**New GasPipeline Multiclass Technical Report Response**

**Date**: Jan 22, 2025

**Tech Report Filename**: Multiclass\_GasPipeline\_Tech\_Report\_Response.docx

This page is to be printed on the back of the cover page. This text is hidden so it will not print.

Table of Contents

[1 Introduction 6](#_Toc192164927)

[2 Research Objectives 8](#_Toc192164928)

[3 Uncertainties and Research Challenges 9](#_Toc192164929)

[4 SCADA Architecture 10](#_Toc192164930)

[4.1 Sensor/Actuator: 11](#_Toc192164931)

[4.2 PLC/RTU/IED: 11](#_Toc192164932)

[4.3 SCADA Gateways: 12](#_Toc192164933)

[4.4 SCADA Server - MTU 12](#_Toc192164934)

[4.5 Data Historian: 12](#_Toc192164935)

[4.6 HMI: 13](#_Toc192164936)

[5 Gas Pipeline SCADA Testbed and Dataset 13](#_Toc192164937)

[5.1.1 Data Preprocessing and Exploratory Data Analysis (EDA) on Gas Pipeline SCADA System Dataset 18](#_Toc192164938)

[6 Methodology/Approach 20](#_Toc192164939)

[6.1 Feature Ranking 21](#_Toc192164940)

[6.1.1 Statistical Properties of Dataset (SP) 22](#_Toc192164941)

[6.1.2 WFI Using Random Forest 24](#_Toc192164942)

[6.1.3 WFI Using XGboost 25](#_Toc192164943)

[6.1.4 Feature Reduction Analysis 26](#_Toc192164944)

[6.2 Selective Feature Selection (SPFS) 27](#_Toc192164945)

[6.3 Recursive Feature Elimination (RFE) 28](#_Toc192164946)

[6.4 Cross-Validation 30](#_Toc192164947)

[6.5 Metrics Evaluation 30](#_Toc192164948)

[6.6 Averaging Methods for Metrics Evaluation 31](#_Toc192164949)

[6.6.1 Micro Averaging 31](#_Toc192164950)

[6.6.2 Macro Averaging 32](#_Toc192164951)

[6.6.3 Weighted Averaging 33](#_Toc192164952)

[6.6.4 Evaluation of Averaging Methods 33](#_Toc192164953)

[7 Design/Framework/Testbed 34](#_Toc192164954)

[8 Results and Discussion 36](#_Toc192164955)

[8.1 Overview of Result Metrics 36](#_Toc192164956)

[8.1.1 Key Performance Metrics 38](#_Toc192164957)

[8.1.2 Performance Trends 38](#_Toc192164958)

[8.1.3 Comparison by Model Type, Ranking Method, and Feature Selection 39](#_Toc192164959)

[8.2 Ranking Results 40](#_Toc192164960)

[8.2.1 Statistical Properties Ranking Results 40](#_Toc192164961)

[8.2.2 Weighted feature selection ranking results using RandomForest Classifier 42](#_Toc192164962)

[8.2.3 Weighted feature selection ranking results using XGBoost Classifier 44](#_Toc192164963)

[8.3 Summary of Best-Performing Models 46](#_Toc192164964)

[8.3.1 Best F1 Scores 46](#_Toc192164965)

[8.3.2 Best accuracy 47](#_Toc192164966)

[8.3.3 Performance of Models with Limited Features 51](#_Toc192164967)

[8.3.4 Performance on time taken 54](#_Toc192164968)

[8.4 Summary of Findings 56](#_Toc192164969)

[8.4.1 Key Observations 58](#_Toc192164970)

[8.4.2 Summary of Best-Performing Configurations 58](#_Toc192164971)

[9 Conclusion 60](#_Toc192164972)

[10 Scope of Future Researche 62](#_Toc192164973)

[11 References 62](#_Toc192164974)

[Appendix A: xxx 63](#_Toc192164975)

List of Figures

[Figure 1 SCADA System Architecture [13] 11](#_Toc193097214)

[Figure 2 Gas pipeline SCADA testbed [17] 15](#_Toc193097215)

[Figure 3 Label Distribution in Response Dataset 20](#_Toc193097216)

[Figure 4 Flowchart of methodology 22](#_Toc193097217)

[Figure 5 Flowchart of SPFS process 29](#_Toc193097218)

