AI-based Diagnosis of Combustion Anomalies in Hydrogen Engines

Master's Thesis

for the degree of

Master of Science (M.Sc.)

Medical Engineering

at the Faculty of Engineering of
Friedrich-Alexander-Universität Erlangen-Nürnberg

submitted on 30.06.2024

by **Arun Sai Kumar Reddy Thunga**

Supervisors:
Prof. Dr. Enrique Zuazua
Prof. Dr. Tobias Reichenbach

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Contents

List of Abbreviations

SVM	Support Vector Machine
M^2SVMs	Multi-task Multi-class SVMs
MKVMs	Multi-Class Kernelized Vector Machines
LR	Logistic Regression
NB	Naïve Bayes
KNN	k-Nearest Neighbor
DT	Decision Tree
RF	Random Forest
XGB	Extreme Gradient Boosting
XGBoost	Extreme Gradient Boosting
MLP	Multi-layer Perception
FFN	Feed Forward Neural Network
ADASYN	Adaptive Synthetic Sampling
SMOTE	Synthetic Minority Over-sampling Technique
OvR	One-vs-Rest
OvO	One-vs-One
OVA	One-vs-All
ReLU	Rectified Linear Unit
LightGBM	Light Gradient Boosting Machine
CatBoost	Categorical Boosting
RBF	Radial Basis Function
GLU	Gated Linear Unit
SHAP	SHapley Additive Explanations
LIME	Local Interpretable Model-agnostic Explanations
FC	Fully Connected Layer
BN	Batch Normalization Layer

PI Signal	Pressure Indicated
AI Signal	Torque generated when fuel is burnt
ECU	Engine Control Unit
PCA	Principle Component Analysis
FNR	False Negative Rate
FPR	False Positive Rate
TP	True Positives
FP	False Positives
TN	True Negatives
FN	False Negatives

1 Abstract

Detecting irregularities within the combustion chamber of H2 engines presents a formidable challenge due to the intricacies of the combustion process and the necessity for precise monitoring. A wide range of sensors captures critical metrics such as pressure, mean effective pressure, engine speed, crankshaft rotation, boost pressure, boost temperature, coolant temperature, torque, pre-combustion, and several other parameters. However, traditional data processing methods struggle to differentiate between inconsequential anomalies and genuine signals in this complex environment. To address this challenge, an innovative approach harnesses advanced neural network architectures and machine learning algorithms, aiming to construct models adept at distinguishing anomalies from authentic signals.

Despite a scarcity of data in the automotive sector, especially concerning H2 engines, strategies involving data oversampling have been implemented to address this challenge. This project, conducted at AVL, focuses on establishing a robust pipeline by deploying Multi-class Multi-output classification models. The multi-class model undergoes training and is then employed in a multi-output classifier, implementing diverse strategies to enhance accuracy.

In this thesis, the utilization of Multi-class SVMs, Random Forest Classifier, and XGBoost algorithms with multi-output classifiers are outlined. Additionally, two deep learning methodologies are introduced: a classical feed-forward neural network and TabNet. These strategies include fine-tuning, optimizing feature selection, and employing ensemble methods to achieve improved predictive performance. The model calculates various metrics, such as accuracies and F1 score, and implements confusion matrix scores to detect false positive rates and false negative rates, providing a comprehensive assessment of its classification performance.

This initiative seeks to detect anomalies within the combustion chamber through a supervised learning framework, with the primary goal of precisely pinpointing irregularities crucial for maintaining the engine's reliability. In summary, the integration of Multi-class Multi-output classification models architecture represents a pivotal step forward in the domain of anomaly detection within H2 engine combustion chambers. This research underscores the practical and impactful application of these methodologies, ensuring the early detection and mitigation of potential issues for improved operational safety.

2 Introduction

Within the expansive realm of anomaly detection, the infusion of machine learning (ML) methodologies has sparked a revolutionary transformation. This thesis embarks on an exploration into the intricate landscape of anomaly detection, focusing on the strategic deployment and far-reaching applications of ML techniques across an array of industries and diverse research disciplines.

Anomaly detection, at its core, embodies the pursuit of identifying patterns or occurrences significantly deviating from anticipated norms within datasets or the behaviors of complex systems. This pivotal process facilitates preemptive actions and proactive interventions, alerting user to potential faults, irregularities, or deviations demanding immediate attention and analysis.The identification of anomalies holds a very important role in data analysis across domains such as finance [42], cybersecurity [5], healthcare [8], Automotive industries [58] and more. Anomaly detection also involves in signaling potential fraudulent activities, system failures, or unforeseen events. This comprehensive guide delves into various anomaly detection techniques, encompassing supervised learning methods.

Anomalies, also known as outliers, represent data points or observations markedly deviating from the expected or typical behavior within a dataset. These deviations may arise from factors like data collection errors, rare events, system malfunctions, or deliberate fraudulent actions [24]. Grasping the concept of anomalies is crucial due to its substantial implications across diverse applications. An increasing amount of research is being done in order to detect anomalies in large-scale data, which has a lot of real-world applications [8]. However, many existing anomaly detection techniques fail to retain sufficient accuracy [59].

Machine learning can be employed with significant architecture to detect these anomalies. Moreover, the depth and complexity of machine learning architectures align seamlessly with the multifaceted nature of anomaly detection. These algorithms, ranging from classical methodologies such as Support Vector Machines (SVMs) and k-Nearest Neighbors (k-NN) to the sophisticated neural network architectures within deep learning, possess an innate capacity to capture subtle nuances and intricate patterns within data. This ability empowers these models to detect even the tiniest changes while minimizing false positives caused by irregularities, thus significantly enhancing their efficacy in anomaly detection.

Furthermore, the integration of machine learning reduces the need for manual inspection of extensive datasets, expediting the analysis process with its automation capabilities [7]. This approach contributes substantially to the improved reliability, safety, and maintenance efficiency of critical systems, such as combustion chambers. The automation enabled by deep learning models not only enhances the accuracy of anomaly detection, but also expedites the identification of potential faults, ensuring timely interventions and maintenance activities [3].

In addition, deep learning algorithms also reduces the need for manual inspection of large data sets and facilitates a quicker analysis of them with its automation capabilities. As a result, this approach contributes to improved combustion chamber reliability, safety, and maintenance efficiency.

This comprehensive thesis seeks to meticulously dissect the multifaceted landscape of anomaly detection, shedding light on the diverse applications and methodologies of ML algorithms. By examining the nuanced effectiveness and persistent challenges across industries and research fields, this study endeavors to unravel the profound potential of ML in not just detecting anomalies but also in augmenting the decision-making processes vital to various applications.

Identifying the type of anomalies is crucial as it allows you to choose the right algorithm to detect them. This initial step in the anomaly detection process ensures that the selected machine learning algorithm aligns with the specific characteristics and patterns associated with the identified anomaly types. Moreover, it sets the foundation for a more nuanced exploration of outlier detection methods, connecting seamlessly with the subsequent consideration of the two primary types of outliers: univariate and multivariate.

Additionally, in this study, a novel multi-task learning framework for Anomaly Detection is established, which benefits from acquiring common correlation features among multiple tasks [62]. Subsequently, evaluate the purity of the data to ascertain whether it is class-balanced or class-imbalanced. In the event of class imbalance, employ data oversampling techniques such as SMOTE or ADASYN to address the issue.

The proposed methods in this master's thesis address two primary challenges: the selection of an appropriate oversampling technique and the identification of anomalies using suitable machine learning algorithms. To begin with, data over sampling techniques can be used to generate synthetic numerical data that resemble the original data addressing data deficiency issue which provides an abundance of data for machine learning models. Second, employing machine learning algorithms to identify anomalies that can effectively differentiate between typical and atypical data patterns.These challenges can significantly degrade the performance of machine learning models, leading to biased results. To overcome these challenges, the thesis employs advanced oversampling techniques such as SMOTE or ADASYN to balance the classes, and applies robust machine learning algorithms, including Multi-class SVMs, Random Forest, XGBoost, and deep learning algorithms to accurately detect anomalies.

We will go deeper into the technical complexities of machine learning model development, training, and the assessment in following sections of this thesis, all of which are crucial to the effective application of machine learning in the domain of H2-engine maintenance. We will investigate the intricacies of data preparation, model architecture selection and performance analysis of model. This study is a pioneering attempt at the interface of cutting-edge machine learning technology and automotive industry demands.

2.1 Problem statement

In the field of automobile maintenance, a wide range of sensors are useful for capturing the signals inside the combustion chamber of engines. Addressing the intricacies of anomaly detection in H2 engine combustion chambers using numerical data requires a tailored approach that leverages the power of machine learning (ML). It is difficult for standard data processing techniques to distinguish between good signals and abnormal signals. This cutting-edge technology captures even the minute deviations from the ground truth signal. However, the data related to the H2 Engines, faces a data deficit. The lack of data makes training and generalizing models particularly challenging. Therefore, the over sampling techniques are employed to produce synthetic data.

Later, multi-class, multi-output classification models utilizing machine learning and deep learning algorithms are proposed. These proposed approaches simultaneously analyzes the relationships between various parameters, offering a more comprehensive understanding of anomalies that may be overlooked by traditional, parameter-centric approaches. Both machine learning and deep learning introduces a paradigm shift by allowing the system to understand deep patterns and make nuanced decisions. Implementing a multi-output, multi-class classification system enables the integration of these advanced techniques, providing a more nuanced and accurate approach to anomaly detection.

2.2 Objective

Tackling the complexities of anomaly detection in H2 engine combustion chambers using numerical data necessitates a specialized approach that harnesses machine learning (ML). To overcome challenges like limited data, feature engineering, and multi-output multi-class issues, the proposed method involves developing a robust pipeline. This pipeline generates synthetic training samples through oversampling methods, transform the data and implements a multi-class, multi-output classification model chosen based on three criteria: low model complexity, minimal prediction time, and high accuracy.

One crucial element of this pipeline involves generating synthetic data without directly duplicating the minority class samples for solving class imbalance problem. This approach involves training the multi class algorithm and then given it to multi output classifiers to identify the between anomalies and good signals. Implementing a multi-output multi-class classification system integrates these advanced techniques, offering a more accurate approach to anomaly detection. This not only improves the safety and reliability of H2 engines but also aligns with the industry's shift towards data-driven solutions for complex engineering challenges.

Ultimately, the goal is to enhance the efficiency of anomaly detection procedures during combustion chamber maintenance by combining synthetic data generation with machine learning techniques. This method addresses data constraints and enables the model to accurately identify anomaly configurations observed in real-world situations.

2.3 Structure of the thesis

The upcoming chapters delve into specific aspects of the research topics and objectives. Chapter 3 provides a comprehensive review of the literature and an overview of existing machine learning and deep learning approaches. Chapter 4 discusses the theoretical concepts of oversampling techniques and machine learning models such as multi-class multi-output classifications with SVMs, random forest classifiers, extreme gradient boosting classifiers, and deep learning models like TabNet and feed-forward neural networks, highlighting their strengths and challenges. Chapter 5 details the implementation of these models, including data preprocessing and oversampling methods used. As the experiment progresses, Chapter 6 presents the results and a thorough analysis of the findings. The final chapter offers a detailed conclusion and suggests directions for future research.

This master's thesis seeks to address the challenges of finding the anomalies and improve the accuracy, robustness, and generalization capabilities of machine learning models in this domain by exploring the potential of multi class SVM's, and neural networks. With a comprehensive investigation and empirical evaluation, this research provides useful insights and tools to advance the detection of anomalies analysis.

3 Literature Review

The recent surge in research exploring the synergies between multi-task classification and the challenges of improving accuracy has led to the development and refinement of multi-output classifiers in the field of machine learning. This literature review aims to meticulously navigate the pivotal contributions of researchers who have shaped the intersection of various machine learning models with multi-output classifiers. The foundational birth of multi-class classification models marked a significant milestone in the evolution of machine learning. These models were designed to classify instances into multiple classes, enabling more nuanced and granular predictions than binary classification systems. However, as the complexity of classification tasks grew, researchers recognized the limitations of traditional multi-class classifiers in handling diverse datasets with multiple output variables.

In response to these challenges, the research community began to explore the concept of multi-output classifiers, which extend the capabilities of traditional classifiers to predict multiple output variables simultaneously. These approaches are particularly well-suited for tasks where each instance is associated with multiple labels or target variables, such as multi-label classification or regression problems with multiple dependent variables.

3.1 Multi-task Multi-class SVMs (M^2SVMs)

The research conducted by You Ji et al. [36] presents a novel approach to multi-task learning using Support Vector Machine (SVM). Traditional SVMs are designed for binary classification tasks, but in multi-task learning, there are multiple related tasks, each with its own set of classes. The paper addresses this challenge by extending the SVM framework to handle multiple tasks simultaneously. It introduces a joint optimization problem that incorporates task-specific parameters and regularization to leverage shared information among tasks while accommodating their unique characteristics.

Evgenious et al. [22] proposed a theory which introduces an extension of single-task SVMs to handle multiple related tasks. In this framework, T classifiers are learned, each dedicated to a specific task, along with a common classifier capturing shared information among all tasks. The optimization problem involves minimizing a combination of regularization terms and hinge loss, with lambda controlling the regularization for each task. This approach allows for the modeling of relations between tasks, and kernel methods enable linear and non-linear multi-task learning. While this formulation and experimental results outperform single-task counterparts, it assumes all tasks share the same class labels, limiting its applicability to multi-class problems where correlations between classes are not captured by binary classifiers.

SVMs are transformed in to two main approaches, constructing a multi-class classifier from binary ones (e.g., one-against-all, one-against-one), or considering all classes together. Author also mentioned about the Multi-Class Kernelized Vector Machines

(MKVMs) [17] extend SVMs to multi-class scenarios, aiming to minimize an empirical error on training samples while generalizing well. Results suggest that considering all classes simultaneously requires fewer support vectors for large-scale problems compared to methods using binary classifiers.

The proposed method M^2SVMs aims to learn a model that can perform well on all tasks simultaneously by considering both the shared and task-specific components of the classification models. This is achieved through a combination of task-specific parameters and a regularization term that encourages similarity among tasks. By jointly optimizing these components, the M^2SVMs approach can effectively exploit the similarities and differences among tasks to improve classification performance.

3.2 Multi task learning using random forest

In [63], Qing Wang et al. introduces a new ensemble method for decision tress that integrates principles from multi-task learning. The approach aims to enhance the predictive performance of decision tree ensembles by jointly training them on multiple related tasks. Unlike traditional decision tree ensembles, which are trained independently on each task, Multi-task Forest leverages the relationships and dependencies between tasks to improve predictive accuracy.

In this paper, author discussed various ensemble methods for classifiers, including Bagging and Boosting. Bagging, involves the generation of multiple classifiers by repeatedly sampling subsets of the training data with replacement. This approach aims to create diverse classifiers that collectively contribute to a robust final prediction. In contrast, Boosting operates iteratively by sequentially building a series of weak learners, each focusing on the instances that were misclassified by preceding models. By assigning higher weights to these misclassified instances. Random Forest, combines the random sub-spaces and bagging for decision trees, performing comparably or better than Boosting on noise-free data and proving more robust on noisy data.

In Random Forest technique, it incorporates elements of random subspace and bagging specifically for using decision trees as the base classifier. At every tree node, a random subset of features is selected, and the optimal split is determined from this subset. Additionally, bagging is employed to create the training set of data items for each individual tree. The number of features randomly chosen from the total features at each node is a parameter of this approach. From the experiments conducted, it was observed that Random Forest is comparable to Boosting algorithms and sometimes better on noise-free data.

3.3 Comparison of multi-class classification techniques

Comparison of multi-class classification techniques by Md Salauddin Khan et al,[52] conducted multiple experiments using eight classifiers, including Logistic Regression (LR), Naïve Bayes (NB), k-Nearest Neighbor (KNN), Decision Tree (DT), Random

Forest (RF), Extreme Gradient Boosting (XGB), SVM, and Multi-layer Perception (MLP). The author highlights recent advancements in various classifications, showcasing the application of machine learning algorithms with various features. Despite progress, traditional approaches face inherent limitations. In response, the author introduces a novel machine learning algorithm designed to address these challenges by integrating dimensional and shape features with classifiers. This approach is systematically outlined through stages including data collection, outlier checking, balancing, scaling, and classification using the above mentioned 8 classifiers. Performance evaluation metrics are meticulously computed to assess classifier effectiveness.

XGB represents the natural evolution of decision trees, consolidating multiple trees to collectively determine the final output, thereby reducing reliance on individual trees. Suitable for a range of supervised learning tasks such as regression, classification, and ranking, XGB employs a collection of weak estimators to generate an optimized model. This will be elaborated further in chapter 4.3.3.

Embracing a phase-based approach akin to other boosting techniques, By adopting a tree-generation method that follows gradient descent principles, XGB efficiently constructs robust trees, guiding the target function along the most direct path towards optimization (Zhang et al., 2017)[64].

The utilization of the XGB classifier within the above 7 machine learning algorithm demonstrates superior performance, attributed to its gradient boosting capabilities, effective minimization of loss functions, and mitigation of over-fitting risks [64]. Furthermore, the algorithm efficiently addresses computational complexities by employing an approximate greedy algorithm and utilizing percentiles for data segmentation and threshold determination. These parameters significantly contribute to the classifier's enhanced performance. In the results, all the models perform well. However, XGBoost classifier outperformed when compared to other models.

This study also delves into the classification performance of imbalanced and balanced distributions within the dataset, which holds significant implications for both data science and real world domain. Utilizing the Adaptive Synthetic Sampling (ADASYN) or Synthetic Minority Over-sampling Technique (SMOTE) algorithm alongside various machine learning techniques detailed in the classification stage, the study evaluates their performance under different parameter configurations. Notably, XGB emerges as the top-performing approach across both balanced and imbalanced classes, achieving a superior accuracy.

3.4 An Adaptive Multi-class Imbalanced Classification Framework Based on Ensemble Methods and Deep Network

In paper [37], by Xuezheng Jiang et al. addressed the critical challenge of imbalanced classification in multi-class scenarios.

The proposed framework integrates ensemble methods and deep neural networks to enhance the classification performance across all classes, particularly focusing on improving the recognition of minority classes. This paper describes two techniques: Improved boosting framework FL-boosting and HAFL-Boosting Integration.

In FL boosting model, it is constructed using custom loss functions. These models implement specific formulas within the custom loss functions to accommodate the Focal Loss (FL) approach. The FL method integrates with Softmax for multi-classification tasks, offering ease of implementation within the boosting algorithm's custom loss function interface. Unlike traditional methods, such as the One-vs-Rest (OvR) algorithm, this approach eliminates the need for transformation, significantly reducing computational complexity and enhancing efficiency.

In second approach, HAFL-Boosting framework is used for deep image dataset classification. The author introduced a two-stage approach for addressing deep imbalanced data classification challenges. In the first stage, Feature Extraction with Deep Network is used, Where a deep neural network is employed to extract features from the dataset. This deep network is designed to mimic perceptions and performs both supervised and unsupervised learning. The features extracted by this deep network serve as input for subsequent processing. In the second stage, HAFL-Boosting is used, where the deep network is modified by removing layers from the top. This trimmed deep network is then combined with the HAFL-Boosting framework. HAFL-Boosting, a boosting algorithm tailored for handling imbalanced data, is applied to the features extracted by the deep network. By integrating HAFL-Boosting with the deep network, the model aims to improve classification performance on imbalanced datasets.

3.5 TabNet: Attentive Interpretable Tabular Learning

The work of Sercan O, Arik, Tomas Pfister [10] laid the ground work for TabNet. The author proposed a novel high-performance and interpretable canonical deep tabular data learning architecture named TabNet. At each decision step, it focuses on important features while employing a sequential attention mechanism, enhancing interpretability and optimizing learning efficiency by prioritizing key features. Their extensive experiments proved that TabNet performs better than other methods across different tabular datasets. The authors explored self-supervised learning methods designed for tabular data, demonstrating significant performance improvements, especially when there is plenty of unlabeled data. TabNet revolutionizes the processing of tabular data by working directly with raw data, eliminating the need for preprocessing. It seamlessly integrates into end-to-end learning pipelines thanks to its gradient descent-based optimization during training. Using sequential attention, TabNet dynamically selects pertinent features at each decision step, bolstering interpretability and learning efficiency by concentrating on the most critical features. Unlike traditional methods, TabNet uses a single deep learning structure for both choosing features and making decisions. These strategic choices empower TabNet to either surpass or rival other tabular learning models across diverse datasets and domains, excelling in classification and regression tasks alike. Furthermore, TabNet provides two levels of interpretability: local interpretability, which visualizes the importance and interactions of features, and global interpretability, which measures the contribution of each feature to the model.

