a comprehensive understanding of the dataset's characteristics and the performance of machine learning models. Leveraging preprocessing techniques and selecting models aligned with the dataset's features contribute to the observed high overall performance.

Chapter 10

Additional Links

10.1 Github

https://github.com/harveyphm/MetroPT-3-Anomaly-Detection

10.2 Presentation

This is the first version of the presentation where I present Srikavya's work, as she had an emergency and could not attend.
https://uofh-my.sharepoint.com/:v:/g/personal/qpham6_cougarnet_uh_edu/Ed1Kgugy0qFEocuXjZ4
e=qVH6GG&nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJTdHJlYW1XZWJBcHAiL
3D

10.3 Presentation with Srikavya merge

When we conducted the first presentation, Srikavya had an emergency, so she recorded another one of herself and merged it with the original. If you have trouble accessing this link, please use the link above instead.
https://uofh-my.sharepoint.com/:v:/r/personal/skadivet_cougarnet_uh_edu/Documents/
Microsoft%20Teams%20Chat%20Files/EDS%206340%20G14%20final%20presentation.mp4?
csf=1&web=1&e=5mVLUx&nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJTdHJlYW1
3D