[Figure 6 Flowchart of RFE 30](#_Toc193097219)

[Figure 7 Flowchart of model evaluation process 36](#_Toc193097220)

[Figure 8 Impact of Feature Reduction on Model Performance (using best of SP and Micro Averaging) 38](#_Toc193097221)

[Figure 9 Graph showing all combinations best F1-Score 59](#_Toc193097222)

[Figure 10 Content Guidelines 86](#_Toc193097223)

List of Tables

[Table 1 Description of features from New Gas Pipeline dataset 16](#_Toc193097225)

[Table 2 Example Feature Ranking based on metrics from the Function dataset 24](#_Toc193097226)

[Table 3 Feature importance ranking using Random Forest. 25](#_Toc193097227)

[Table 4 Feature importance ranking using XGBoost. 26](#_Toc193097228)

[Table 5 Ranking features in original dataset based on statistical properties 41](#_Toc193097229)

[Table 6 Feature importance rankings for the function dataset using the RandomForest classifier 43](#_Toc193097230)

[Table 7 Feature importance rankings for the Response dataset using the XGBoost classifier 45](#_Toc193097231)

[Table 8 Top 30 best preforming combinations based on greatest F1 score 47](#_Toc193097232)

[Table 9 Best 30 accuracy scores (Micro Averaging) 49](#_Toc193097233)

[Table 10 Best preforming combinations when reducing the number of features to 9 or less (Micro Average) 52](#_Toc193097234)

[Table 11 Ranking of Model Configurations by Total Computational Time (Best 13 F1 Scores) 54](#_Toc193097235)

[Table 12 Time comparison between RF and XGBoost 56](#_Toc193097236)

[*Table 13 Full Evaluation Log csv file* 77](#_Toc193097237)

[Table 14: Formatting Guideline 84](#_Toc193097238)

# Introduction

Supervisory Control and Data Acquisition (SCADA) systems are integral to the operation of critical infrastructure, including utilities like water, electricity, gas, and transportation systems. These systems provide real-time monitoring and control of industrial processes, ensuring efficient and safe operations. SCADA systems gather data from sensors and devices in the field, process this data, and present it to human operators in a comprehensible format, enabling them to make informed decisions and respond swiftly to any issues.

SCADA systems play a crucial role in maintaining the stability and reliability of critical infrastructure. For instance, in the energy sector, SCADA systems help manage the distribution of electricity by monitoring the grid's performance and detecting faults before they escalate into major outages. In the water supply industry, SCADA ensures the quality and availability of water by controlling pumps, valves, and chemical treatment processes. The transportation sector relies on SCADA for the smooth operation of traffic lights, railway signals, and other essential systems.

Given the vital functions that SCADA systems perform, any disruption can have significant repercussions. Ensuring the security and integrity of SCADA systems is paramount to safeguarding the continuous and reliable operation of critical infrastructure.

Vulnerabilities of SCADA Systems

Despite their importance, SCADA systems are inherently vulnerable to cyber threats [1]-[11]. Originally designed for isolated and controlled environments, many SCADA systems were not built with robust security features to counter modern cyber threats. The increasing interconnectivity and integration of SCADA with corporate IT networks and the internet have exposed these systems to a broader range of attack vectors.

One notable example of a SCADA-related cyber incident is the attack on the Colonial Pipeline in May 2021 [6]. The ransomware attack forced the pipeline operator to shut down its operations, leading to significant fuel supply disruptions across the southeastern United States. This incident highlighted the susceptibility of SCADA systems to cyberattacks and the potential for such breaches to cause widespread economic and social disruption.

Other examples of SCADA system breaches include the 2015 cyberattack on Ukraine's power grid [7,8], which resulted in widespread power outages, and the 2010 Stuxnet worm, which targeted Iran's nuclear facilities. These incidents underscore the critical need for enhanced security measures to protect SCADA systems from malicious actors.

Machine Learning and Deep Learning for Anomaly and Attack Detection

The increasing complexity and frequency of cyber threats necessitate advanced solutions for detecting and mitigating attacks on SCADA systems. Machine learning (ML) and deep learning (DL) offer promising approaches to enhance the security of these systems by identifying anomalies and potential attacks in real-time [12-17].