By this approach, TabNet outperforms Decision Trees (DTs)[28] by using sparse, instance-wise feature selection, it helps create decision boundaries in hyperplane form. Its sequential multi-step architecture enhances learning capacity through nonlinear processing of selected features, mimicking ensembling via higher dimensions and more steps. Inspired by top-down attention mechanisms, TabNet's encoding architecture efficiently searches for relevant information in high-dimensional inputs. Employs a sequential attention mechanism to select semantically meaningful features at each decision step, enabling efficient learning and yielding interpretable decision makings.

3.6 Anomalies detection using a feed-forward deep learning network

In signal processing, anomaly detection plays an important role in preventing potential risks.

In research conducted by AL.Sulaiman [4], proposed a groundbreaking approach to directly detect the anomalies, without relying on large machine learning models like Decision tress and support vector machines. Data availability and access to diverse platforms are reshaping Information Systems, posing challenges in explaining complex relationships. Deep learning, a powerful tool in prediction and classification, addresses these challenges by capturing non-linearities and complex interactions while offering flexibility in incorporating relevant information and preventing over-fitting, making it valuable for detecting anomalies [30].

The paper opts for a deep feed-forward neural network instead of recurrent neural networks or long-short term memory networks, as the focus is on predicting anomalies rather than analyzing time series. And, also selection of an appropriate activation function is crucial in deep learning neural networks to expedite convergence and avoid convergence issues. Activation functions can be step, linear, or nonlinear, with the Rectified Linear Unit (ReLU) function being the most commonly used due to its ability to address the vanishing gradient problem. Further details about ReLU are provided in 4.3.5. Additionally, determining the proper structure of a deep neural network involves various factors such as selecting the number of hidden layers and nodes, activation function, and training protocol, which often requires experimental analysis. This study investigates the feasibility of predicting future outcomes based on significant changes in current data, utilizing thresholds to define these changes. Employing a deep forward neural network model, the research assesses the model's accuracy, robustness, and performance compared to benchmark algorithms, highlighting the importance of incorporating relevant factors to enhance predictive capabilities. Additionally, the study suggests integrating external factors to further improve predictive models' effectiveness in real-world applications.

4 Theoretical Concepts

This section majorly focuses on the theoretical foundations for dealing with class imbalance issues, as well as various machine learning and deep learning algorithms for multi-task classification. Throughout this chapter, three parts are discussed. First, the benefits and drawbacks of feature engineering are examined. Next, advantages and disadvantages of oversampling methods for classification tasks are discussed. Finally, the most popular machine learning and deep learning models for multi-class multi-output problems will be examined.

4.1 Feature Engineering

The feature engineering pipeline involves preprocessing steps that transform raw data into usable features for machine learning algorithms, which are used to build predictive models. These models typically comprise outcome and predictor variables which are also known as features, with the feature engineering process involving the creation and selection of the most relevant predictor variables [31]. Automated feature engineering has been integrated into certain machine learning software since 2016, involving three primary steps: Feature Transformations, Feature Extraction, and Feature Selection shown in Figure 1.

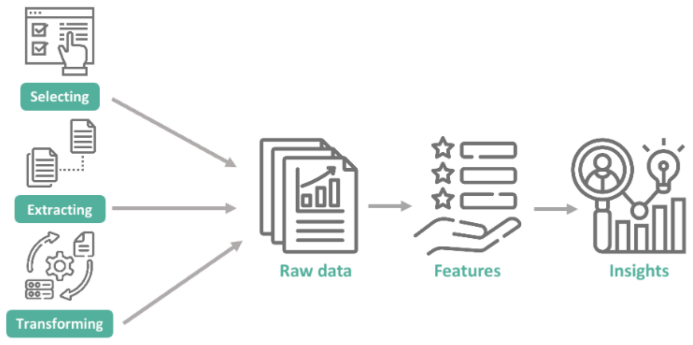


Figure 1: Feature Engineering [57]

In many applications, data often has high dimensions, with some features being less important than others [46]. Feature transformation includes altering, adjusting, or reorganizing current features in your dataset to extract deeper insights or to enhance their compatibility with machine learning algorithms [49]. The algorithm accuracy can be enhanced by removing unnecessary features with the application of feature selection and data transformation. The goal of every machine learning engineer is to maximize the predictive power of their models by transforming their data into a more informative and usable format.Unlike transformation-based methods, which modify existing input features, feature engineering also involves creating entirely new feature spaces. Feature transformation enables the conversion of diverse features into a unified format, simplifying the process for the model to recognize patterns and make accurate predictions [49]. In real world problems, the raw data might not fit to the pre-existing models or does not meet the requirements of out machine learning algorithms. Feature transformation involves reshaping these components to ensure compatibility and coherence. Feature transformation is important for both supervised and unsupervised tasks. Machine learning models often perform better when features are transformed to have similar scales or distributions, as this improves the model's ability to learn from the data.

Features can also be selected based on the feature importance score. Feature importance entails methods that assign a score to each input feature within a particular model, indicating the significance of each feature. A higher score indicates that the corresponding feature will exert a greater influence on the predictive model's ability to forecast a specific variable [55]. Feature importance can be calculated using Gini importance or through XGBoost [55] [13].

4.1.1 Benefits and Drawbacks of Feature Engineering

A fundamental understanding of the dynamic nature of feature engineering must first be established before delving into the nuanced exploration of deep learning. The fundamental capabilities that create important features which are valuable for recognizing patterns easily by the models will be explored, along with the challenges that need to be carefully considered in the theoretical landscape of machine learning. The feature importance plot highlights which features have the greatest impact on the model's predictions as shown in figure2. The top features contribute significantly more to the model's performance compared to the others, indicating their strong predictive power. Conversely, the least important features have a minimal influence, suggesting they provide less valuable information for the model.

As previously stated, feature engineering includes Feature Creation, Transformations, Feature Extraction, and Feature Selection. Each step plays an important role to generate new features and consists of more useful information than raw data. There are pre-existing methods for each steps to generate new features. Features can be selected or transformed through manual inspection or by feeding the raw data into pre-existing models to uncover hidden patterns. In the realm of manual inspection, one of the most direct approaches in feature engineering involves the removal of data. This could be due to redundancy, irrelevance, or data overload [45]. Redundancy poses both practical and technical challenges, as it leads to unnecessary storage and potential inefficiencies in learning systems. Irrelevant features not only occupy space but also mislead training paths, resulting in poor performance during testing. Manual techniques based on domain knowledge, random sampling methods, and model-based approaches that consider feature interaction with learning models. While manual and randomized selection are straightforward but, feature transformation techniques are better in most cases for determining the unseen probabilities, feature importance and hidden patterns [45].

Feature Engineering has several advantages, including:

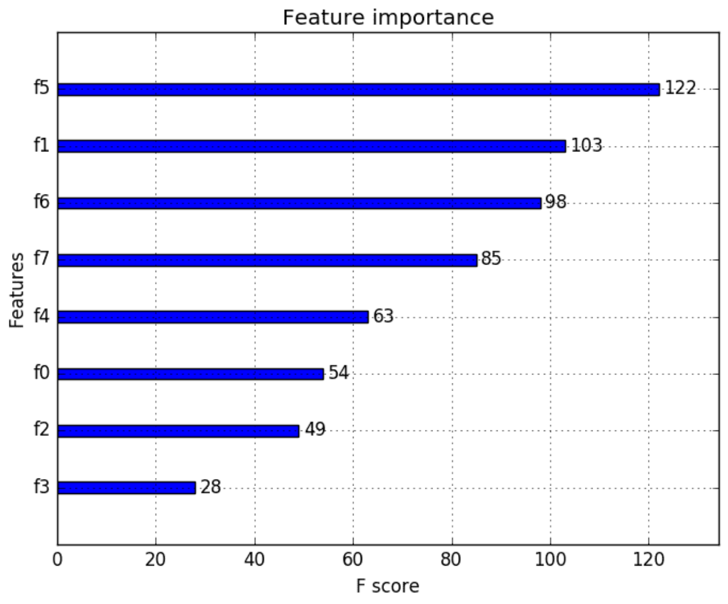


Figure 2: Feature Importance [13]

1. Feature engineering contributes to improved model accuracy and reduced overfitting by selecting and modifying relevant features, thereby ensuring better performance on both training and test data[60].

2. Feature engineering aids in reducing the dimensionality of datasets, particularly beneficial for high-dimensional data prone to overfitting. By selecting only the most important features, the complexity of the dataset is simplified, leading to improved model performance.

3. Simpler models that are easier to maintain and better understanding of the underlying problem [6].

4. It offers a solution to counteract the detrimental effects of outliers by adjusting their values to better align with the dataset, thus minimizing their negative impact on model performance.

The weak points are as follows:

1. The significant risk associated with feature engineering in machine learning arises from the potential for detrimental outcomes when applied without sufficient domain expertise, emphasizing the importance of understanding the technique's principles and implications beforehand[12].

2. Improper feature engineering may introduce bias or inadvertently remove valuable information, leading to sub-optimal model performance.

4.1.2 Challenges of Machine Learning with Feature Engineering

There are numerous challenges of applying Feature Engineering to huge data sets. As organizations harness the power of machine learning to extract insights and drive decision-making, navigating the complexities of feature engineering becomes paramount. However, amidst the promise of enhanced model performance and interpretability lie challenges which are particularly significant because they impact the reliability and accuracy of feature extraction and analysis processes. Following are the concerning issues:

1. Domain Expertise Essential: Successful feature engineering relies on domain expertise to discern which features are relevant and valuable within the context of the use case [23]. Implementing inappropriate transformations can have a cascading impact on both the performance and interpretability of the machine learning model [26].

2. Time-Consuming and Resource-Intensive: Manual feature engineering is a time-consuming and resource-intensive process that involves meticulous testing and refining of individual features. The iterative nature of feature engineering demands continuous exploration for more effective features, often resulting in weeks or months of refinement before identifying high-impact features.

3. Data quality issues: Data quality problems like missing data, outliers, and inaccuracies have the potential to impact both feature quality and the overall performance of the model [19].

4. Potential Errors and Biases: Manual feature engineering is susceptible to errors and biases due to its subjective nature, often leading to overstated connections between data phenomena based on preconceived notions [23]. The process is limited by the data scientist's perception, potentially overlooking novel insights and hindering the discovery of optimal features.

5. Over-fitting : Feature engineering may heighten the likelihood of over-fitting if the feature selection or transformation becomes overly intricate or tailored to the training dataset.

4.2 Oversampling Techniques

The imbalanced classification issue arises when there is a significant disparity in the distribution of classes within our training dataset. While the degree of skewness may vary, it poses a challenge because it can impact the performance of our machine learning algorithms. One common consequence of this imbalance is that the algorithm may overlook the minority class, which is typically the class of greatest interest [48]. The bias present in the training dataset can impact various machine learning algorithms, causing some to disregard the minority class altogether, which is often the class of primary interest for predictions [15]. To address and handle the issue of class imbalance, one strategy involves randomly adjusting the training dataset through resampling. This can be done by either removing instances from the majority class, known as under sampling, or replicating instances from the minority class, known as oversampling. In this section we will explore random oversampling and under sampling for imbalanced classification.

4.2.1 Random Undersampling

Random under-sampling technique is to handle the class imbalance in assigned labeled data by randomly selecting majority class instances and removing them from the training dataset [15]. This results in a reduction in the number of majority class instances in the transformed training dataset, potentially achieving a more balanced class distribution.

While this approach is suitable for datasets with class imbalance but a sufficient number of minority class instances, it comes with limitations. One significant limitation is the deletion of potentially valuable instances from the majority class, which could be crucial for fitting a robust decision boundary. Moreover, the random deletion of instances can lead to the loss of informative data, making the classification performance less effective.

4.2.2 Random oversampling

Random oversampling technique selects instances of minority classes and adds them to the training dataset [48]. This process entails selecting instances randomly with replacement, allowing examples from the minority class to be added multiple times to the new "more balanced" training dataset. Such a technique proves effective for algorithms sensitive to skewed distributions, particularly those that iteratively learn coefficients like artificial neural networks utilizing stochastic gradient descent.

Nevertheless, it is imperative to vigilantly monitor the performance of both the training and test datasets to evaluate the possibility of overfitting, particularly as oversampling could escalate computational expenses owing to the recurrent inclusion of minority class examples. So the problem can be tackled by introducing noise or create the instances close to the original data points. The focus will be on some popular sampling techniques relevant to imbalanced data classification.

1. Synthetic Minority Oversampling Technique (SMOTE)
2. Adaptive Synthetic (ADASYN)

Synthetic Minority Oversampling Technique (SMOTE) Synthetic Minority Oversampling Technique (SMOTE), is a widely utilized approach for tackling class imbalance in machine learning. The fundamental concept underlying the SMOTE algorithm involves creating synthetic data points for the minority class by interpolating between existing instances of that class. Essentially, SMOTE fabricates new data points artificially [25]. In this method, SMOTE randomly selects an instance from the minority class and identifies its k nearest neighbors within the same class.

The value of k determines the number of nearest neighbors used in the interpolation process, typically set to 5 as a default. "It subsequently produces new synthetic samples by interpolating between the original minority instance and its k closest neighbors as shown in figure 3 [25].

The advantages of SMOTE include its ability to generate additional samples based on the existing ones, thereby enriching the dataset and enhancing model performance. However, SMOTE has certain limitations. One significant drawback is the potential introduction of noise through synthetic instances, particularly when the number of nearest neighbors is excessively high. Additionally, SMOTE may struggle with tightly clustered minority class instances or datasets with a limited number of minority class instances.

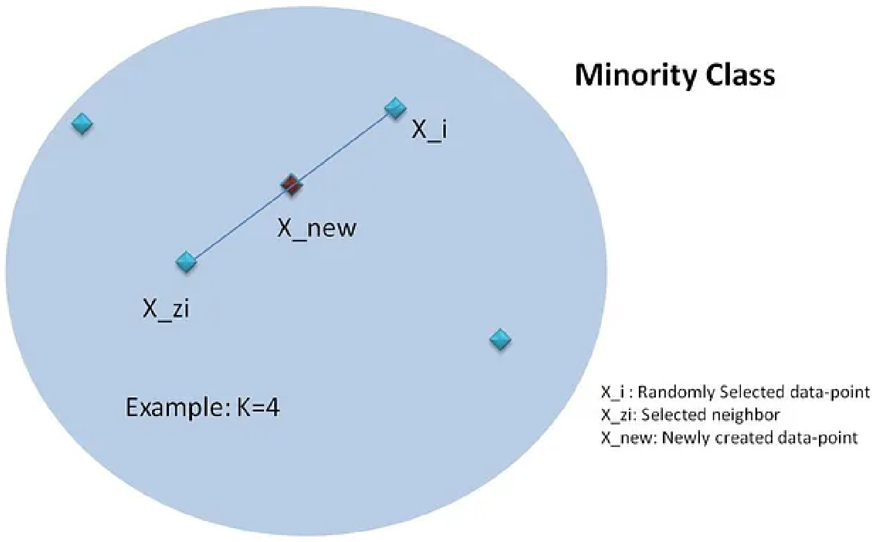

Minority Class
X_i
X_new
X_zi
Example: K=4
X_i : Randomly Selected data-point
X_zi: Selected neighbor
X_new: Newly created data-point

Figure 3: SMOTE oversampling [47]

Adaptive Synthetic (ADASYN)

ADASYN presents an alternative approach to oversampling, specifically targeting the generation of synthetic samples in areas of the feature space that lie closer to the decision boundary. It focuses on producing more synthetic samples for minority class instances that pose greater learning challenges, particularly those situated near the decision boundary.

In ADASYN, synthetic data samples are created from minority class instances that have a significant number of neighboring instances from the opposite class as shown in figure 4. The selection of templates for synthetic samples depends on the proportion of neighboring instances from the opposite class. These templates are then used to generate new examples through interpolation between the template and its nearest neighbors from the same class [2]. Similar to SMOTE, ADASYN aims to rebalance imbalanced datasets by generating synthetic instances of the minority class, with a focus on samples that are heterogeneous or difficult to learn.

The utilization of ADASYN in machine learning offers several advantages: it enhances classification performance for underrepresented classes, mitigates bias towards the majority class commonly observed in imbalanced datasets, improves the generalization capability of models by focusing on difficult-to-learn minority class instances, and finds applications across various domains such as intrusion detection, medical research, and fraud detection.

However, ADASYN also has some limitations. Firstly, the computational complexity may increase due to the creation of more synthetic instances, which can lead to longer training times for the machine learning models. [2]. Secondly, there is a risk of overfitting, as the synthetic samples may not accurately represent the true distribution of the minority class [2]. Lastly, ADASYN is sensitive to noise and outliers in the dataset, potentially impacting the quality of the generated synthetic samples.

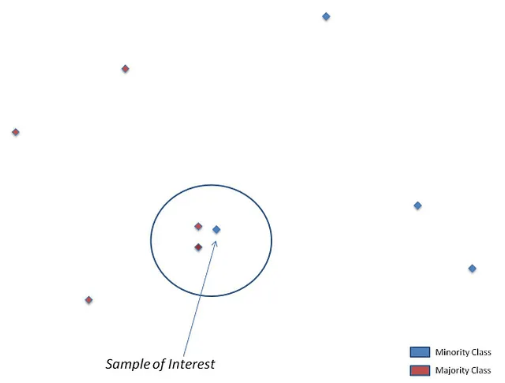


Figure 4: ADASYN oversampling [47]

4.3 Models for multi-task classification

When it comes to Machine Learning, a diverse array of algorithms and techniques offer distinct approaches to addressing classification tasks. These methods leverage various mathematical principles and computational strategies to learn patterns and relationships within data, facilitating accurate predictions and insightful analyses. With each algorithm, practitioners gain unique insights into the underlying structure of their datasets, enabling them to uncover valuable information and make informed decisions. In this section, The exploration will encompass Support Vector Machines (SVMs), Random Forest, XGBoost, TabNet, and Feed-forward Neural Networks for multi-class and multi-output classification tasks.

4.3.1 Multi-class SVM's

Classification entails the capacity of a machine to categorize instances into their respective groups [11]. Multi-class Classification involves categorizing entities into multiple classes, with each entity being allocated to a single class, ensuring no overlap occurs between classes. To perform this task effectively, the machine must learn the patterns associated with each category from the labeled training features available in a training dataset.

SVM, a supervised machine learning algorithm, is utilized for classification or regression tasks, striving to identify an optimal boundary among potential outputs. In essence, SVM undertakes intricate data transformations based on selected kernel functions, aiming to maximize the separation boundaries between data points according to predefined labels or classes. In its fundamental form, known as linear separation, SVM seeks to discover a line that optimally separates a two-class dataset within a two-dimensional space. In broader terms, the goal is to ascertain a hyperplane that maximizes the distinction between data points and their respective classes across an n-dimensional space. These data points, termed Support Vectors, are those closest to the hyperplane. Various kernel functions such as Linear, Polynomial, Gaussian, Radial Basis Function (RBF), and Sigmoid are employed, each influencing the smoothness and effectiveness of class separation. Modifying the hyper-parameters of these functions may result in overfitting or under-fitting [11]. In its simplest form, SVM is designed for binary classification tasks, separating data points into two classes. However, for multi-class classification, SVM employs a technique where the problem is decomposed into multiple binary classification sub problems.

One-vs-One: In One-vs-One (OvO) classification, the task of classifying instances is decomposed into pairwise comparisons between each class. For m classes, this approach trains with 'm(m-1)/2' binary classifiers, each distinguishing between a specific pair of classes as shown in Figure5. During prediction, each classifier casts a vote for its assigned class, and the class with the most votes is selected as the final prediction. OvO is particularly useful when dealing with binary classifiers that are efficient and well-suited for pairwise comparisons. However, as the number of classes increases, the number of classifiers required grows quadratically, potentially leading to increased computational complexity.