Machine learning algorithms can analyze vast amounts of data generated by SCADA systems to establish baseline behaviors and detect deviations that may indicate cyber threats. These algorithms can be trained to recognize patterns associated with normal operations and flag any unusual activities for further investigation. For example, ML-based systems can detect abnormal network traffic patterns, unauthorized access attempts, or unexpected changes in system configurations.

Deep learning, a subset of machine learning, leverages artificial neural networks to model complex relationships in data. Deep learning techniques can improve the accuracy and speed of anomaly detection by processing high-dimensional data and identifying subtle indicators of attacks that traditional methods might overlook. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are particularly effective in analyzing time-series data from SCADA systems, enabling the detection of sophisticated and stealthy attacks.

By integrating ML and DL-based anomaly detection systems, SCADA operators can achieve more robust and proactive security postures. These technologies can provide continuous monitoring, real-time alerts, and automated responses to potential threats, thereby reducing the risk of successful cyberattacks and minimizing their impact on critical infrastructure.

# Research Objectives

The main objective of this research study is to investigate different machine learning classification techniques, data normalization, automated feature selection, and data balancing techniques to explore and achieve robust Machine Learning based models for Intrusion Detection System (IDS) of gas pipeline SCADA system.

In this work, we focus on understanding the data and their processing techniques before modeling through various experiments. This allowed us to select appropriate data processing and feature selection methodologies as well as enable us to develop a new and efficient systematic strategy for obtaining high-performance M/DL model. We proposed in this work a Robust Strategy for Best Machine Learning Model Selection (RS-BMLMS) and validated it on Gas Pipeline dataset as provided by Mississippi State University and Oak-Ridge National Laboratory for industrial control system security research [17]. In this work, we focus on detecting two specific attacks namely Naïve Malicious Response Injection (NMRI) and Complex Malicious Response Injection (CMRI) that are related to response messages/data.

# Uncertainties and Research Challenges

The major research challenge of this work is to design and develop robust machine learning based anomaly detection and intrusion detection models that can be used for intrusion detection in gas pipeline SCADA system.  The developed models will optimize the trade-off between multiple objectives such as reduce the training time, generalize the model to detect new attacks, improve the model accuracy, and minimize the false alarms. At present most of the intrusion detection systems used in practice are signature-based. A signature-based intrusion detection system detects attacks by looking for specific patterns, such as byte sequences in SCADA application, or known malicious instruction sequences used by malware. However, it is difficult to detect new attacks, for which no pattern is available. Machine learning-based anomaly detection can be used for SCADA systems, and it is expected that it will work better than signature-based algorithms for zero-day (new) attacks.  However, the high false alarm rate often limits practitioners to use machine learning based system for intrusion detection. Also, detecting new attack requires the model to be generalized which in turn requires the model to be trained with huge volume of data. Although these models often show excellent performance on the data population they are trained with, these models may not work well with new data.

The uncertainty remains whether the model training can be done efficiently, and model can be generalized to work on new data with high accuracy and low false alarm rate.

Towards overcoming the above uncertainties and challenges, different approaches can be taken such as using ensemble models, automated feature selection techniques and optimizing hyper-parameters. It is expected that these models will be suitable for off-line periodic training and on-line intrusion detection in gas pipeline SCADA system.

The data preprocessing and feature selection impact the model performance significantly. Therefore, in this study we focus on good model development and best model selection strategies by performing a series of experiments as will be discussed in details in subsequent sections.

# SCADA Architecture

A typical SCADA system consists of SCADA Master Terminal Unit (MTU) that is connected to either a remote location unit or connected locally to a local location unit that contains controllers, remote terminal units (RTU) or Programmable Logic Controllers (PLC) at one end and other end is connected to the control system consist of Human Machine Interface (HMI), data historian & workstation. Figure 1 below shows a simple SCADA system architecture [9]. In the following we briefly describe the main components of the SCADA system.

A screenshot of a cell phone

Description automatically generated

Figure 1 SCADA System Architecture [13]

## Sensor/Actuator:

Sensors within industrial field level check whether parts are present, size of the part, color of the part, and whether the product is full or empty, good or bad. Sensors are also used to ensure safety of the equipment, product and operators. A common classification of sensors is contact and non-contact. Another way to classify sensors is by analog and digital.

## PLC/RTU/IED:

SCADA system RTUs are microprocessor-based devices designed to monitor and collect data as well as perform some control functions. These functions depend upon the requirements of the specific industry and the technology used by the RTU manufacturer. The industry has developed smart RTUs with the capability to execute logic and PID closed loop control. The SCADA market has moved away from the traditional proprietary RTUs and toward multi-purpose devices, such as programmable logic controllers (PLCs) and intelligent electronic device (IEDs). However, traditional, purpose-built RTUs are still likely to be used in environmentally challenging applications, such as where extremes of moisture, temperature, or humidity are present, or low power consumption is required [14].

IEC 61131-3, international standard is used to define PLC/RTU programming languages and concepts. Programming languages are divided into two main sections to represent the five main programming languages. Generally, ladder logic and block diagram are the two common methods for the PLC programming.

## SCADA Gateways:

Different devices at field site, uses different types of protocols (e.g. Modbus, serial, fieldbus, dnp3 etc.) to communicate amongst each-other. SCADA gateways are used to connect different protocols to an IP network via ethernet, GSM, Wi-Fi, Zigbee, LoRa, GPRS etc. These devices might not be air-gapped, as they are having antennas, and they are interconnected by different wireless protocols [10]. Generally, these devices are soft target devices to intercept and get into the network by firmware analysis and reverse engineering methodology.

## SCADA Server - MTU

Master Terminal Unit (MTU) is located at the control center. MTU acts as heart of the SCADA system. It issues the commands to the field devices like PLCs/RTUs which are located at remote places from the control center to gather the required data from the plant floor. Furthermore, it processes the information, stores the important status information at data historian and display the information in the form of graphs, curves and tables on HMI to help in taking control decisions. Moreover, the communication between MTU and field control devices are only initiated by the program resides in the MTU. Hence, MTU act as a Master and RTU is the slave, however the level of communication is considered as peer-to-peer communication.

## Data Historian:

A Data Historian (also known as a Process Historian or Operational Historian) is a software program that records and retrieves production and process data by time; it stores the information in a time series database that can efficiently store data with minimal disk space and fast retrieval [16]. Time series information is often displayed in a trend or as tabular data over a time range (ex. the last day, last 8 hours, last year).

## HMI:

An HMI is a software application that presents information to the users about the state of the process and to accept the user and control instructions. HMI provides a graphical representation of the control system process and provides real time data acquisition. It monitors data and makes it available to users, as well as provides an interface for inputs from the users. One of the advantages that HMI provides is that they provide an interface between the user and the machine within the factory floor, a centralized control unit and a local interface of the process for quick monitoring and adjusting.

# Gas Pipeline SCADA Testbed and Dataset

Mississippi State University’s in-house SCADA lab has a gas pipeline system to collect data for cyber attack research as shown in Figure 2. The system consists of three major components: sensors and actuators, a communication network, and supervisory control as shown in the picture below [17].   This section summarizes the testbed components and the methods by which the dataset was collected using this testbed.  Detailed information can be found in [17].

The gas pipeline has two actuators namely a pump and a solenoid as well as a pressure sensor. The actuators are used to maintain the pressure. The pressure is set by the supervisory control system. The gas pipeline has three main system modes: automatic, manual, off. In automated mode, there are two schemes – the pump mode, which turns the pump on and off to keep the pressure in the pipe at the set point and the solenoid mode, in which a relief valve controlled by a solenoid is opened and closed to regulate pressure. Both the pump and solenoid modes used a Proportional-Integral-Derivative (PID) control scheme. In manual mode, the operator must manually control the pump and solenoid.

The next component is the communication network in which the protocol used is serial Modbus RTU. Modbus packets include a header and a payload. For Modbus over a Serial Line, a packet includes a device address, function code, payload, and a cyclic 21 redundancy code (CRC) or linear redundancy code (LRC).

A collage of pipes and a pressure gauge

Description automatically generated

Figure 2 Gas pipeline SCADA testbed [17]

The gas pipeline database of cyber attacks was originally created by Tommy Morris and Wei Gao in 2012 [18].  However, study on the original dataset shows some major flaws in the dataset and it was declared unsuitable for machine learning research [17]. Later Ian Turnipseed of Mississippi State University collected more realistic cyber attack gas pipeline dataset known as “new gas pipeline dataset” and made it publicly available for research community.  In our study the new gas pipeline dataset is used.