One-vs-Rest: OvR, also known as One-vs-All (OVA), involves training m binary classifiers, each designed to differentiate between one specific class and all other classes combined. In the figure 6 during training, each classifier learns to distinguish its designated class from the rest of the classes in the dataset. In the prediction phase, the instance is evaluated against each classifier, and the class with the highest confidence score becomes the predicted class. OvR is beneficial when dealing with classifiers that naturally support binary classification and are efficient with large datasets. However, OvR might lead to imbalanced training sets, especially if the

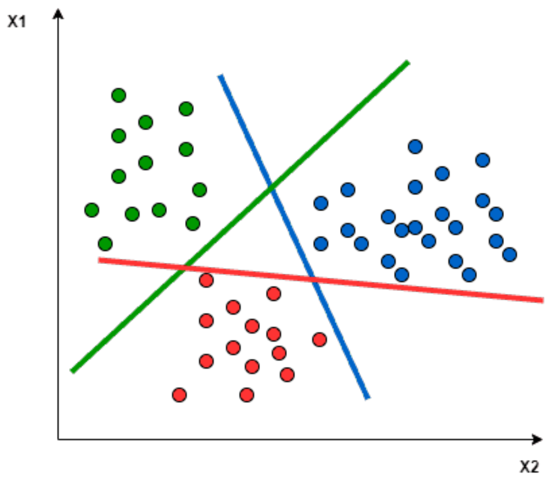


Figure 5: One-vs-One [11]

dataset is highly imbalanced across classes, which can affect the model's performance.

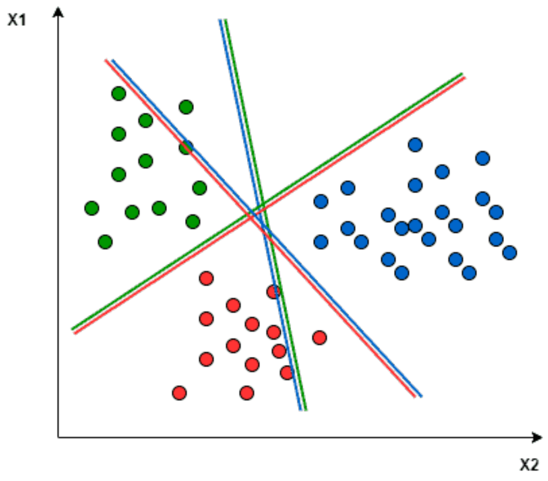


Figure 6: One-vs-Rest [11]

4.3.2 Random Forest Classifiers

A decision tree serves as a non-parametric supervised learning method employed in classification and regression scenarios. This hierarchical structure encompasses a root node, branches, internal nodes, and leaf nodes, offering a straightforward representation of decisions and their potential outcomes, incorporating various factors such as chance events and resource expenses [51]. Random forest, a supervised learning technique, constructs an ensemble of decision trees through a process typically known as bagging. This methodology involves training multiple decision trees and then combining their outputs to yield a more well grounded and robust prediction [21].

The random forest algorithm extends the bagging method by incorporating both bagging and feature randomness to construct an independent forest of decision trees as shown in figure 7. Feature randomness, also referred to as feature bagging or the "random subspace method," generates a random subset of features, ensuring minimal correlation among decision trees. This distinction is fundamental between decision trees and random forests, as decision trees consider all possible feature splits, whereas random forests only utilize a subset of these features.

In figure 7 , based on the majority voting, the class c is the final class from all the decision tress.

Random forests offer several advantages in the realm of machine learning. Firstly, they significantly reduce the risk of overfitting, bias, and overall variance by accommodating the complete range of data variability, resulting in more precise predictions [32]. Random forests are versatile, easy to tune, and resistant to overfitting with ample trees [21] .This ensures that the classifier maintains accuracy without tightly fitting the training data. Secondly, it provides robust solution for both regression and classification tasks with high accuracy. Moreover, they excel at estimating missing values, thanks to feature bagging.

Evaluating feature importance is also simplified with random forests, as they offer various methods such as Gini importance as discussed in 4.1 and mean decrease in impurity. However, random forests pose challenges as well. They can be time-consuming due to their ability to handle large datasets, which often involves processing numerous decision trees in parallel and they require more resources for storing data [40]. Furthermore, interpreting predictions from a random forest can be more complex compared to a single decision tree, adding a layer of intricacy to the modeling process.

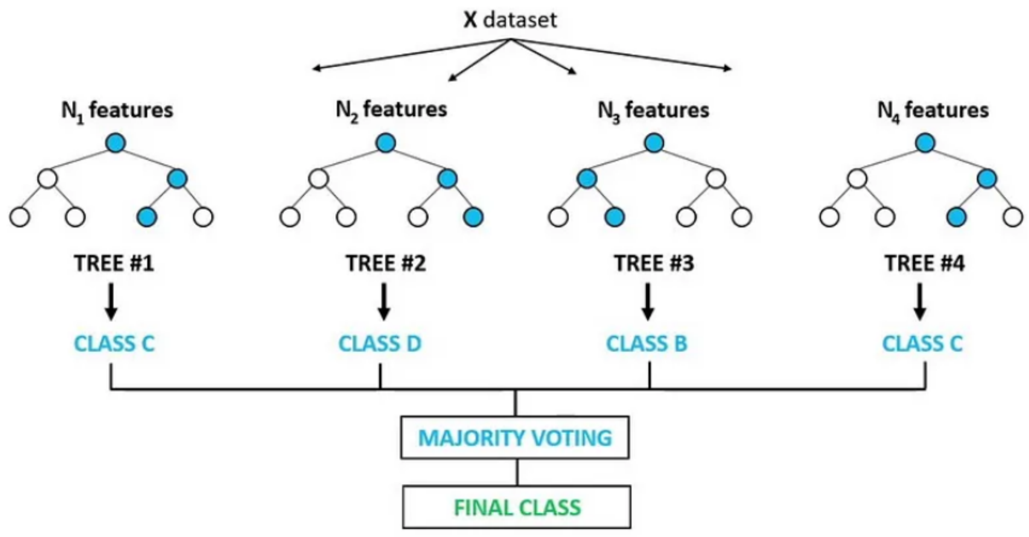


Figure 7: Random Forest Classifier [40]

4.3.3 XGBoost Classifiers

Extreme Gradient Boosting (XGBoost) is short for “Extreme Gradient Boosting”, with "Gradient Boosting" originating from Friedman’s paper [20]. XGBoost is a tree-based algorithm, designed for speed, ease of use, and performance on large datasets where where optimization and tuning are not required [56]. XGBoost is an optimized and parallelized version of the gradient boosting algorithm, significantly reducing training time [29]. Rather than focusing on training a single optimal model, XGBoost trains multiple models on various subsets of the training data and then selects the best-performing one through voting as shown in figure 8. In many scenarios, XGBoost outperforms traditional gradient boosting algorithms.

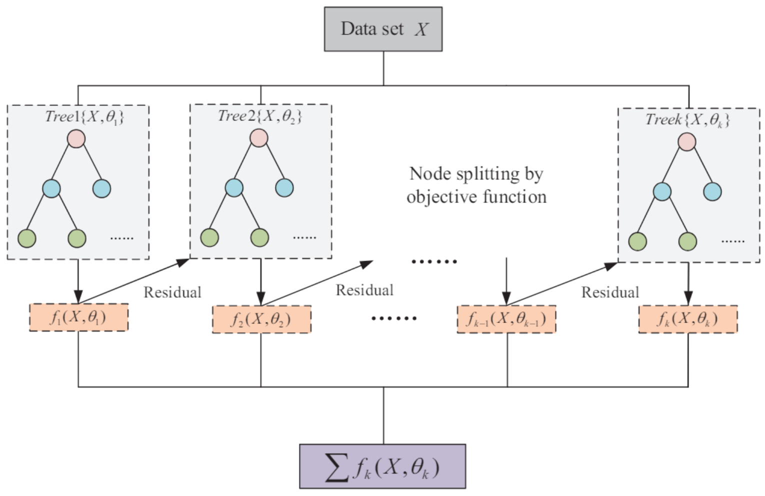


Figure 8: XGBoost Classifier [29]

Key features of XGBoost include parallelization for training with multiple CPU cores, regularization techniques to prevent overfitting [56], the ability to capture and learn from non-linear data patterns, and built-in cross-validation functionality [29].

In general, XGBoost demonstrates remarkable speed, especially in comparison to alternative implementations of gradient boosting. Gradient Boosting offers an algorithm that is straightforward to comprehend and interpret, simplifying the handling of most predictions [29]. Boosting serves as a sturdy and effective technique that adeptly combats and mitigates overfitting. XGBoost demonstrates remarkable performance[14][29], particularly on medium and small datasets with subgroups, as well as structured datasets featuring a moderate number of features. Given that the majority of real-world problems revolve around classification and regression tasks, XGBoost emerges as a dominant choice, excelling in these areas. XGBoost may not excel with sparse and unstructured data, and it’s essential to note that Gradient Boosting can be highly sensitive to outliers due to its iterative error-fixing nature. Additionally, the scalability of the method is limited as each estimator relies on the correctness of previous predictors, leading to challenges in streamlining the process.

4.3.4 TabNet

TabNet leverages a unique Deep Neural Network architecture, akin to tree models, to discern vital features and enhance data representation a promising solution to the challenges posed by tabular data [38]. Sercan O. Arik and Tomas Pfister introduced TabNet, a pioneering deep learning architecture designed for efficient and interpretable tabular data processing, as outlined in their work [10].

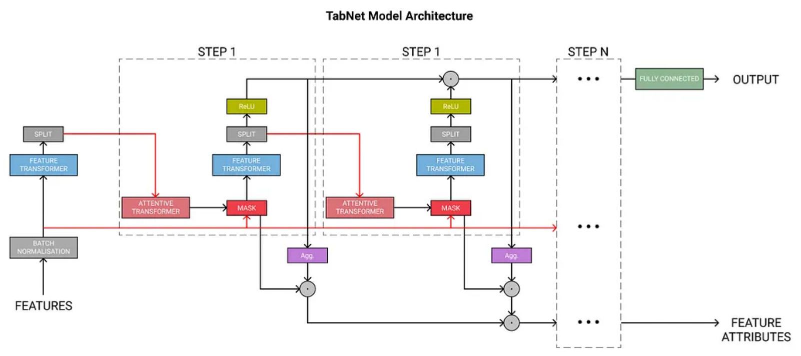


Figure 9: TabNet Architecture [10]

TabNet's encoder architecture operates in sequential steps, passing inputs through each step. A single step involves three processes: a feature transformer, an attentive transformer, and a mask are arranged as in Figure 9. The feature transformer utilizes a series of Gated Linear Unit (GLU) decision blocks to transform features. The attentive transformer employs sparse-matrix operations to select salient features, enhancing interpretability and learning[33]. A mask is then applied to generate decision parameters, which are fed into the next step. Initially, the entire dataset is taken without feature engineering, normalized, and processed through the feature transformer[27]. The output from this step provides predictions for continuous numbers or classes. Subsequently, the attentive transformer identifies important features, and their importance is aggregated across steps. This aggregated importance aids in explaining the model without requiring additional techniques like SHapley Additive Explanations (SHAP) or Local Interpretable Model-agnostic Explanations (LIME). These methods helps to break down the importance of each feature for individual predictions, which can be useful for understanding the model better. Finally, decision outputs from the feature transformers are aggregated and embedded for further processing.

The TabNet architecture is composed of two main parts: the Feature Transformer and the Attentive Transformer. The Feature Transformer uses four consecutive blocks with a Fully Connected Layer (FC), Batch Normalization Layer (BN), and GLU as shown in Figure 10. This setup ensures robust learning by sharing layers and stabilizing variance. The resulting outputs, no.of decision features (n(d)) and number of attention features (n(a)), are crucial for subsequent steps[27]. The Attentive Transformer includes FC and BN layers,which are involved in preprocessing the input

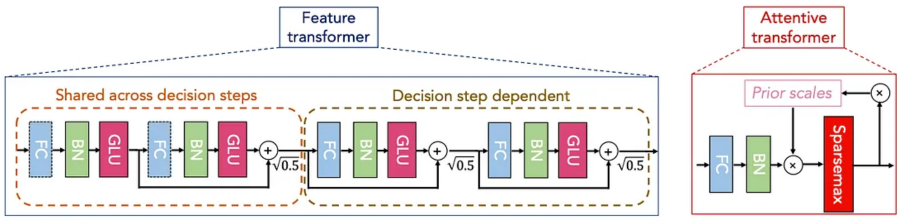


Figure 10: Feature Transformer and Attentive Transformer [10]

features. While the Prior Scales and Sparsemax layers fine-tune feature importance, aiding in more effective for decision-making within the TabNet architecture. It also incorporates an attention mask to identify selected features and understand aggregate feature importance. This comprehensive approach allows TabNet to perform feature selection and output mapping effectively within a single model.

By merging the advantages of both tree learning and neural networks, this approach facilitates the utilization of various learning techniques, including representation learning and meta-learning, in the context of tabular data. This fusion represents a significant advancement in tabular data analysis, introducing innovative strategies and perspectives to enhance the comprehension and utilization of this data format. It's attentive transformer enables sparse feature selection, enhancing the interpretability of the model's decision-making process. By employing soft feature selection, TabNet can focus on relevant features while disregarding irrelevant ones, improving its performance on tabular data.

4.3.5 Feed Forward Network

A feedforward neural network stands as a fundamental archetype within the realm of artificial neural networks, characterized by the unidirectional flow of information from input nodes through any hidden layers, if present, to the output nodes [18]. A feed forward neural network is structured with an input layer, hidden layers, and an output layer [39]. Neurons are linked by weighted connections of each layer, enabling the flow of information from input to output. The input layer receives the data, hidden layers performs the mathematical computations, and the output layer produces the final results as shown in Figure 11[35]. By adjusting connection weights, the network learns to minimize errors and improve accuracy in predicting outputs. At each layer, it introduces a non-linearity by employing activation functions, which incorporate non-linear shapes such as hyper-planes, thereby applying nonlinear transformations to the data. [54].

Activation functions are pivotal in feed forward neural networks, infusing non-linear characteristics that enable the model to discern intricate patterns. They are connected to the end of each hidden layers. Popular activation functions include sigmoid, tanh, and ReLU (Rectified Linear Unit)[18][41]. Training a feed forward neural network entails adjusting the weights of connections between neurons based on a dataset. This

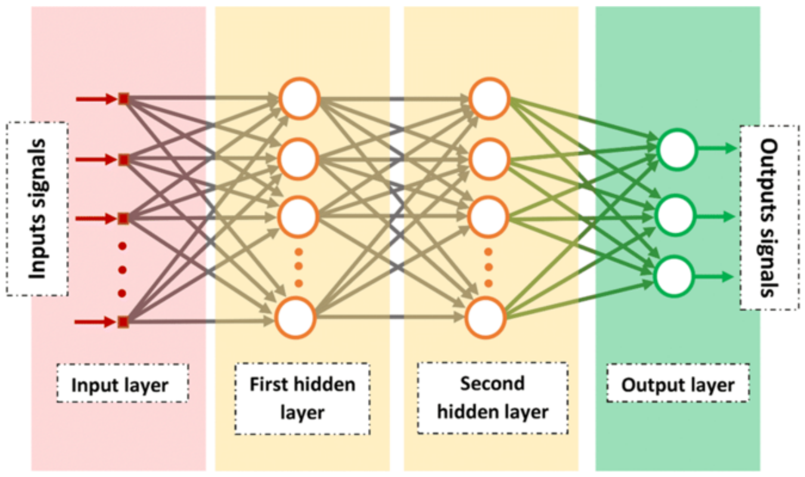


Figure 11: Feed Forward Neural Network [35]

iterative process involves passing the dataset through the network multiple times, with weights updated each time to minimize prediction errors. This iterative optimization process, known as gradient descent, persists until the network achieves satisfactory performance on the training data. Feed forward neural networks offer advantages such as ease of training and implementation, requiring fewer parameters and no back propagation through time. They operate faster and more efficiently, processing data in parallel without the need for storing previous states. However, they are less adept at handling sequential or temporal data and are prone to overfitting due to a lack of memory and regularization mechanisms.

4.3.6 Multi-output classifier's chain

Multi-output classifiers are a special class of machine learning models designed to handle tasks where each input can be associated with multiple labels or classes simultaneously. One popular approach to multi-output classification is the chain classifier method. In this method, a separate binary classifier is trained for each output in the dataset, forming a chain of classifiers. During prediction, each classifier in the chain not only predicts its corresponding output but also considers the predictions of the preceding classifiers in the chain. This approach allows the model to capture dependencies and correlations between different outputs, improving overall prediction accuracy. Chain classifiers are particularly useful in scenarios where outputs are interrelated or hierarchical in nature, such as multi- task classification tasks.

5 Implementation

This section delves into practical considerations when employing various techniques to tackle challenges in multi-class multi-output classification. It is divided into two parts: Python Implementation and Matlab/Simulink Implementation. In the Python implementation, approaches like Multi class SVMs, Random Forest classifiers, XGBoost classifiers, TabNet, and Feed forward networks were implemented and have proven effective. Methodological diversity is crucial for resilience and adaptability across different scenarios. This analysis includes a thorough exploration of these methods, covering data preprocessing, oversampling strategies, architectural choices, and tailored training methodologies for each model. This section provide detailed insights into the training and optimization of these machine learning and deep learning models. Finally, the model with the least complexity is selected for implementation in Simulink. In the Simulink implementation, it includes data preprocessing, construction of the prediction pipeline with the chosen model weights, and performs validation with ground truth data.

5.1 Python Implementation

In the Python implementation, commencing with the extraction of data from the AVL Concerto tool, the process entails organizing the information meticulously into a tabular format within Excel. Subsequently, introducing a mathematical function to assign the classes marks the transition to supervised learning. Following class assignment, the process advances to data preprocessing, oversampling, and model training.

5.1.1 Data Preprocessing

Initially, AVL Concerto was employed to visualize the dataset, exporting it to Excel for preprocessing. In preparation for supervised learning, classes are assigned to the output label columns. The input dataset encompasses 6 Pressure Indicated (PI Signal) values, 6 Torque generated when fuel is burnt (AI Signal) values, alongside supplementary inputs such as speed, data sginals and few other signals. As PI Signal signals and AI Signal signals are crucial for anomaly detection in engine performance monitoring. PI Signal signals provide insights into the combustion process within the engine cylinder, detecting anomalies such as misfires or incomplete combustion that may result from faulty spark plugs or sensor malfunctions. Similarly, AI Signal signals indicate changes in combustion efficiency or characteristics, highlighting issues like fuel quality problems or air-fuel mixture imbalances. Monitoring these signals enables early anomaly detection, facilitating timely maintenance to prevent potential damage or performance degradation. By analyzing PI Signal and AI Signal signals, engine performance deviations or malfunctions can be identified promptly, allowing for proactive maintenance and optimization efforts to maintain optimal performance, prevent failures, and minimize downtime. Also, PI Signal and AI Signal are correlated with all other signals. In the event of a deviation in PI Signal and AI Signal signals, a corresponding anomaly is expected in the data signal. The absence of PI Signal and AI Signal signals in the Engine Control Unit (ECU) precludes the use of mathematical functions for label prediction. Six output columns are created individually for PI Signal and AI Signal, resulting in a total of twelve output label columns. Utilizing six PI Signal values generates output labels associated with PI Signal, while six AI values generate output labels linked to AI Signal.

A mathematical function is introduced to classify the data into three classes: class1(good), class2(poor), and class3(extremely critically). For creating PI Signal output labels, outlier detection is applied using a threshold of 10 percent deviation from the average PI value. The values outside this range are considered as outliers. Outliers are then removed, and a new average PI is calculated for classification. Inputs within ±2 percent deviation are classified as class 1, those within ±8 percent as class 2, and deviations beyond 8 percent as class 3, indicating extreme criticality.

To generate AI Signal output labels, a similar approach is employed to identify outliers, but without using a threshold. Instead, the reference average AI is calculated by averaging the first ten observations of average AI Signal. Subsequently, another mathematical function for AI Signal is developed to assign classes. Unlike the previous PI Signal classification, which utilized deviation as a percentage, absolute values are used here. If the input AI deviates by ±4 from the average AI, it falls into class 1. If the deviation is ±10, it is categorized as class 2. Any deviation greater than ±10 is classified as class 3. Classes are assigned to each cylinder, resulting in a total of 12 output columns—6 for PI and 6 for AI, with each column containing three classes labeled as 1, 2, and 3. Through feature engineering, the other input signals were transformed, reducing the dimensions while identifying the slopes correlated with PI and AI.

5.1.2 SMOTE Oversampling

After preprocessing the data, the distribution of classes across the dataset is examined. In our scenario, the data is not uniformly distributed among all classes, resulting in a class imbalance issue. As outlined in theoretical concepts, SMOTE is among the most effective oversampling techniques for addressing class imbalance problems. The number of samples per each classes per each cylinder is shown in Figure 12 . For instance, in the case of cylinder 1, when examining the PI values, there are 574 samples for Class 1, 6 samples for Class 2, and 3 samples for Class 3. Similarly, while examining the AI values, there are 531 samples classified as Class 1, 46 samples classified as Class 2, and 6 samples classified as Class 3. The number of samples of class 1 is high when compared to class 2 and class 3. As discussed in the previous data preprocessing section, each cylinder consists of two output columns, i.e. PI output label and AI output label.

Figure 12 depicts a scenario where samples belonging to class-1 constitute the majority, while classes 2 and 3 are in the minority. When training a model using such a dataset, there's a risk of the model overfitting to class 1 and neglecting the other classes, resulting in a class imbalance problem. To address the class imbalance issue, employed the SMOTE oversampling technique. However, since a minimum of six observations for each class is required as discussed in chapter 4.2.2, and the dataset lacks this minimum for some classes. So random sampling is conducted first. Random sampling is like creating duplicates of the minority classes and ensures a minimum of six observations per class.

Cylinder number	PI			AI50		
	Class-1	Class-2	Class-3	Class-1	Class-2	Class-3
1	574	6	3	531	46	6
2	577	3	3	537	38	8
3	559	20	4	522	32	29
4	581	1	1	558	19	6
5	577	3	3	550	28	5
6	574	6	3	536	36	11

Figure 12: Distribution of classes before oversampling

Following random sampling, we apply SMOTE oversampling to generate additional samples for the minority classes, effectively mitigating the imbalance problem. A synthetic minority sample is generated by SMOTE by interpolating between existing minority class samples. This process involves selecting a minority class sample, identifying its nearest neighbors from the same class, and creating synthetic samples along the line segments connecting the sample to its neighbors.

Cylinder number	PI			AI50		
	Class-1	Class-2	Class-3	Class-1	Class-2	Class-3
1	1429	698	334	943	867	651
2	1375	587	430	920	879	593
3	1216	801	554	1527	522	522
4	1415	255	251	558	558	305
5	1203	699	508	927	880	603
6	1210	630	636	970	887	713

Figure 13: Distribution of classes after oversampling

By synthesizing new samples, SMOTE helps balance the class distribution, addressing the imbalance problem in the dataset. more samples are available for minority classes after oversampling, as shown in figure 13. As the frequency of the minority classes increases, the model endeavors to equally accommodate all classes when fitting the nonlinear function. Following the random sampling process, duplicates may exist in the oversampled dataset; these duplicates are subsequently eliminated.

5.1.3 Model Architecture and Training

After data preprocessing and data oversampling, final input features are given to the model, altogether six input features for each each cylinder, whereas outputs are PI and AI output label columns.

Multi class SVM's with Multi output classifier:

In this section, the multi-class SVMs are implemented with a multi-output classifier, as each cylinder has two output columns to predict simultaneously. After assigning the features and labels, the data is divided into training and testing sets, with 80% of the data allocated for training and the remaining 20% for testing. The architecture of the model is as illustrated in Figure 15. The input data is passed to Principle Component Analysis (PCA) block for dimensionality reduction. Data that is high-dimensional is transformed into a lower-dimensional space using PCA, while still preserving the majority of its variability [34]. It accomplishes this by identifying principal components, which are orthogonal vectors representing the directions of maximum variance in the data. The number of principle components created are 2 for each cylinder and shown in figure14. These components are organized based on their explained variance, enabling the selection of the most informative ones. Through the projection of data onto these principal components, PCA diminishes the number of features while conserving the fundamental information essential for modeling.

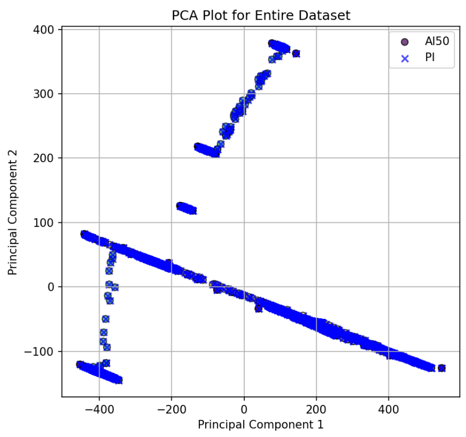


Figure 14: Principle Component Analysis for cylinder 1

After performing PCA, the principal components are utilized in Support Vector Machines (SVMs) employing the one-vs-rest (OvR) strategy, which is a supervised learning method utilized for multi-class classification tasks. This approach involves training a separate binary classifier for each class, where the samples belonging to the target class are treated as positive examples, and all other samples are treated as negative examples. While prediction, the class which is having highest confidence score from the binary classifiers is selected as the final prediction. By iteratively comparing each class against the rest, SVMs can effectively handle multi-class classification tasks with binary classifiers. The PCA and SVM are combined using the pipeline block. It is a convenient way to chain together multiple processing steps into a single estimator.

Continuing from the pipeline block, multi class SVMs with Multi-output classifiers are incorporated, which extend the concept of multi-class classification to scenarios where each sample may belong to multiple classes simultaneously, resulting in multi-label classification. These classifiers predict multiple output labels for each sample, with each label representing a distinct target variable.

This model trains the Multi-output classifiers which enable the modeling of complex relationships between input features and multiple output variables. This block provide a versatile framework for addressing multi-label classification problems by simultaneously predicting multiple output labels for each input sample, as 2 output columns are predicted for each cylinder. After training the multi-output classifier, predictions for each column will occur simultaneously, each with three classes.

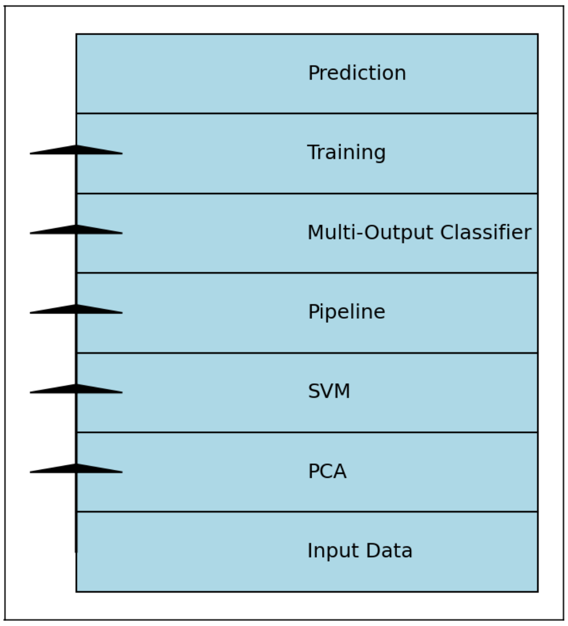


Figure 15: Architecture of Multi class SVM's with Multi output classifier

Random Forest Classifier with Multi-output Classifier:

In this model, the input features are passed on to random forest classifier which creates collection of decision trees, where each tree makes a prediction based on a random subset of features. The final output is based on the majority vote of the trees. The random forest classifiers is fitted into multi-output classifier which produces 2 random forests simultaneously with 2 output labels. Moving on, multi output classifier are trained and after being trained on the input data, the model can make predictions on new, unseen data. In the case of a multi-output classifier, the model would provide two outputs, corresponding to the two possible labels.

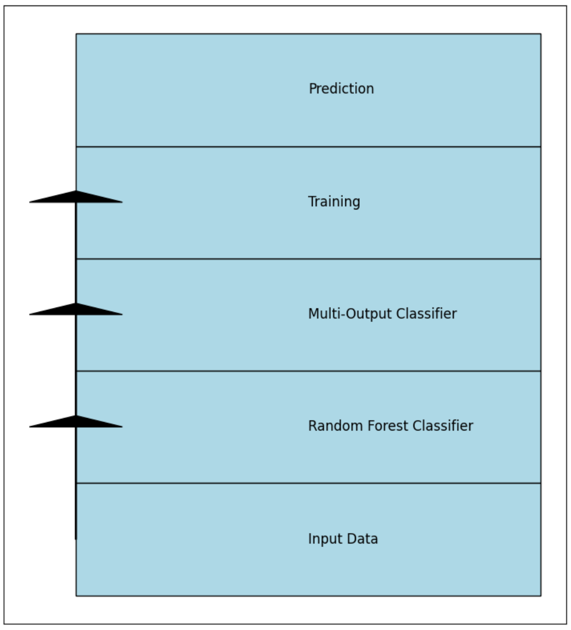


Figure 16: Architecture of Random Forest

This model tackles classification problems with two distinct output labels by leveraging a multi-output random forest architecture. At the core lies the concept of a random forest, a powerful ensemble learning technique. Imagine a forest teeming with individual decision trees, each acting as a classifier. Going in depth, as shown in the figure 16. The model starts by feeding your data with six input features into this forest.

To prevent the model from becoming overly reliant on specific features, each decision tree is built using a random subset of the six inputs. This injects diversity into the forest, making it more robust to potential biases in the training data. Each tree in the forest independently analyzes the data based on its chosen feature subset.

However, unlike a standard random forest predicting a single label, this multi-output version tackles the challenge of predicting two distinct labels simultaneously. Each decision tree within the ensemble makes individual predictions for multiple potential outputs. To achieve simultaneous prediction of multiple outputs, two key concepts are employed: Multi-Output Integration and the Wrapper Approach. In Multi-Output Integration, the prediction process is refined to handle multiple output labels. This approach utilizes a multi-output classifier, acting as a wrapper, to train separate random forests within the ensemble. The Wrapper Approach further enhances this process by encapsulating the individual random forests, ensuring that each specializes in predicting one output label while optimizing for the second label. This dual optimization strategy ensures that both prediction tasks are conducted effectively, leading to more accurate and comprehensive predictions.

After training on a labeled dataset, the model is ready to make predictions on unseen data. Presented with a new data point containing the six input features, the model sends it through each tree in the forest. Each tree provides individual predictions for both output labels.

Finally, the model harnesses the collective wisdom of the entire forest. It gathers the individual predictions for each output label from all the trees and performs a majority vote. The label receiving the most votes across the entire forest becomes the final prediction for that specific output. This approach leverages the strengths of multiple decision trees, leading to potentially more accurate predictions. In essence, this multi-output random forest architecture combines the power of random forests – handling complex relationships and reducing overfitting – with the ability to tackle classification problems with two distinct output labels. It provides a robust and versatile approach for various classification tasks.

XGBoost Classifier with Multi-output Classifier:

This model requires an additional preprocessing step, crucial for ensuring effective interpretation of the information by the XGBoost Classifier. The preprocessing involves converting the output classes from (1, 2, 3) to (0, 1, 2) format to align with the internal requirements of the XGBoost library. This conversion resolves compatibility issues and prevents errors that may arise during model training and prediction

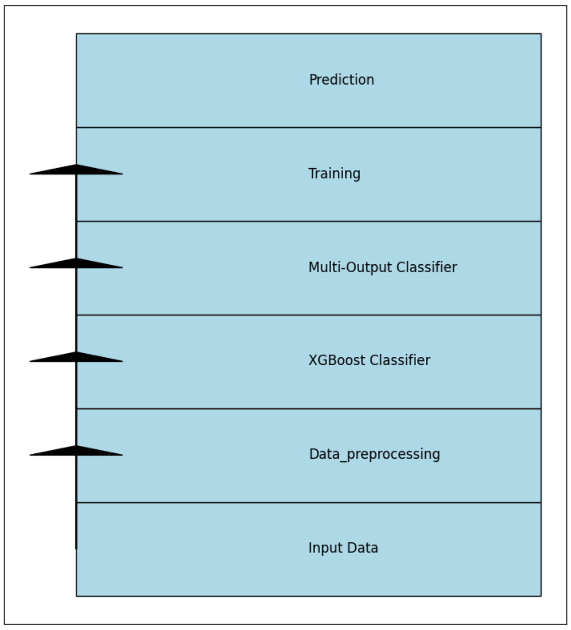


Figure 17: Architecture of XGBoost Classifier with Multi output classifier

After preprocessing the data, it is inputted into the XGBoost classifier, as illustrated in Figure 17, which combines multiple decision trees to make a more robust prediction. In this case, with two output labels (each having classes 0, 1, and 2), XGBoost will internally build an ensemble of trees specifically for each output label similar to random forests but additionally it computes residuals for each tree. The term "residuals" refers to the differences between observed values and the values predicted by a model. These residuals to iteratively improve the ensemble of trees, effectively minimizing the errors and enhancing the predictive performance of the model. Each tree focuses on learning the relationships between the 6 input features and the probabilities of each class (0, 1, or 2) for its corresponding output label.

As the model trains, it iteratively refines these trees to minimize prediction errors and it optimizes its internal structure (the trees and their connections) to accurately predict the class labels for both output variables. Further, the XGBoost classifier is fed into multi output classifier to generate the decision trees simultaneously for each output column, It uses the same wrapper approach as same as in random forest. Then the model is trained to predict the outputs with more accuracy.

TabNet with Multi-output Classifier:

The Figure 18 illustrates a machine learning architecture designed for a multi-output classification task using a TabNet classifier. The process begins with the collection of raw input data, which undergoes data preprocessing to clean and transform it into a suitable format. This preprocessing step includes handling missing values, encoding categorical variables, and normalizing numerical features.

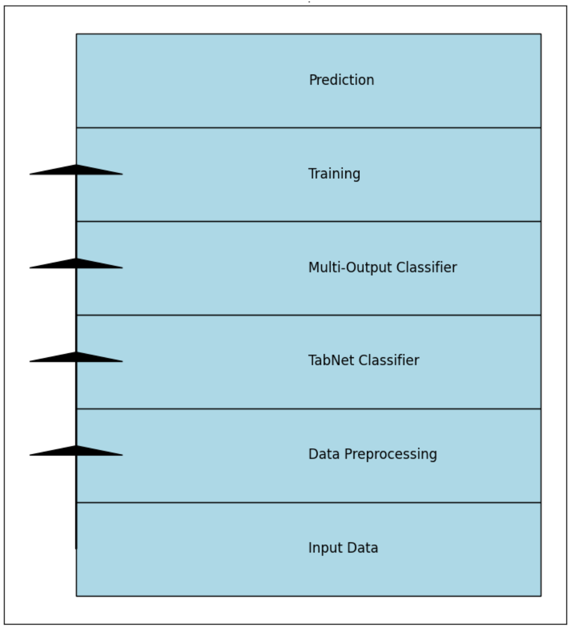


Figure 18: Architecture of TabNet with Multi output classifier

Additionally, for this model, preprocessing involves combining label columns into a single multi-dimensional array, standardizing the data, converting data to PyTorch tensors, and creating PyTorch Dataset and Data Loader objects. The preprocessed data is then fed into a TabNet classifier, a deep learning model optimized for tabular data. TabNet leverages sequential attention and decision steps to learn effective feature representations. This classifier is integrated within a multi-output framework, allowing it to predict multiple labels simultaneously for each data instance.

Once the data is prepared and the model framework is established, the system enters the training phase. Here, the multi-output TabNet classifier is trained on the processed data, optimizing its parameters to improve prediction accuracy. The training involves iterative learning to minimize the loss function, refining the model's ability to generalize from the training dataset. After training, the model is ready to make predictions on new, unseen data. The trained model is applied to the test dataset or any new input data to generate predictions for multiple output labels, and its performance is evaluated using relevant metrics.

This architecture effectively combines the advantages of deep learning and decision trees, providing a robust solution for complex, multi-output classification tasks.

Feed Forward Neural Network:

The network architecture in Figure 19 begins with the Input layer, which takes the input data. In the image, it has a shape of (None, 6), indicating that it can accept a batch of any size (represented by None) and each sample in the batch has 6 features. The Dense layer is a standard fully-connected layer, which is the most common type of layer in neural networks. The input data is transformed linearly, followed by a non-linear activation function. Activation functions that are non-linear help the network learn patterns of data that are more complex.

The first Dense layer in the image has 64 units, meaning it transforms the 6-dimensional input vector into a 64-dimensional vector. The output of the dense layer is then passed to a ReLU activation function. The BN layer is not a standard part of every neural network architecture, but it can be helpful for improving the training process. Batch normalization normalizes the activations of the previous layer, which can help to prevent the gradients from exploding or vanishing during training. This can make the training process more stable and efficient.

The Dense1 layer is another Dense layer, similar to the first one, but it takes the output of the Batch Normalization layer as input and has 32 units. So it transforms the 64-dimensional vector from the previous layer into a 32-dimensional vector. Batch Normalization1 is another Batch Normalization layer, similar to the one used earlier. The Output Layers form the final part of the architecture, which splits into two branches, each with a Dense layer. This suggests that the model is performing a multi-class classification task, where it needs to predict multiple categories for each input sample. The first Dense layer in this section has 3 units, and the second Dense layer also has 3 units. Each unit corresponds to a class. This indicates that the model is trying to classify the input data into 3 different categories. Furthermore, the outputs from these neurons are typically passed through a softmax function, which

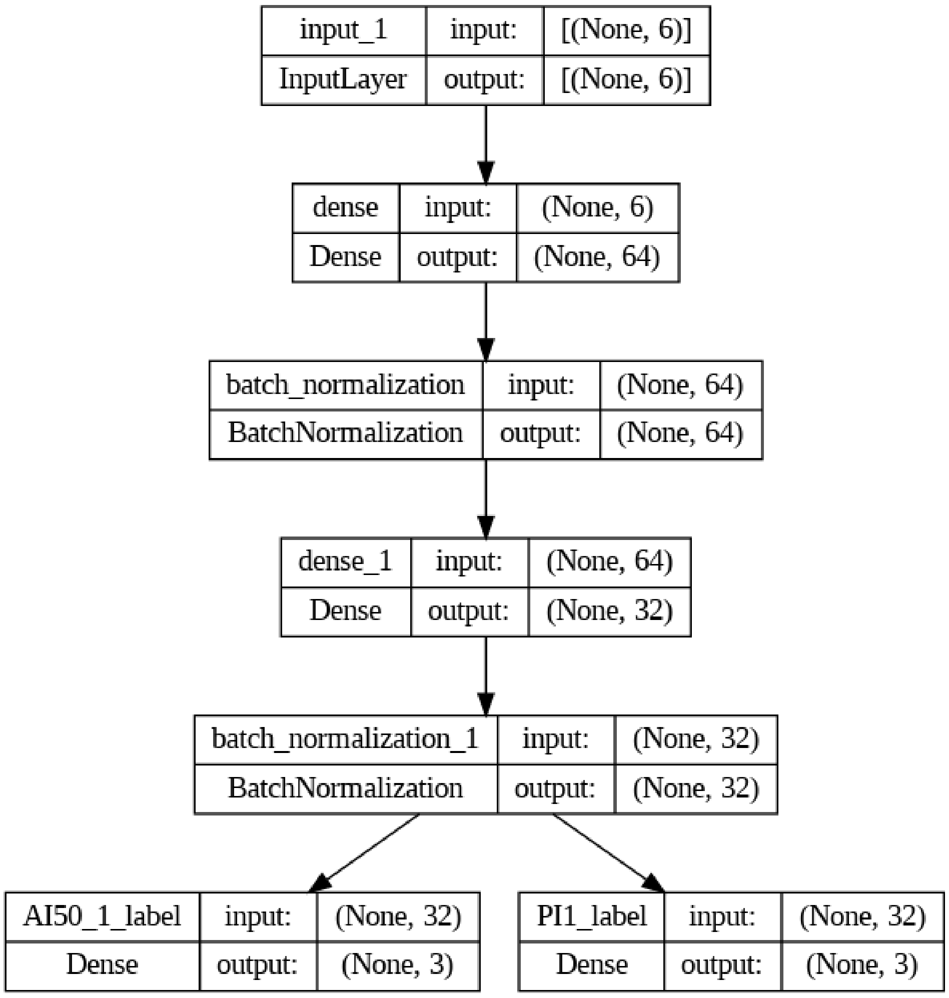


Figure 19: Architecture of Feed Forward Network

ensures that the maximum probability corresponds to the predicted class, enhancing the interpretability and stability of the model's predictions.