The “new gas pipeline” dataset is available in two formats. The first form is a comma separated value (CSV) text file, and the second form is an Attribute Relationship File Format (ARFF). The ARFF dataset was created to make it compatible with Waikato Environment for Knowledge Analysis (WEKA), a tool that has a comprehensive list of machine learning algorithms and has been used by many researchers world-wide for testing the performance of specific machine learning algorithms [14]. Although we are not using WEKA, we preprocessed the ARFF dataset to make it ready for training different classification models.  Each record or instance in the dataset represents one packet being delivered to either the MTU or to the RTU. Each instance in the dataset contains SCADA application information along with payload information.   SCADA systems have fixed network topologies and the transactions between the components are repetitive and regular as opposed to IT network whereas the SCADA application data is very dynamic and irregular in nature. The second category of features is the payload information. The payload information provides information about the gas pipeline’s state, settings, and parameters. These values are critical to detect if any anomaly is found in the system which could be due to malfunction of the system or malicious activities by the cyber attackers. There are total of 274,628 instances or rows in the dataset and each row contains twenty columns. The columns are commonly referred to as features. These features are summarized with a brief description in Table 1.

Table 1 Description of features from New Gas Pipeline dataset

|  |  |  |  |
| --- | --- | --- | --- |
| Feature Id | Feature Symbol | Type | Description |
| 1 | address | real | The address of the slave device. Each slave device in the Modbus is assigned a 8-bit address to identify the slave device the master is communicating to and from |
| 2 | function | real | The function codes are primarily used in the gas pipeline to indicate a read (0x03) and write commands (0x16). But there are possibilities of total 256 such commands. A denial of service attack can be launched by setting a function code of 0x08 which corresponds to diagnostic mode where the device would be always in listening mode. |
| 3 | length | real | Length of the Modbus frame. This is fixed for each command and response frame. Frames that are not of specific length can be easily detected as attack |
| 4 | setpoint | real | This value controls the pressure in the gas pipeline |
| 5 | gain | real | Gain parameter of the PID controller |
| 6 | reset rate | real | Reset rate parameter of PID controller |
| 7 | deadband | real | Deadband parameter of PID controller |
| 8 | cycle time | real | Cycle time parameter of PID controller |
| 9 | rate | real | Rate parameter of PID controller |
| 10 | system mode | {0,1,2} | Controls the duty cycle of the system. The following modes are valid  0 – Off  1 – Manual  2 - Automatic |
| 11 | control scheme | {0,1} | The control scheme in the gas pipeline determines whether the system will be controlled by the pump or by the solenoid. There are two schemes:  0 – Pump  1 - Solenoid |
| 12 | pump | {0,1} | This is the state of the pump when system mode is set to manual. There are two possible values:  0 – Off  1 - On |
| 13 | solenoid | {0,1} | This represents the state of the solenoid valve. There are two possible values:  0 – Closed  1 – Open |
| 14 | pressure | real | The current pressure measurement from the gas pipeline |
| 15 | crc rate | real | The Cyclic Redundancy Check (CRC) allows system to check error within a Modbus frame |
| 16 | command response | {0,1} | This value allows the IDS to learn about command and response frame. Two possible values:  0 – Response  1 – Command |
| 17 | time | real | Timestamp of the instance |
| 18 | binary result | {0,1} | Labels to indicate either attack (1) or normal (0) instance |
| 19 | categorized result | Range(0:7) | Labels to indicate the category of attack |
| 20 | specific result | Range(0:35) | Labels to indicate the specific type of attack |

### ****Data Preprocessing and Exploratory Data Analysis (EDA) on Gas Pipeline SCADA System Dataset****

In this subsection, we discuss the preprocessing and feature extraction steps applied to the gas pipeline dataset, as introduced in Section 5. For the simulation study reported in this section, the original gas pipeline dataset was preprocessed with necessary feature extraction into the following categories:

* **Dataset 1:** The entire gas pipeline dataset was preprocessed to remove data instances with a majority of missing feature values, along with features that are not relevant to this study. This resulted in a reduced dataset containing **64,100 rows (data instances) and 12 columns (11 features plus output).**
* **Dataset 2:** The full dataset was further divided into three subsets based on dataset types: **Command (C), Function (F), and Response (R).** After removing instances with missing features, the resulting subsets had sizes of **64,100 × 13 (Command), 68,848 × 4 (Function), and 141,680 × 7 (Response),** respectively.
* **Dataset 3:** Inspired by the approach described in [18,19], new features were introduced in the **Command and Response subsets** of Dataset 2 by computing delta values of the original feature values and centroids of **N past delta values (user-defined).**
* **Dataset 4:** Inspired by the method described in [20], **special imputation techniques** and **one-hot encoding** were used to preprocess the dataset into three subsets based on dataset types: **Command (C), Function (F), and Response (R).**