5.2 Matlab/Simulink Implementation

In the Simulink implementation, the trained model from Python is implemented in Simulink. In the Simulink model implementation, it undergoes data preprocessing in MATLAB, including data normalization and building the neural network as shown in the Figure 20. The best model from the above algorithms is considered into account by accuracy, prediction time and model complexity. The feed forward neural network is selected as the appropriate model for our scenario, considering these conditions.

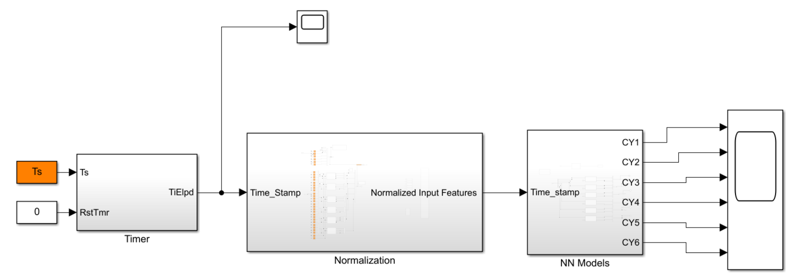


Figure 20: Simulink Model

5.2.1 Data preparation

Data preprocessing is done in the MATLAB environment. First, the parameters saved in CSV files is imported into MATLAB and load the weights and biases of each layer into the workspace. The raw data from the AVL Concerto tool is obtained, extracting features . Initially, it undergoes the same preprocessing steps as in Python to create the features for each cylinder. In total, we provide six features as input to the Simulink model.

5.2.2 Timer

Within this system, the Timer block is used to count up the time and select inputs from a table based on the timing.

5.2.3 Data Normalization

Once the input data is prepared from Data preparation block, it must be normalized because the model trained in the python was trained with normalized inputs, making it essential for predictions to also use normalized inputs. The mean and standard deviations are saved during the training phase in the Python environment. These are used for normalizing the input data during the prediction phase in the Simulink environment by using the equation 1. After normalization, the data is passed to the next block, the Feed Forward Neural Network (FFN). The complete architecture is shown in Figure 20 If the inputs are already normalized, they can be directly passed to the FFN block. This process is controlled by a switch: if the switch is set to 1, the data is considered normalized; if set to 0, the raw data is passed to the normalization blocks to produce normalized output features, which then serve as inputs to theFFN block. We use the Vector Concatenate block from Simulink to combine all the inputs into a single matrix.

$$\text{StandardScaler}(x) = \frac{x - \mu}{\sigma} \tag{1}$$

where:

- x is the input value,
- μ is the mean of the training data,
- σ is the standard deviation of the training data.

5.2.4 Feed Forward Neural Network

In Figure 20, the feed-forward block depicts the initial step where the input matrix is fed into a demux block within Simulink. The demux function serves to separate the matrix into its individual components, thereby creating six distinct features corresponding to the six cylinders. For each cylinder, a separate feed-forward network block is established. These network blocks possess two inputs and two outputs: one input represents the feature vector input and the cylinder number, while the outputs comprise predicted AI and PI labels.

The architecture established in the Python implementation is replicated to ensure consistency in predictions. As previously discussed, all weights and biases for each layer are preloaded into the workspace, enabling their direct utilization in mathematical operations.

Construction of the feed-forward network commences with the multiplication of inputs by the first dense layer, followed by addition with the corresponding biases. Subsequently, the output undergoes the ReLU activation function 2, defined as the maximum of zero and the output from the initial operation.

$$\text{ReLU}(x) = \max(0, x) \tag{2}$$

where:

- x is the input value,
- $\text{ReLU}(x)$ is the output of the ReLU function, which is x if x is greater than 0, and 0 otherwise.

The resulting output then proceeds to the batch normalization block, comprising parameters such as beta, gamma, mean, and variance, where the following mathematical operations are executed 3. Upon completion, the data is forwarded to the subsequent dense layer for analogous processing: multiplication with the dense layer's weights, addition of biases, followed by ReLU activation and batch normalization.

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{3}$$

where:

- β is a learnable shift parameter,
- γ is a learnable scale parameter,
- μ_B is the mini-batch mean,
- σ_B^2 is the mini-batch variance.

The resultant output is multiplied with the PI dense weights to obtain the vector probabilities. Similarly, for the AI predictions, the output from batch normalization is multiplied with the AI dense layer weights and biases. A manual implementation of the softmax function 4 determines the maximum probability among the generated output labels. Finally, the predicted labels are obtained individually from the model, concatenated using vector concatenation, and displayed in the scope. Predicted results are then saved to the workspace with a timestamp using the scope settings.

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{4}$$

where z_i is the i-th element of the input vector z, and N is the number of elements in the input vector.

6 Results & Discussion

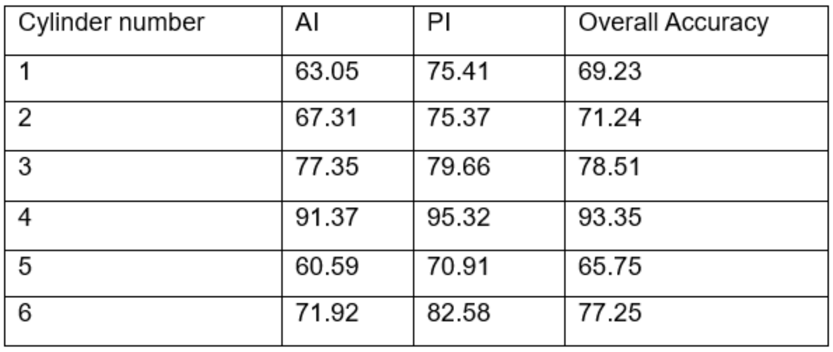
6.1 Python Results

The results section presents the predictions generated by various models discussed in the implementation methodologies. This chapter thoroughly examines the accuracy's alongside their respective confusion matrices. Before delving deeper, the definition of the confusion matrix and the process of calculating accuracies and false negatives, which denote misclassifications by the model, are elucidated.

In this scenario, a 3x3 confusion matrix is employed, where the y-axis displays the actual labels as ground truth and the x-axis denotes the predicted labels from the model. The diagonal of the matrix signifies correct predictions, while all other cells indicate miss classifications. A detailed calculation for each model will be delved into. The values along the diagonal of the confusion matrix indicates that the predicted label aligns with the true label. Off-diagonal elements, however, represent instances where the predicted label does not correspond to the true label, indicating miss classifications made by the model.

6.1.1 Multi class SVMs with Multi output classifier

In the implementation section, the utilization of training data, comprising 80% of the dataset, for model training purposes was discussed, with the remaining 20% of the dataset set aside for testing and prediction generation. Accuracy metrics are computed independently for each output column using the accuracy score function, allowing for an evaluation of the model's performance with regard to individual labels. Additionally, accuracy's can be calculated using the confusion matrix formula. Later, the overall accuracy is calculated by averaging the accuracy's across all output columns, providing insights into the model's capability to predict multiple labels simultaneously. Figure 21 depicts the accuracy's of output columns corresponding to each cylinder.



Cylinder number	AI	PI	Overall Accuracy
1	63.05	75.41	69.23
2	67.31	75.37	71.24
3	77.35	79.66	78.51
4	91.37	95.32	93.35
5	60.59	70.91	65.75
6	71.92	82.58	77.25

Figure 21: Accuracies of SVMs

The table presents performance metrics for six cylinders, each designated by a numerical identifier and associated with two metrics: PI and AI. These metrics likely contribute to combined "overall accuracy" values. Cylinder1 and cylinder5 exhibit the lowest accuracy's in relation to AI, standing at 63.05% and 60.59% respectively, indicating a notable deficiency. While overall accuracy's are noteworthy, individual label accuracy's hold significance. Conversely, models associated with cylinder 3 demonstrate commendable performance, achieving individual accuracy's of 91.37% and 95.32% respectively. Moreover, the overall accuracy for cylinder 3 surpasses that of the remaining cylinders, indicating superior predictive capability.

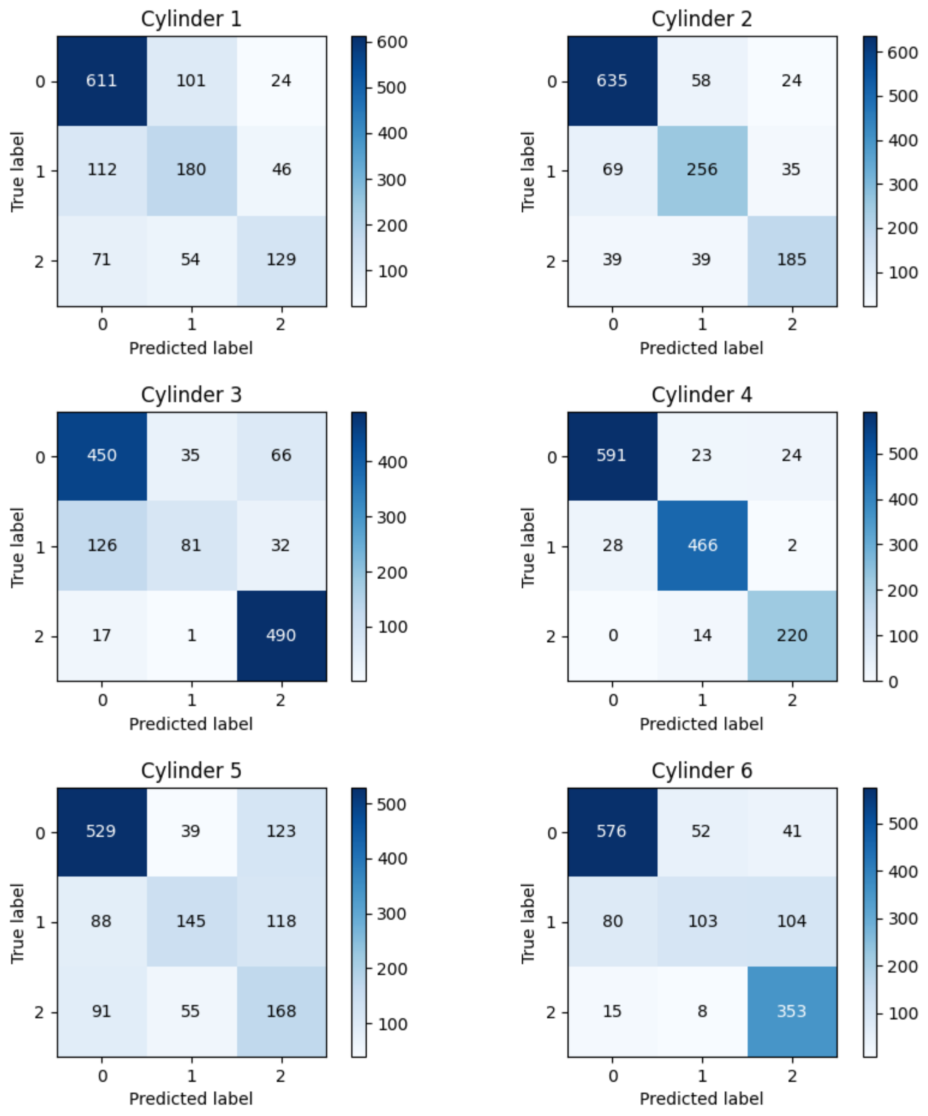


Figure 22: SVMs Confusion Matrices

The next step involves delving into confusion matrices, as outlined in the preceding section of the results, where the definition and mathematical computations necessary for determining accuracies and false negatives were discussed. False positives are particularly significant in assessing miss classifications and building trust in the model. These occurrences arise when the actual class of a data point is negative, yet the predicted class is positive, also known as Type I error.

The confusion matrices for all cylinders are depicted in Figure 22. For illustration, let's focus on the first confusion matrix pertaining to cylinder 1. The labels displayed on the axes represent the classes; in our scenario, classes 1, 2, and 3 are considered although the confusion matrix generated by the scikit-learn library considers these classes as 0, 1, and 2. Nonetheless, the underlying principles remain consistent. Class 0 denotes good, class 1 denotes poor, and class 2 denotes extremely poor. The total number of samples belonging to class 1 is 736, class 2 is 338, and class 3 is 259. However, as previously discussed, the diagonal blocks indicate correct predictions. Specifically, correct predictions related to class 1 amount to 611, class 2 to 180, and class 3 to 129. Individual accuracy's can be calculated using the formula provided below 5 and 6.

$$\text{Accuracy for each class} = \frac{\text{Correct Predictions of each class}}{\text{Total Samples}} \tag{5}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

Individual class accuracy's are determined by dividing the number of correctly predicted samples by the total number of samples belonging to each respective class. In our context, false negative rates are equally crucial to assess, are calculated using the formula 7. Where, True Positives (TP) occurs when a classification model correctly predicts a positive output, and the actual output is also a positive. False Negatives (FN) occurs when a classification model incorrectly predicts a negative outcome, but the actual outcome is positive. Safety concerns are paramount, as missed anomalies can lead to catastrophic failures such as fires or explosions in combustion chambers. Therefore, minimizing the False Negative Rate (FNR) is crucial to ensure safety, performance, and efficiency in combustion chambers. While it is important to keep the False Positive Rate (FPR) reasonably low to avoid unnecessary interventions and associated costs, the primary focus should be on ensuring that actual anomalies are not missed. FPR is calculated by using the formula 8 Given the high percentage of false negative rates for class 1 and class 2, labeling this model as the best choice would be inappropriate due to its subpar performance.

$$\text{False Negative rate} = \frac{FN}{FN + TP} \tag{7}$$

$$\text{False positive rate} = \frac{FP}{FP + TP} \tag{8}$$

Class 1:

$$\begin{aligned}
\text{Accuracy} &= \frac{611+409}{611+125+183+409} = \frac{1020}{1328} \approx 0.7675 \quad \text{(or 76.75\%)} \\
\text{FNR} &= \frac{125}{125+611} = \frac{125}{736} \approx 0.1699 \quad \text{(or 16.99\%)} \\
\text{FPR} &= \frac{183}{183+409} = \frac{183}{592} \approx 0.3091 \quad \text{(or 30.91\%)}
\end{aligned}$$

Class 2:

$$\begin{aligned}
\text{Accuracy} &= \frac{180+285}{180+158+256+285} = \frac{465}{879} \approx 0.5296 \quad \text{(or 52.96\%)} \\
\text{FNR} &= \frac{158}{158+180} = \frac{158}{338} \approx 0.4675 \quad \text{(or 46.75\%)} \\
\text{FPR} &= \frac{256}{256+285} = \frac{256}{541} \approx 0.4732 \quad \text{(or 47.32\%)}
\end{aligned}$$

Class 3:

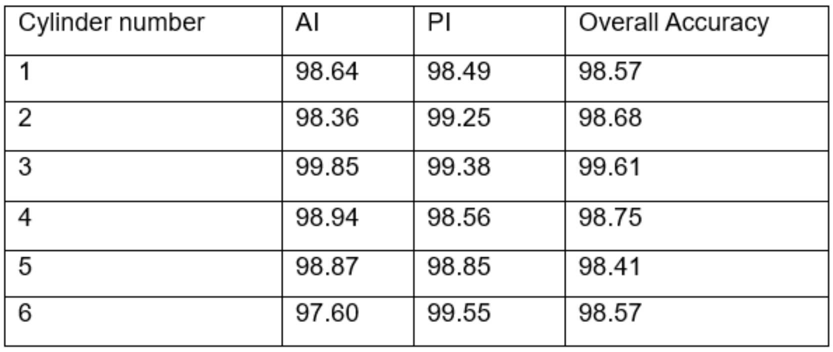
$$\begin{aligned}
\text{Accuracy} &= \frac{129+1004}{129+125+70+1004} = \frac{1133}{1328} \approx 0.8538 \quad \text{(or 85.38\%)} \\
\text{FNR} &= \frac{125}{125+129} = \frac{125}{254} \approx 0.4921 \quad \text{(or 49.21\%)} \\
\text{FPR} &= \frac{70}{70+1004} = \frac{70}{1074} \approx 0.0652 \quad \text{(or 6.52\%)}
\end{aligned}$$

Class 1 demonstrates a moderate accuracy of approximately 76.75%, with a false negative rate (FNR) of around 16.99% and a false positive rate (FPR) of approximately 30.91%. Class 2 exhibits a lower accuracy of about 52.96%, with a FNR of 46.75% and a FPR of approximately 47.32%. In contrast, Class 3 showcases an impressive accuracy of approximately 85.38%, with a notably lower FNR of around 49.21% and an even lower FPR of approximately 6.52%. The FPR regards to class 2 and class 3 are higher resulting incorrect positive predictions.

6.1.2 Random Forest Classifier with Multi output classifier

The model is trained using 80% of the dataset, with the remaining 20% set aside for testing and generating predictions. Accuracy metrics are then computed for each output column using the accuracy score function, enabling an evaluation of the model's performance for individual labels. Following this, the overall accuracy is calculated by averaging the accuracy's across all output columns, providing insight into the model's capacity to predict multiple labels concurrently. The table below presents the accuracies of output columns corresponding to each cylinder.

The table 23 offers comprehensive performance data for six distinct cylinders, each identified by a unique number. It records two specific metrics, AI and PI, for each



Cylinder number	AI	PI	Overall Accuracy
1	98.64	98.49	98.57
2	98.36	99.25	98.68
3	99.85	99.38	99.61
4	98.94	98.56	98.75
5	98.87	98.85	98.41
6	97.60	99.55	98.57

Figure 23: Accuracies of Random Forest Classifiers

cylinder, likely reflecting crucial characteristics or measurements pertaining to them. Furthermore, the table includes an "Overall accuracy" column, which likely integrates the performance on both AI and PI metrics to provide a holistic assessment of each cylinder's performance. The overall accuracy values range from 98.41% to 99.61%, indicating variability in performance among the cylinders. For example, cylinder number 3 exhibits the highest overall accuracy of 99.61%, with AI and PI values of 99.85% and 99.38% respectively, while cylinder number 5 displays a slightly lower accuracy of 98.41%, with corresponding AI and PI values of 98.87% and 98.85%.

The Figure 24 presents the confusion matrices for 6 cylinders, with cylinder 1 being discussed as an example. The total number of samples belonging to class 1, class 2, and class 3 are 736, 338, and 259 respectively. As previously mentioned, the diagonal blocks represent the correct predictions. Specifically, correct predictions for class 1, class 2, and class 3 amount to 724, 325, and 254 respectively. In the equation 7 is used to calculate the false negative rate for each class individually.

Class 1:

$$\text{Accuracy} = \frac{724+578}{724+12+9+578} = \frac{1302}{1323} \approx 0.9842 \quad \text{(or 98.42\%)}$$
$$\text{FNR} = \frac{12}{12+724} = \frac{12}{736} \approx 0.0163 \quad \text{(or 1.63\%)}$$
$$\text{FPR} = \frac{9}{9+578} = \frac{9}{587} \approx 0.0153 \quad \text{(or 1.53\%)}$$

Class 2:

$$\text{Accuracy} = \frac{325+978}{325+13+12+978} = \frac{1303}{1328} \approx 0.9812 \quad \text{(or 98.12\%)}$$
$$\text{FNR} = \frac{13}{13+325} = \frac{13}{338} \approx 0.0385 \quad \text{(or 3.85\%)}$$
$$\text{FPR} = \frac{12}{12+978} = \frac{12}{990} \approx 0.0121 \quad \text{(or 1.21\%)}$$

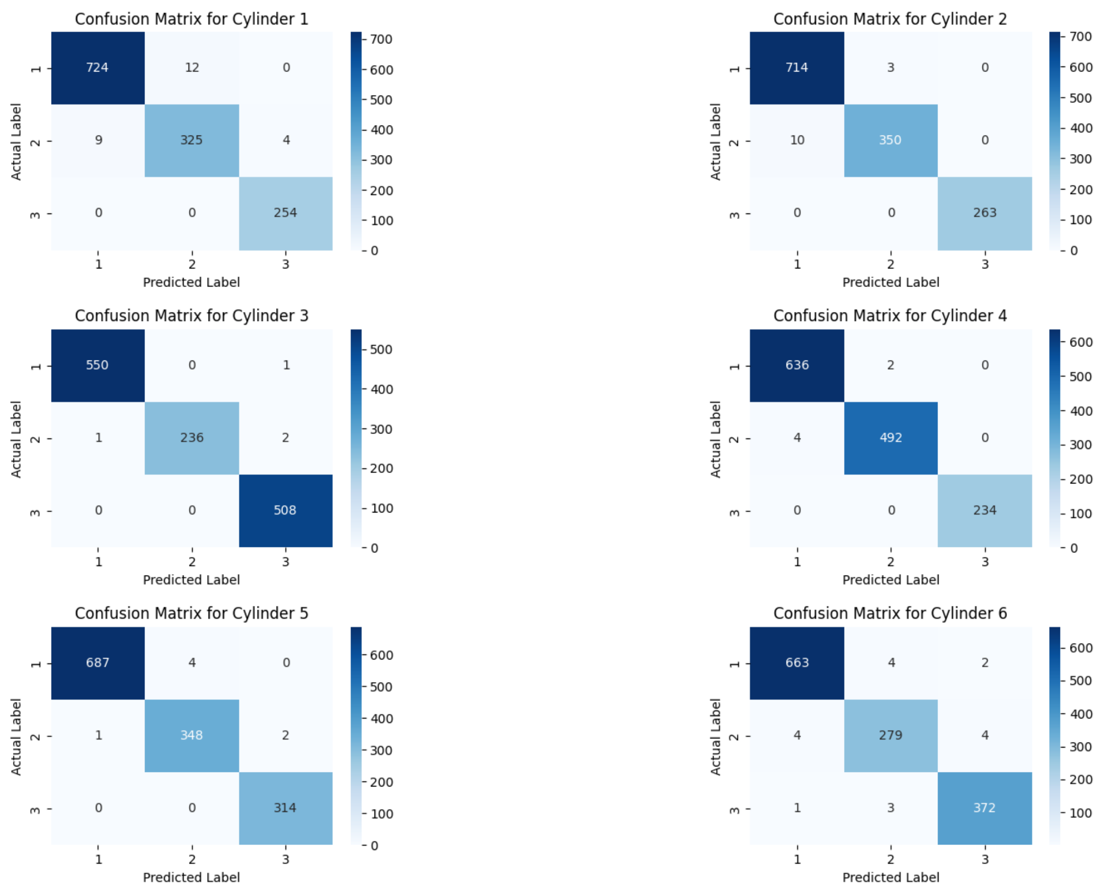


Figure 24: Random Forest confusion matrices

Class 3:

$$\text{Accuracy} = \frac{254+1070}{254+0+0+1070} = \frac{1324}{1324} = 1 \quad \text{(or 100\%)}$$

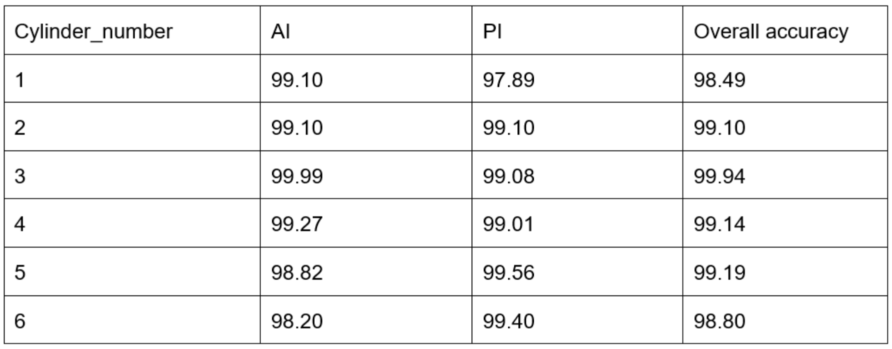
$$\text{FNR} = \frac{0}{0+254} = 0 \quad \text{(or 0\%)}$$

$$\text{FPR} = \frac{4}{4+1070} \approx 0.0037 \quad \text{(or 0.37\%)}$$

In comparing the performance across three classes, it's evident that each exhibits varying levels of accuracy and error rates. Class 1 achieves an accuracy of approximately 98.42%, with a low false negative rate (FNR) of around 1.63% and a moderate false positive rate (FPR) of approximately 1.53%. Class 2 also demonstrates a high accuracy of about 98.12%, albeit with a slightly higher FNR of around 3.85% and a relatively lower FPR of approximately 1.21%. Notably, Class 3 stands out as a model with perfect accuracy, achieving 100% accuracy, and boasting zero false negatives and 0.37% positives. Our model demonstrates perfect accuracy in predicting the extremely critical class, which is particularly significant. These findings highlight the superior performance of the random forest model, particularly for class 3 across all cylinders, with accuracy's ranging from 98% to 99% for class 1 and class 2 as well.

6.1.3 XGBoost classifier with Multi output classifier

Moving forward to XGBoost classifier with multi output classifier, The accuracy findings are prominently illustrated for each cylinder in Figure 25. Across the models, accuracy rates ranging from 97.89% to 99.99% are observed for both PI and AI outputs.



Cylinder_number	AI	PI	Overall accuracy
1	99.10	97.89	98.49
2	99.10	99.10	99.10
3	99.99	99.08	99.94
4	99.27	99.01	99.14
5	98.82	99.56	99.19
6	98.20	99.40	98.80

Figure 25: XGBoost Accuracy

Notably, the lowest accuracy score is recorded for the PI output of cylinder 1, while the highest is attained for the AI output of cylinder 3. Impressively, cylinder 3 attains the highest overall accuracy of 99.94%, underscoring its exceptional predictive performance. Conversely, cylinder 1 registers the lowest overall accuracy of 98.49%, reflecting comparatively less precision in predictions. Certainly, the variability in performance among the cylinders can indeed be attributed to the discrepancy in the available samples for training each cylinder. When certain cylinders have a greater abundance of samples across all classes, the resulting model tends to be more robust and accurate in classifying instances associated with those cylinders. Conversely, cylinders with fewer available samples may lead to less reliable models with higher error rates. This difference in sample size directly impacts the model's ability to learn and generalize patterns effectively. Consequently, the performance disparity observed across cylinders can be largely attributed to the inherent imbalance in the dataset, emphasizing the importance of equitable sampling strategies to ensure fair and representative model training across all cylinders.

The confusion matrices depicted in Figure 26 provide insights into the classification performance of each cylinder. Notably, cylinder 1 exhibits a greater number of miss classifications compared to the other cylinders, particularly evident in class 2 where 15 inputs are misclassified. Conversely, in cylinder 6, three inputs with class 3 are misclassified, while for the remaining cylinders, the number of misclassified inputs is negligible. For instance, cylinder 1 is taken as example; In class 1, comprising a total of 736 samples, the model accurately predicted 727 instances, while 9 samples were falsely classified. Moving to class 2, consisting of 338 samples, 323 were correctly identified, with 13 false positives recorded. Remarkably, class 3 witnessed perfect

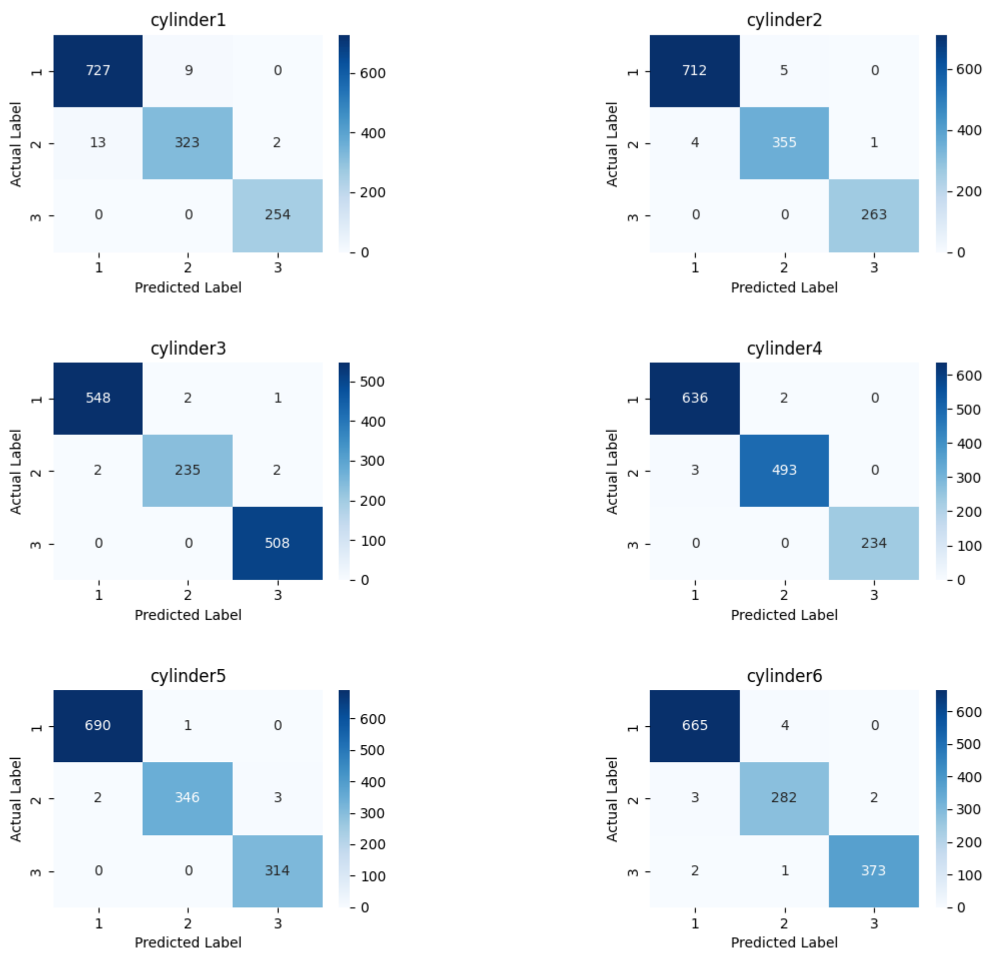


Figure 26: XGBoost Confusion matrices

prediction, with all 254 samples correctly classified and no false positives encountered. By utilizing the equation 7 8 below are the false negative rates and false positive rates for each class.

Class 1:

$$\text{Accuracy} = \frac{727 + 579}{727 + 9 + 13 + 579} = \frac{1306}{1328} \approx 0.9831 \quad (\text{or } 98.31\%)$$

$$\text{FNR} = \frac{9}{9 + 727} = \frac{9}{736} \approx 0.0122 \quad (\text{or } 1.22\%)$$

$$\text{FPR} = \frac{13}{13 + 579} = \frac{13}{592} \approx 0.0219 \quad (\text{or } 2.19\%)$$

Class 2:

$$
\begin{aligned}
\text{Accuracy} &= \frac{323+983}{323+15+9+983} = \frac{1306}{1330} \approx 0.9812 \quad \text{(or 98.12\%)} \\
\text{FNR} &= \frac{15}{15+323} = \frac{15}{338} \approx 0.0444 \quad \text{(or 4.44\%)} \\
\text{FPR} &= \frac{9}{9+983} = \frac{9}{992} \approx 0.0091 \quad \text{(or 0.91\%)}
\end{aligned}
$$

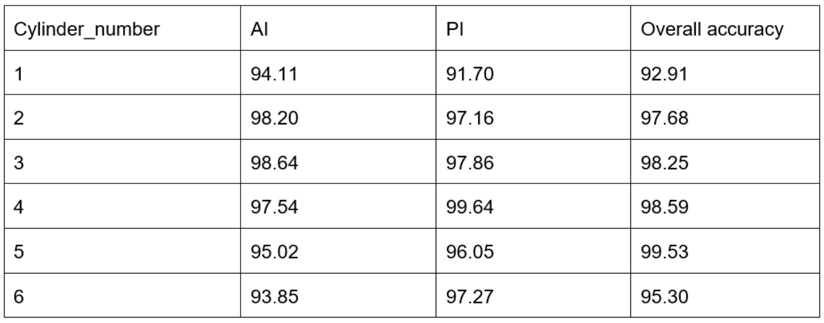
Class 3:

$$
\begin{aligned}
\text{Accuracy} &= \frac{254+1072}{254+0+2+1072} = \frac{1326}{1328} \approx 0.9985 \quad \text{(or 99.85\%)} \\
\text{FNR} &= \frac{0}{0+254} = 0 \quad \text{(or 0\%)} \\
\text{FPR} &= \frac{2}{2+1072} = \frac{2}{1074} \approx 0.0019 \quad \text{(or 0.19\%)}
\end{aligned}
$$

Comparing Classes 1 to 3, Class 3 emerges as critical due to its potential impact. While Class 1 exhibits an accuracy of approximately 98.31%, with a False Negative Rate (FNR) of around 1.22% and a False Positive Rate (FPR) of approximately 2.19%, and Class 2 showcases an impressive accuracy of about 99.12%, with a FNR of 4.44% and a FPR of approximately 0.91%, Class 3 stands out with its perfect accuracy, FNR of zero and extremely low FPR of around 0.19%. This implies that there are no miss classifications of actual positive cases in Class 3, highlighting its critical nature.

6.1.4 TabNet classifier with Multi output classifier

The TabNet classifier with multi-output functionality will be explored. The accuracy results for each cylinder are prominently displayed in Figure 27. Across the models, accuracy rates ranging from 92.91% to 99.53% are observed for both PI and AI outputs.



Cylinder_number	AI	PI	Overall accuracy
1	94.11	91.70	92.91
2	98.20	97.16	97.68
3	98.64	97.86	98.25
4	97.54	99.64	98.59
5	95.02	96.05	99.53
6	93.85	97.27	95.30

Figure 27: TabNet Accuracy

Notably, the lowest accuracy scores are recorded for the PI outputs of cylinders 6 and 1, while the highest accuracy is achieved for the AI output of cylinder 4. Remarkably, cylinder 4 demonstrates the highest overall accuracy of 99.53%, highlighting its exceptional predictive capability. Conversely, cylinder 1 exhibits the lowest overall accuracy of 92.91%, indicating relatively lower precision in its predictions.

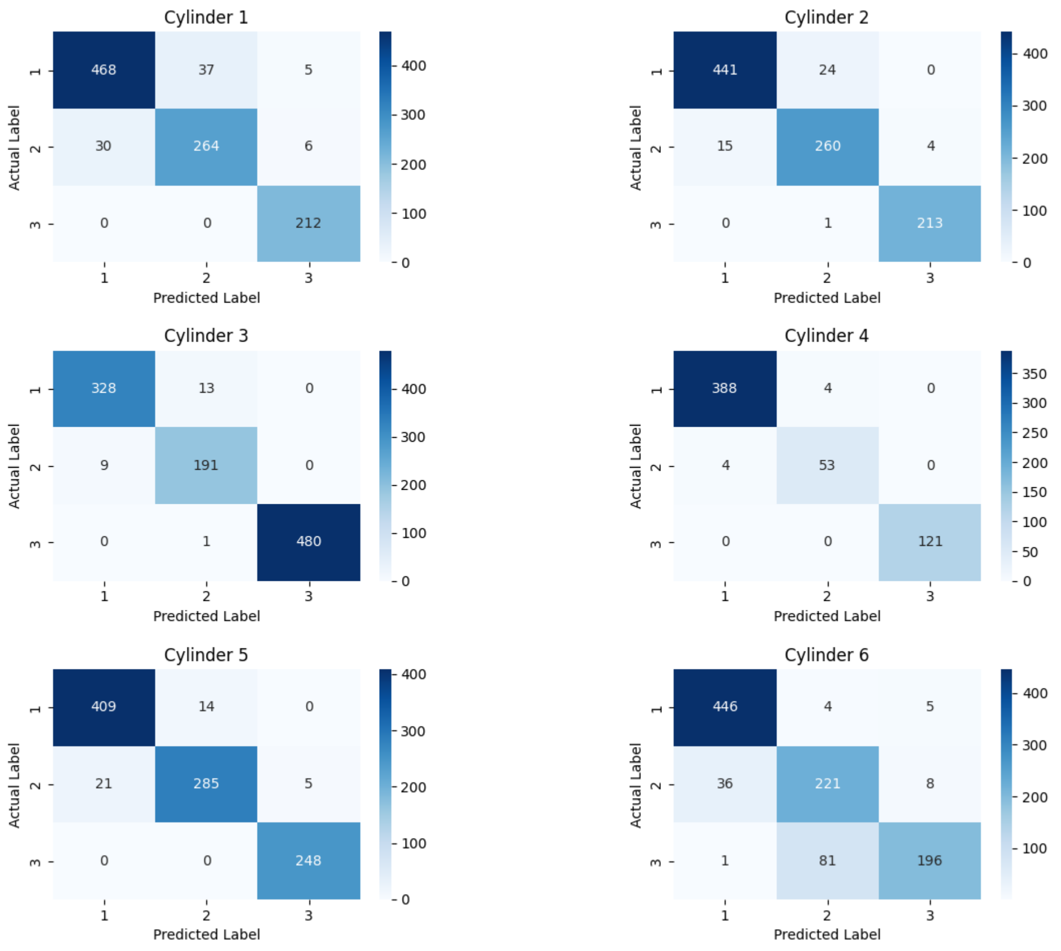


Figure 28: TabNet Confusion Matrices

Figure 28 presents the confusion matrices for each cylinder. Among all cylinders, cylinder 6 stands out with a higher number of miss classifications. Specifically, for class 3, 82 samples were misclassified, while 44 samples of class 2 were misclassified for the same cylinder. For example, let's consider cylinder 1: In class 1, comprising a total of 510 samples, the model accurately predicted 468 instances, while 42 samples were falsely classified. Transitioning to class 2, consisting of 300 samples, 264 were correctly identified, with 36 false positives recorded. Utilizing equation 7 and 8, false negative rate and false positive rate for each class can be determined.

For class 1:

$$\begin{aligned}
\text{Accuracy}_{\text{class 1}} &= \frac{468+482}{468+482+30+42} = \frac{950}{1022} \approx 0.929 \\
\text{FNR}_{\text{class 1}} &= \frac{42}{42+468} = \frac{42}{510} \approx 0.082 \\
\text{FPR}_{\text{class 1}} &= \frac{30}{30+482} = \frac{30}{512} \approx 0.059
\end{aligned}$$

For class 2:

$$\begin{aligned}
\text{Accuracy}_{\text{class 2}} &= \frac{264+685}{264+685+37+36} = \frac{949}{1022} \approx 0.928 \\
\text{FNR}_{\text{class 2}} &= \frac{36}{36+264} = \frac{36}{300} = 0.12 \\
\text{FPR}_{\text{class 2}} &= \frac{37}{37+685} = \frac{37}{722} \approx 0.051
\end{aligned}$$

For class 3:

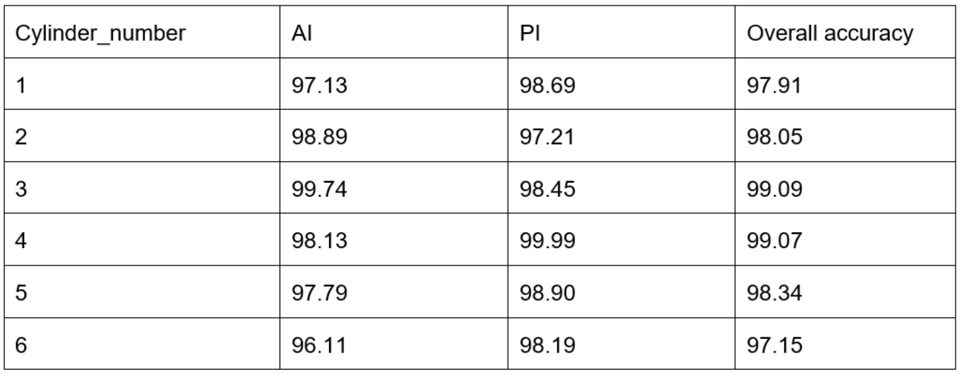
$$\begin{aligned}
\text{Accuracy}_{\text{class 3}} &= \frac{212+779}{212+779+11+0} = \frac{991}{1002} \approx 0.989 \\
\text{FNR}_{\text{class 3}} &= \frac{0}{0+212} = 0 \\
\text{FPR}_{\text{class 3}} &= \frac{11}{11+779} = \frac{11}{790} \approx 0.014
\end{aligned}$$

Class 3 stands out with the highest accuracy at 98.9%, showcasing its proficiency in correctly predicting both positive and negative instances. This is closely followed by Class 2, which achieves an accuracy of 92.8%. However, when examining the false negative rates (FNR), it becomes evident that Class 3 performs flawlessly with no instances of false negatives, whereas class 2 exhibits the highest FNR at 12%, indicating a notable proportion of positive instances being misclassified. On the other hand, in terms of false positive rates (FPR), Class 3 again excels with the lowest rate at 1.4%, demonstrating its precision in avoiding false predictions. Conversely, Class 1 displays the highest FPR at 5.1%, indicating a relatively higher rate of false alarms.