For this study, we specifically focus on the Response dataset, which contains 68, 849 **rows and 12 features.** This dataset captures key functional attributes of the gas pipeline system and is used to analyze feature importance and optimize feature selection strategies. The experiments conducted in this report aim to formulate an efficient modeling strategy to enhance detection performance while maintaining computational efficiency. The distribution of label categories for the Function dataset is shown below in figure 1:

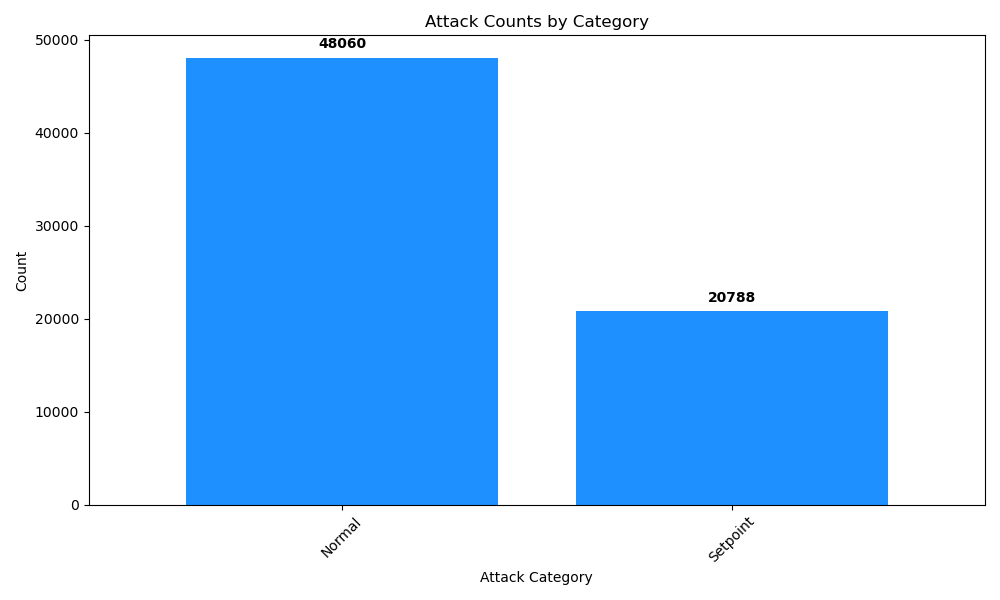


Figure 3 Label Distribution in Response Dataset

The bar chart in Figure 3 illustrates the distribution of different label categories within the full dataset after preprocessing. As shown, the dataset is highly imbalanced, with a significantly larger number of normal instances compared to the attack categories.

To provide a clearer understanding of the attack categories represented in the dataset, Table 2 below outlines the corresponding label numbers assigned to each cyberattack type.

*Table 2 Specific categories of cyber attacks in a SCADA system*

|  |  |  |  |
| --- | --- | --- | --- |
| Attack Number Category Description | | | |
| Setpoint | 1–2 | MPCI | Changes the pressure set point outside and inside of the range of normal operation |
| PID gain | 3–4 | MPCI | Changes the gain outside and inside of the range of normal operation |
| PID reset rate | 5–6 | MPCI | Changes the reset rate outside and inside of the range of normal operation |
| PID rate | 7–8 | MPCI | Changes the rate outside and inside of the range of normal operation |

The dataset's severe class imbalance must be considered when training and evaluating machine learning models. The high prevalence of normal instances compared to attack categories could impact predictive performance, necessitating strategies such as weighted loss functions or data resampling techniques to improve detection capabilities.