6.1.5 Feed Forward Network

This analysis explores the Feed-Forward Neural Network (FFNN) architecture, specifically its multi-output functionality. The accuracy results for each cylinder are prominently displayed in Figure 29. The accuracy of both the PI and AI outputs generated by the model is impressive, ranging from 96.11% to 99.99%.

Calculate the accuracies using equation 6. Interestingly, the model struggles most with the PI outputs of cylinder 6, whereas it performs best on the AI output of cylinder 4. Remarkably, cylinder 3 and 4 demonstrates the highest overall accuracy of 99.07%



Cylinder_number	AI	PI	Overall accuracy
1	97.13	98.69	97.91
2	98.89	97.21	98.05
3	99.74	98.45	99.09
4	98.13	99.99	99.07
5	97.79	98.90	98.34
6	96.11	98.19	97.15

Figure 29: Feed Forward Neural Network accuracy

and 99.09% respectively, highlighting its exceptional predictive capability. Conversely, cylinder 6 exhibits the lowest overall accuracy of 97.15%, indicating relatively lower precision in its predictions when compared to other cylinders.

Figure 30 shows the confusion matrices for each cylinder. Cylinder 6 stands out with a higher number of miss classifications, with 8 samples misclassified in class 1 and 10 in class 2. For example, in cylinder 1: For class 1, out of 359 samples, 353 were accurately predicted while 6 were misclassified. In class 2, out of 242 samples, 232 were correctly identified with 10 false positives. Notably, class 3 had perfect predictions, with all 165 samples correctly classified and no false positives. The false positive rate and false negative rate for each class can be calculated using equation 7 and equation 8.

Class 1:

$$\text{Accuracy} = \frac{353+400}{353+6+7+400} = \frac{753}{766} \approx 0.9829 \quad \text{(or 98.29\%)}$$
$$\text{FNR} = \frac{6}{6+353} = \frac{6}{359} \approx 0.0167 \quad \text{(or 1.67\%)}$$
$$\text{FPR} = \frac{7}{7+400} = \frac{7}{407} \approx 0.0172 \quad \text{(or 1.72\%)}$$

Class 2:

$$\text{Accuracy} = \frac{232+521}{232+10+3+521} = \frac{753}{766} \approx 0.9829 \quad \text{(or 98.29\%)}$$
$$\text{FNR} = \frac{10}{10+232} = \frac{10}{242} \approx 0.0413 \quad \text{(or 4.13\%)}$$
$$\text{FPR} = \frac{3}{3+521} = \frac{3}{524} \approx 0.0057 \quad \text{(or 0.57\%)}$$

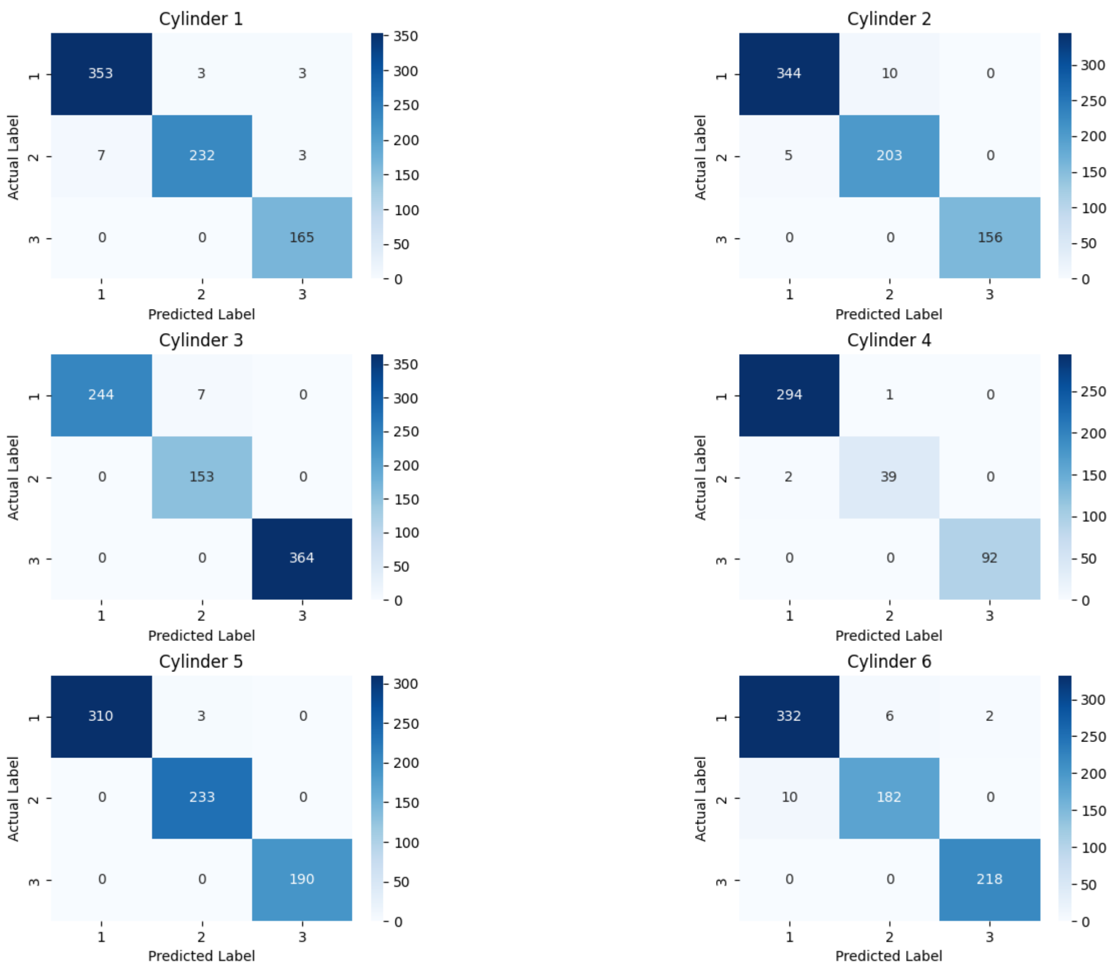


Figure 30: Feed Forward Neural Network Confusion Matrices

Class 3:

$$\text{Accuracy} = \frac{165+595}{165+0+6+595} = \frac{760}{766} \approx 0.9922 \quad \text{(or 99.22\%)}$$

$$\text{FNR} = \frac{0}{0+165} = 0 \quad \text{(or 0\%)}$$

$$\text{FPR} = \frac{6}{6+595} = \frac{6}{601} \approx 0.0100 \quad \text{(or 1.00\%)}$$

6.2 Matlab Results

In the Matlab implementation, outcomes are stored in the workspace using the scope settings, incorporating a timestamped structure. Subsequently, the results are exported from Matlab and is done in Python environment. Here, the performance is assessed by testing the predictions against the ground truth using the test data, and the confusion matrix for each cylinder is visualized. Remarkably, the confusion matrices obtained from Matlab mirror the results obtained in Python. Hence, the performance achieved in Simulink closely aligns with that achieved in Python.

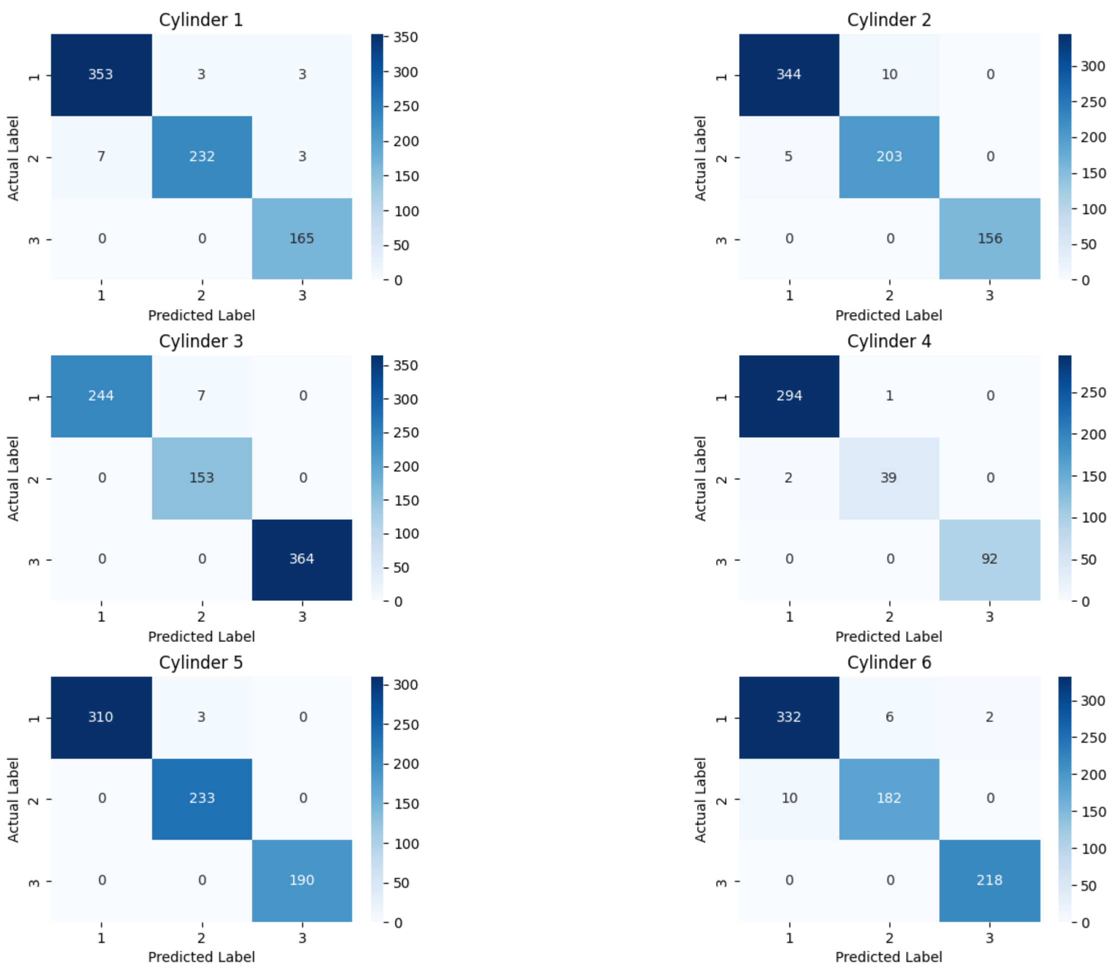
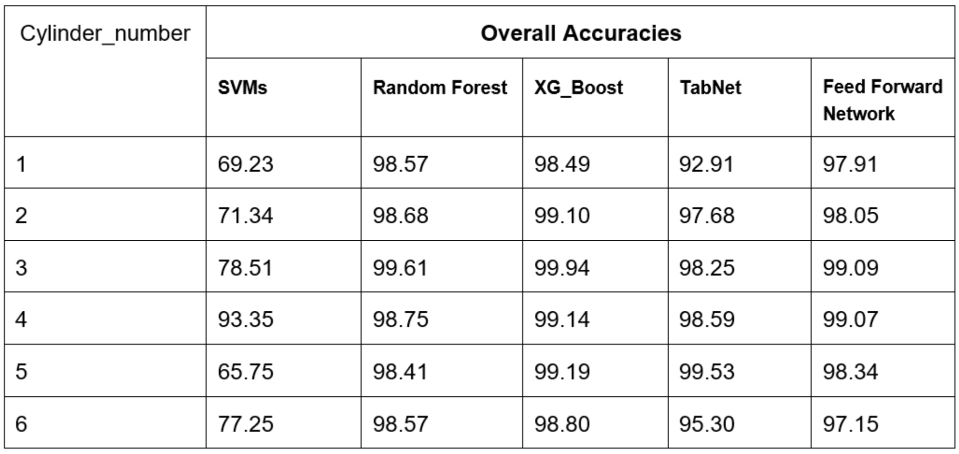


Figure 31: Confusion matrices for Simulink prediction

In Figure 31, the confusion matrices generated by Simulink predictions are observed alongside the ground truth test data from Python. The operation of plotting these matrices was conducted in Python due to its ease of use compared to Matlab scripting. Let's delve into the details focusing on cylinder 1: There are 359 samples related to class 1, 242 to class 2, and 165 to class 3. The diagonal elements represent correct predictions, while off-diagonal elements indicate miss classifications. For instance, in class 1, out of 359 samples, 353 were correctly predicted while 6 were misclassified. Among these misclassified samples, 3 were classified as class 2 and the remaining 3 as class 3. Notably, for the critical class 3, all samples were correctly classified without any miss classifications. However, cylinders 1 and 6 exhibit some miss classification concerning class 1, where a few samples, despite belonging to class 1, were mistakenly classified as class 2 or class 3. False positives were taken into account, calculated using the provided d and no false positives. The false positive rate and false negative rate for each class can be calculated using equation 7 and equation 8, resulting in a false positive percentage of less than 1% for class 3.

6.3 Discussion

The Figure 32 summarizes the performance of various machine learning models across six cylinders. In Cylinder 1, the Random Forest model stands out with the highest overall accuracy, followed by XGBoost and then the Feed Forward Network, while SVMs and TabNet have lower accuracy. Cylinder 2 sees XGBoost as the top performer, followed by Random Forest and Feed Forward Neural Network, with SVMs and TabNet again showing lower accuracy. In Cylinder 3, XGBoost continues to dominate, followed closely by Feed Forward Neural Network and Random Forest, with SVMs and TabNet maintaining lower accuracy. In Cylinder 4, XGBoost leads in accuracy, with Feed Forward Neural Network and Random Forest close behind, while SVMs and TabNet show comparatively lower accuracy. Cylinder 5 highlights TabNet with the highest accuracy, followed by XGBoost and Feed-Forward Neural Network, while Random Forest and SVMs demonstrate lower accuracy. Lastly, in Cylinder 6, XGBoost maintains its dominance, followed by Feed Forward Neural Network and Random Forest, with TabNet and SVMs exhibiting the lowest accuracy in this category. In general, XGBoost consistently performs well across multiple cylinders, while Feed Forward Neural Network and Random Forests also showcase strong overall accuracy. Comparatively to the other algorithms tested, SVM and TabNet models tend to have a lower accuracy.

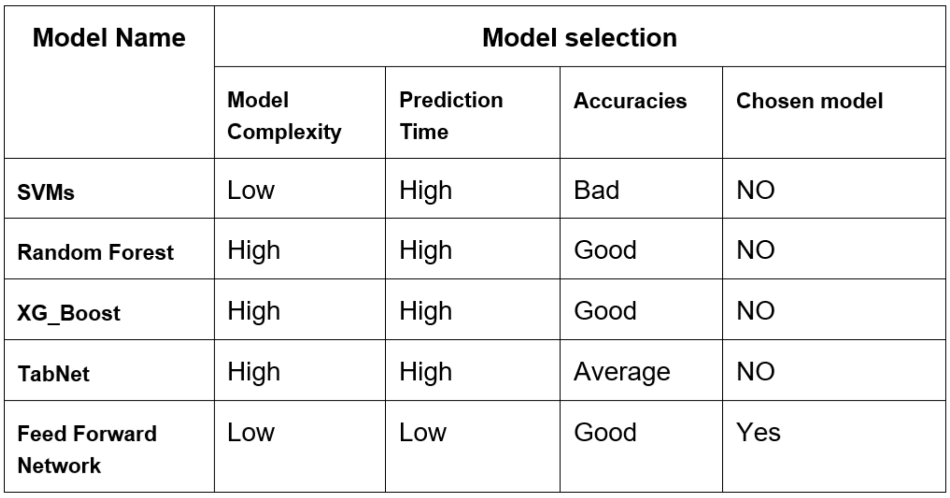


Cylinder_number	**Overall Accuracies**				
	SVMs	**Random Forest**	**XG_Boost**	**TabNet**	**Feed Forward Network**
1	69.23	98.57	98.49	92.91	97.91
2	71.34	98.68	99.10	97.68	98.05
3	78.51	99.61	99.94	98.25	99.09
4	93.35	98.75	99.14	98.59	99.07
5	65.75	98.41	99.19	99.53	98.34
6	77.25	98.57	98.80	95.30	97.15

Figure 32: Overall Accuracy’s for all Models

According to the above discussion, Random Forest, XGBoost, and Feed-Forward Neural Network managed to achieve the best results across all cylinders. To implement the trained model for prediction in Simulink, several key considerations were taken into account when selecting the Feed Forward Neural Network. This model was chosen due to its comparable accuracy with other models. In MATLAB, implementing machine learning methods directly requires an additional license. However, constructing neural networks manually is straightforward and does not require additional packages.

The Figure 33 presents a comprehensive overview of machine learning model performance and selection criteria across cylinders. The "Model Name" column identifies the machine learning models utilized, encompassing SVMs (Support Vector Machines), Random Forest, XGBoost, TabNet, and Feed Forward Neural Network, each bringing its unique algorithmic framework and approach to the predictive task. "Model Complexity" likely denotes the intricacy of each model, possibly in terms of the number of features or variables considered during predictions, with higher complexity models potentially offering increased accuracy but demanding greater computational resources as our idea is to implement it without licensed packages in MATLAB. As [50], stated that in data science filed python would be better than Matlab.



Model Name	Model selection			
	Model Complexity	Prediction Time	Accuracies	Chosen model
SVMs	Low	High	Bad	NO
Random Forest	High	High	Good	NO
XG_Boost	High	High	Good	NO
TabNet	High	High	Average	NO
Feed Forward Network	Low	Low	Good	Yes

Figure 33: Comparison of ML Techniques for Model Selection

The "Prediction Accuracy" column showcases the overall accuracy achieved by each model in predicting outcomes for the cylinders, ranging from 86% to 99% from bad to good, if the accuracy less than 92%, it is considered as bad, average between 92% and 98%, greater than 98 are considered good. accuracy indicating generally high predictive performance across all models and cylinders. The "Chosen Model" column delineates whether a particular model was selected (Yes or No), for further implementation in Simulink, considering that in future work this Simulink model will be integrated with AVL SW and deployed for validation in an ECU.

Potential reasons for favoring a model could include its achieved accuracy, balance between accuracy, complexity. The prediction time in Python is high, taking seconds, which is not suitable for considering the model for constructing prediction model in Simulink, where outcomes need to be predicted within milliseconds. Constructing a Feed Forward Neural Network (FFN) with two hidden layers without hyper parameter tuning is generally less complex than manually constructing a Random Forest (RF) or XGBoost. This is because a FFN has a straightforward architecture with input, hidden, and output layers, requiring minimal steps to define and implement. The code for an FFN can be concise, using basic matrix operations and activation functions. In contrast, constructing a Random Forest involves creating and managing multiple decision trees, ensuring they work together in an ensemble, and handling bootstrap sampling for training subsets. Similarly, building an XGBoost model requires constructing trees sequentially, managing boosting parameters, and handling advanced features like regularization and missing value handling, all of which add complexity compared to a simple FFN. Therefore, for a basic neural network architecture, the FFN is generally simpler to construct manually compared to RF or XGBoost. Considering all the factors in to account, a Feed Forward Neural Network is chosen and further implemented in Simulink. After integrating the Feed Forward Neural Network into Simulink, the outcomes precisely matched in accuracy and were delivered within the designated prediction time frame. This was accomplished without encountering any errors or inconsistencies.

7 Conclusion & Future Work

7.1 Conclusion

In conclusion, experimentation with multiple models in machine learning, such as multi-class SVMs with multi-output classifiers, random forest classifiers with multi-output classifiers, extreme gradient boosting classifiers with multi-output classifiers, and deep learning models like TabNet and feed forward network approaches for classifying anomalies, has yielded valuable insights in the realm of H2 engine care. The challenge of detecting even the tiniest deviations within combustion chambers, complicated by sensor data intricacies involving good signal features and anomalies, prompted the exploration of advanced neural network architectures and deep learning methodologies.

The primary aim of the implementation was to address the prevalent issue of anomaly detection in H2 engine maintenance. Utilizing a deep learning feed forward neural network, a robust pipeline was successfully established to detect signal deviations compared with the ground truth. This process included solving the imbalance problems caused by data deficits. This led to generating synthesized data, where a deep learning model is trained with both authentic ground truth signals and deviated signals, learning the complex patterns between them. Validation was performed with unseen data to evaluate how well the model can identify anomalies and good signals, thereby improving anomaly detection within the combustion chamber of H2 engines.

Specifically, the outcomes of the implementation highlighted the superior effectiveness of the decision tree classifiers approach in closely replicating the accuracies of feed forward neural networks compared to the TabNet and multi-class SVMs with multi-output classifier approaches. The feed forward neural network demonstrated advanced capabilities in efficiently capturing the intricate features of combustion chambers, showcasing its reduced prediction time 6.3. The architecture of the feed forward neural network is simple and fast compared to other approaches, which are the reasons for the model's success in classifying anomaly signals with high quality.

In practical terms, implementation contributes to the refinement of data-driven methodologies in safety-critical industries, particularly H2 engine maintenance. The implementation of all approaches provides a comparative understanding of their efficacy in detecting the anomalies in combustion chamber. This research outcome holds promise for future applications, offering innovative solutions to enhance automobile maintenance practices through the utilization of deep learning and advanced training techniques.

7.2 Future Work

As the field of anomaly detection for automotive maintenance advances, the trajectory of future research directions will be oriented to push it beyond its current limitations.

Building upon the advancements achieved, this forward-looking exploration is poised to refine and expand the capabilities of machine learning models. The optimization of machine learning model architectures, specifically tailored for anomaly detection, stands as a pivotal objective. By delving into modifications or novel approaches within the machine learning frameworks, the aim is to reduce false positives rates and false negative rates and increase the accuracies [1].

Advanced Machine Learning Algorithms: Developing and refining machine learning algorithms tailored for anomaly detection in H2 engines can significantly enhance detection accuracy and response times. Deep learning techniques, like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) [44] , can be employed to analyze vast amounts of sensor data and identify subtle patterns indicative of anomalies.

Real-time Monitoring Systems: Integrating real-time data acquisition and processing systems with advanced anomaly detection algorithms will enable continuous monitoring of H2 engines. This can facilitate immediate identification and mitigation of potential issues.

Sensor Fusion and Data Integration: Utilizing data from multiple sensors, such as temperature, pressure, and knocking sensors, and integrating this information using sensor fusion techniques can improve the robustness and reliability of anomaly detection systems. As a result, an extensive understanding of the engine's operating conditions is gained.

Edge Computing: Deploying edge computing solutions for anomaly detection in H2 engines allows for data processing [61]. This reduces latency and enhances the system's ability to make real-time decisions. Edge devices equipped with machine learning models can analyze data locally and send alerts or take corrective actions immediately.

Hybrid Models: Combining physics-based models with data-driven approaches can improve the accuracy and reliability of anomaly detection systems. Physics-based models can provide a fundamental understanding of engine behavior, while data-driven models can learn from historical data to detect anomalies more effectively.

By pursuing these research directions, the field of anomaly detection in hydrogen engines can advance significantly, contributing to the development of safer, more efficient, and more reliable hydrogen propulsion systems.

7.2.1 Future Scope in Medical Sector

The future of AI in the medical sector is promising, especially with the integration of advanced anomaly detection techniques. Anomalies detection in Biomedical signals is critical for early diagnosis and treatment planning. This subsection explores the leveraging of machine learning and deep learning techniques used for anomaly detection in domain of medical signal processing, originally designed for hydrogen combustion engine signals. Through a transfer learning approach, the effectiveness of this methodology in detecting abnormalities in various medical signals is demonstrated, offering potential advancements in diagnostic accuracy and patient care. Here are some potential areas where AI-driven anomaly detection can be used in healthcare:

1. Early Diagnosis of the patient: AI-driven anomaly detection techniques can significantly improve screening programs for various diseases like breast cancer, lung cancer, Cholesterol and Osteoporosis. For instance, the constructed models could analyze a chest X-ray to detect not only lung cancer but also other respiratory conditions such as tuberculosis, pneumonia, and chronic obstructive pulmonary disease (COPD) [53]. By providing a comprehensive analysis in one go, these algorithms enhance the efficiency and effectiveness of screening programs.

2. Disease Prediction: Machine learning methods can helpful in detecting the diseases. For instance, the level Of activity of patients with stroke is helpful in detecting diseases by enhancing the prediction of stroke recovery outcomes [9]. By using multi class multi output algorithms, study effectively can classifies patients' activity levels, which can inform personalized rehabilitation plans. This approach can be adapted to other diseases, providing a framework for predicting patient outcomes and improving early intervention and treatment strategies in various medical conditions.

3. Real-Time Monitoring and Alerts: These models can also continuously track vital signs, analyzing various physiological data streams simultaneously and other health indicators. These Multi-task models involves processing multiple types of signals, such as ECG, EEG, and blood pressure, to detect abnormalities and predict potential health issues. By integrating and interpreting these signals in real time, healthcare providers can receive immediate alerts about critical changes in a patient's condition, enabling faster and more accurate interventions. These systems improves patient outcomes, optimizes resource use, and provides continuous, comprehensive monitoring in diverse healthcare settings.

4. Predictive Maintenance of Medical Equipment: Multi-task models are advantageous for predictive maintenance of medical equipment using signal data due to their ability to extract relevant features across diverse sensor inputs, improving computational efficiency and resource management [16]. They enhance predictive accuracy by leveraging relationships between different tasks, adapt well to new signals and conditions, and provide regularization benefits for robust generalization. Additionally, these models can offer interpretable insights into how various signals contribute to equipment failure predictions, crucial for informed decision-making in healthcare settings.

5. Detecting Abnormalities in Bio Medical Signals: AI models demonstrate proficiency in extracting pertinent characteristics from various types of biomedical data like ECG, EEG, or EMG signals, thereby enhancing efficiency and optimizing resource allocation [43]. By jointly learning to predict various abnormalities, these models enhance accuracy through shared representations and task relationships, adapting well to new signal types or changing conditions. They provide regularization, ensuring robust generalization, and offer interpretable insights into the complex interactions

within biomedical signals, crucial for accurate diagnosis and proactive healthcare management.

References

[1] AI Frameworks. https://harvard-edge.github.io/cs249r_book/contents/frameworks/frameworks.html

[2] ACTIVELOOP: *Adaptive Synthetic Sampling (ADASYN)*. https://www.activeloop.ai/resources/glossary/adaptive-synthetic-sampling-adasyn/

[3] AHMED, Shams F. ; ALAM, Md. Sakib B. ; HASSAN, Maruf ; ROZBU, Mahtabin R. ; ISHTIAK, Taoseef ; RAFA, Nazifa ; MOFIJUR, M. ; SHAWKAT ALI, A. B. M. ; GANDOMI, Amir H.: Deep learning modelling techniques: current progress, applications, advantages, and challenges. In: *Artificial Intelligence Review* 56 (2023), Nov, Nr. 11, 13521-13617. http://dx.doi.org/10.1007/s10462-023-10466-8. – DOI 10.1007/s10462–023–10466–8. – ISSN 1573–7462

[4] AL-SULAIMAN, Talal: Predicting reactions to anomalies in stock movements using a feed-forward deep learning network. In: *International Journal of Information Management Data Insights* 2 (2022), Nr. 1, 100071. http://dx.doi.org/https://doi.org/10.1016/j.jjimei.2022.100071. – DOI https://doi.org/10.1016/j.jjimei.2022.100071. – ISSN 2667–0968

[5] ALABADI, Montdher ; CELIK, Yuksel: Anomaly Detection for Cyber-Security Based on Convolution Neural Network : A survey, 2020, S. 1–14

[6] ALTERYX: *Feature Engineering*. https://www.alteryx.com/glossary/feature-engineering

[7] ALZUBAIDI, Laith ; ZHANG, Jinglan ; HUMAIDI, Amjad J. ; AL-DUJAILI, Ayad ; DUAN, Ye ; AL-SHAMMA, Omran ; SANTAMARÍA, J. ; FADHEL, Mohammed A. ; AL-AMIDIE, Muthana ; FARHAN, Laith: Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. In: *Journal of Big Data* 8 (2021), Mar, Nr. 1, 53. http://dx.doi.org/10.1186/s40537-021-00444-8. – DOI 10.1186/s40537–021–00444–8. – ISSN 2196–1115

[8] ANDRYSIAK, Tomasz: Machine Learning Techniques Applied to Data Analysis and Anomaly Detection in ECG Signals. In: *Applied Artificial Intelligence* 30 (2016), Nr. 6, 610-634. http://dx.doi.org/10.1080/08839514.2016.1193720. – DOI 10.1080/08839514.2016.1193720

[9] APAO, Norma J. ; FELISCUZO, Larmie S. ; ROMANA, Cherry Lyn C. S. ; TAGARO, J.: Multiclass Classification Using Random Forest Algorithm To Prognosticate The Level Of Activity Of Patients With Stroke. In: *International Journal of Scientific & Technology Research* 9 (2020), 1233-1240. https://api.semanticscholar.org/CorpusID:216654256

[10] ARIK, Sercan O. ; PFISTER, Tomas: *TabNet: Attentive Interpretable Tabular Learning*. 2020

[11] BAELDUNG: *Adaptive Synthetic Sampling (ADASYN)*. https://www.baeldung.com/cs/svm-multiclass-classification. Version: March 18, 2024

[12] BIGDATA: Pros And Cons Of Feature Engineering. (27 April 2021). https://bigdataanalyticsnews.com/pros-cons-of-feature-engineering/

[13] BROWNLEE, Jason: Feature Importance and Feature Selection With XGBoost in Python. (August 27, 2020). https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/

[14] BROWNLEE, Jason: A Gentle Introduction to XGBoost for Applied Machine Learning. (February 17, 2021). https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[15] BROWNLEE, Jason: Random Oversampling and Undersampling for Imbalanced Classification. (January 5, 2021). https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

[16] CANDA, Jam: *Machine Learning Techniques for Predictive Maintenance*. https://medium.com/@jam.canda/machine-learning-techniques-for-predictive-maintenance-662d056e7f08

[17] CRAMMER, Koby ; SINGER, Yoram: On the algorithmic implementation of multiclass kernel-based vector machines. In: *J. Mach. Learn. Res.* 2 (2002), mar, S. 265–292. – ISSN 1532–4435

[18] DEEPAI: Feed Forward Neural Network. https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network

[19] DEVANSH, Devansh: What are the benefits and challenges of feature engineering for complex data sets? https://www.linkedin.com/advice/0/what-benefits-challenges-feature-engineering

[20] DLMC, xgboost d.: Introduction to Boosted Trees. (2022). https://xgboost.readthedocs.io/en/stable/tutorials/model.html

[21] DONGES, Niklas: Random Forest: A Complete Guide for Machine Learning. (Mar 08, 2024). https://builtin.com/data-science/random-forest-algorithm

[22] EVGENIOU, Theodoros ; PONTIL, Massimiliano: Regularized multi–task learning. In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA : Association for Computing Machinery, 2004 (KDD ’04). – ISBN 1581138881, 109–117

[23] EXPLORIUM: 5 Reasons Why Feature Engineering is Challenging. (AUG 06, 2023). https://www.explorium.ai/blog/machine-learning/5-reasons-why-feature-engineering-is-challenging/

[24] FANELLI, Daniele: How Many Scientists Fabricate and Falsify Research? A Systematic Review and Meta-Analysis of Survey Data. (May 29, 2009). http://dx.doi.org/https://doi.org/10.1371/journal.pone.0005738. – DOI https://doi.org/10.1371/journal.pone.0005738

[25] GALLI, Sole: Exploring Oversampling Techniques for Imbalanced. (Mar 20, 2023). https://www.blog.trainindata.com/oversampling-techniques-for-imbalanced-data/

[26] GALLI, Soledad: The Challenges of Creating Features for Machine Learning. (February 21, 2022). https://www.kdnuggets.com/2022/02/challenges-creating-features-machine-learning.html

[27] GEEKSFORGEEKS: Tabnet. (23 Feb, 2023). https://www.geeksforgeeks.org/tabnet/

[28] GRABCZEWSKI, K. ; JANKOWSKI, N.: Feature selection with decision tree criterion. In: *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, 2005, S. 6 pp.–

[29] HACHCHAM, Aymane: XGBoost: Everything You Need to Know. (11th August, 2023). https://neptune.ai/blog/xgboost-everything-you-need-to-know

[30] HEATON, JB ; POLSON, Nicholas G. ; WITTE, Jan H.: Deep learning in finance. In: *arXiv preprint arXiv:1602.06561* (2016)

[31] HEAVY.AI: *Feature Engineering*. https://www.heavy.ai/technical-glossary/feature-engineering

[32] IBM: *What is random forest?* https://www.ibm.com/topics/random-forest

[33] ILANGO, Vigneshwar: Tabnet — Deep Learning for Tabular data: Architecture Overview. (Apr 11, 2021). https://vigneshwarilango.medium.com/tabnet-deep-learning-for-tabular-data-architecture-overview-448ced8f8cfc

[34] JAADI, Zakaria: A Step-by-Step Explanation of Principal Component Analysis (PCA). https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[35] JALLAL, Mohammed A. ; SAMIRA, Chabaa ; ZEROUAL, Abdelouhab: A new artificial multi-neural approach to estimate the hourly global solar radiation in a semi-arid climate site. In: *Theoretical and Applied Climatology* 139 (2020), 02. http://dx.doi.org/10.1007/s00704-019-03033-1. – DOI 10.1007/s00704–019–03033–1

[36] JI, You ; SUN, Shiliang: Multitask Multiclass Support Vector Machines. In: *2011 IEEE 11th International Conference on Data Mining Workshops*, 2011, S. 512–518

[37] JIANG, Xuezheng ; WANG, Junyi ; MENG, Qinggang ; SAADA, Mohamad ; CAI, Haibin: An adaptive multi-class imbalanced classification framework based on ensemble methods and deep network. In: *Neural Computing and Applications* 35 (2023), May, Nr. 15, 11141-11159. http://dx.doi.org/10.1007/s00521-023-08290-w. – DOI 10.1007/s00521–023–08290–w. – ISSN 1433–3058

[38] JIHWAN: TabNet —Deep Neural Net for Tabular. (Jun 17, 2023). https://medium.com/@okpo65/paper-review-tabnet-deep-neural-net-for-tabular-a97c43290969

[39] KAMALI, Kaivan: Feedforward neural networks (FNN). (Apr 28, 2021). https://training.galaxyproject.org/training-material/topics/statistics/tutorials/FNN/tutorial.html

[40] KHUSHAKTOV, Farkhod: Introduction Random Forest Classification. (Aug 26, 2023). https://medium.com/@mrmaster907/introduction-random-forest-classification-by-example-6983d95c7b91

[41] KURAMA, Vihar: Feedforward Neural Networks: A Quick Primer for Deep Learning. (Aug 31, 2022). https://builtin.com/data-science/feedforward-neural-network-intro

[42] LENDERINK, R.J.: *Unsupervised Outlier Detection in Financial Statement Audits*. http://essay.utwente.nl/79813/. Version: September 2019

[43] LI, Chenyang ; SUN, Le ; PENG, Dandan ; SUBRAMANI, Sudha ; NICOLAS, Shangwe C.: A multi-label classification system for anomaly classification in electrocardiogram. In: *Health Inf. Sci. Syst.* 10 (2022), Dezember, Nr. 1, S. 19

[44] LIU, Shiya ; LIU, Lingjia ; YI, Yang: Quantized Reservoir Computing on Edge Devices for Communication Applications. In: *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, 2020, S. 445–449

[45] LOWE, ANDREA: Manual Feature Engineering. (2019-08-20). https://domino.ai/blog/manual-feature-engineering

[46] MASAELI, Mahdokht ; FUNG, Glenn ; DY, Jennifer G.: From Transformation-Based Dimensionality Reduction to Feature Selection. In: *International Conference on Machine Learning*, 2010

[47] MONDAL, Debmalya: Imbalanced data classification: Oversampling and Undersampling. (Feb 6, 2023). https://medium.com/@debspeaks/imbalanced-data-classification-oversampling-and-undersampling-297ba21fbd7c

[48] PYKES, Kurtis: Oversampling and Undersampling. (Sep 10, 2020). https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56dcf

[49] R, Vinod Kumar G.: Feature Transformation Topics in Machine Learning. (2023). https://medium.com/@vinodkumargr/5-feature-transformation-topics-in-machine-learning-930ad908148e#:~:text=Feature%20transformation%20is%20the%20process,have%20similar%20scales%20or%20distributions.

[50] RANE, Zulie: There's a Clear Winner Between Matlab vs Python. (Oct 1, 2021). https://towardsdatascience.com/theres-a-clear-winner-between-matlab-vs-python-f6bb56b2b930#75a7

[51] SAINI, Anshul: What is Decision Tree? (18 Apr, 2024). https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/

[52] SALAUDDIN KHAN, Md ; NATH, Tushar D. ; MURAD HOSSAIN, Md ; MUKHERJEE, Arnab ; BIN HASNATH, Hafiz ; MANHAZ MEEM, Tahera ; KHAN, Umama: Comparison of multiclass classification techniques using dry bean dataset. In: *International Journal of Cognitive Computing in Engineering* 4 (2023), 6-20. http://dx.doi.org/https://doi.org/10.1016/j.ijcce.2023.01.002. – DOI https://doi.org/10.1016/j.ijcce.2023.01.002. – ISSN 2666–3074

[53] SÁNCHEZ-GUTIÉRREZ, Máximo E. ; GONZÁLEZ-PÉREZ, Pedro P.: Multi-class classification of medical data based on neural network pruning and information-entropy measures. In: *Entropy (Basel)* 24 (2022), Januar, Nr. 2, S. 196

[54] SHARMA, Pranshu: Introduction to Feed-Forward Neural Network in Deep Learning. (08 Feb, 2024). https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-feed-forward-network-in-deep-learning/

[55] SHIN, Terence: *Understanding Feature Importance in Machine Learning*. https://builtin.com/data-science/feature-importance. Version: Nov. 07, 2023

[56] SIMPLILEARN: What is XGBoost? An Introduction to XGBoost Algorithm in Machine Learning. (Nov 7, 2023). https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article

[57] TEAM, Wallstreetmojo: Feature Engineering, January 5, 2024

[58] THEISSLER, Andreas: Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection. In: *Knowledge-Based Systems* 123 (2017), 163-173. http://dx.doi.org/https://doi.org/10.1016/j.knosys.2017.02.023. – DOI https://doi.org/10.1016/j.knosys.2017.02.023. – ISSN 0950–7051

[59] THUDUMU, S. et a.: A comprehensive survey of anomaly detection techniques for high dimensional big data. (2020). http://dx.doi.org/10.1186/s40537-020-00320-x. – DOI 10.1186/s40537–020–00320–x

[60] TUTORIALSPOINT: *Importance of Feature Engineering in Model Building*. https://www.tutorialspoint.com/importance-of-feature-engineering-in-model-building. Version: Nov. 07, 2023

[61] YEUNG, Tiffany: What Is Edge Computing? (Oct 22, 2019). https://blogs.nvidia.com/blog/what-is-edge-computing/

[62] ZENG, Y et a. Peng: A new deep belief network-based multi-task learning for diagnosis of Alzheimer's disease. (2023). http://dx.doi.org/doi.org/10.1007/s00521-021-06149-6. – DOI doi.org/10.1007/s00521–021–06149–6

[63] ZHANG, Liang ; CHI, Mingmin ; GUO, Jiankui: MTForest: Ensemble Decision Trees based on Multi-Task Learning, 2008, S. 122–126

[64] ZHANG, Licheng ; ZHAN, Cheng: Machine learning in rock facies classification: An application of XGBoost. In: *International Geophysical Conference, Qingdao, China, 17-20 April 2017* Society of Exploration Geophysicists and Chinese Petroleum Society, 2017, S. 1371–1374

Declaration

I hereby certify that I have written this thesis independently and that I have not used any sources or aids other than those indicated, that all passages of the work which have been taken over verbatim or in spirit from other sources from other sources have been marked as such and that the work has not yet been has not yet been submitted to any examination authority in the same or a similar form.

Erlangen, June 28, 2024

T. Arun Sai

my signature