# Methodology/Approach

The methodology begins with data ingestion and preprocessing, where the script reads a CSV file containing gas pipeline data (e.g., commands, responses, and functions), initializes paths for storing results, metrics, and models, and optionally normalizes (e.g., Min–Max scaling) and labels datasets if needed. Feature ranking is performed using statistical measures such as standard deviation, absolute difference (mean vs. median), skewness, and kurtosis to assign importance scores to features, with an overall rank determined by aggregating these metrics. Feature selection is conducted using methods like Recursive Feature Elimination (RFE), which iteratively removes less important features, or Sequential Progressive Feature Selection (SPFS), which incrementally builds a feature set by retaining only features that improve model performance. Models, such as Random Forest or XGBoost, are trained during this process, with performance evaluated using metrics like accuracy, precision, recall, F1 score, false positive rate (FPR), and false negative rate (FNR), while training time is also recorded for comparison. Although the current code does not explicitly include cross-validation or prediction, k-fold cross-validation can be applied after feature selection to validate model performance, and the final trained model is saved as a serialized file (e.g., .pkl) for future prediction tasks. Figure 4 shows the proposed methodology:

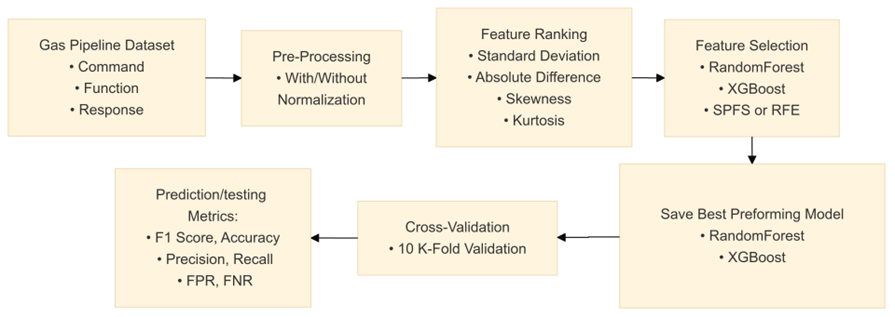


Figure 4 Flowchart of methodology

## Feature Ranking

To preprocess the dataset and identify the most significant features, we applied multiple feature ranking techniques. These methods aim to quantify each feature’s importance in relation to the target variable, ensuring a refined dataset for improved model performance.

Our approach combining statistical analysis and machine learning-based importance to achieve a comprehensive ranking. First, we analyzed the dataset’s statistical properties (SP), evaluating metrics such as standard deviation, skewness, and kurtosis to identify key patterns and variations.

To further refine feature ranking, we employed Weighted Feature Importance (WFI) using two machine learning models: Random Forest and XGBoost. These models assessed feature significance based on their contribution to predictive accuracy, providing a more data-driven selection process. The integration of statistical and model-based ranking ensured that only the most relevant features were retained for subsequent analysis.

### Statistical Properties of Dataset (SP)

Before applying advanced feature selection methods, we first analyzed the statistical properties of the dataset to identify features with high variability, skewed distributions, or extreme outliers that could influence model performance. Understanding these statistical properties allowed us to make informed decisions on which features to prioritize or remove before proceeding with more sophisticated ranking techniques.

To assess the characteristics of each feature, we computed several key statistical metrics. Standard deviation was used to measure variability, as features with higher variance are typically more informative for classification. The absolute difference between the mean and median helped highlight skewed distributions, which often indicate the presence of outliers or unique patterns in the data. Skewness was calculated to quantify data asymmetry, as highly skewed features may introduce biases into the model. Finally, kurtosis was measured to assess the "tailedness" of a distribution, as extreme values could indicate whether a feature had heavy-tailed or light-tailed characteristics, potentially affecting predictive power.

The ranking process was implemented using Python, where each statistical metric was computed for all features. Features were then ranked individually based on their values for each metric. To create a comprehensive ranking system, a total rank was assigned to each feature by summing its rankings across all four metrics, with lower total ranks indicating higher feature importance. This ranking provided a structured approach to selecting the most relevant features for subsequent modeling steps.

The final output of this process was a detailed ranking table containing individual metric values, rankings for each metric, and a consolidated total rank. This table served as a foundation for further feature selection methods, ensuring that the most statistically significant features were retained for model training and evaluation.

One of the outputs from the statistical properties ranking method is shown in Table 2:

Table 2 Example Feature Ranking based on metrics from the Function dataset

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Std Dev | Abs Diff | Skew | Kurtosis | Std Dev Rank | Abs Diff Rank | Skew Rank | Kurtosis Rank | Total Rank |
| address | 0.02 | 0.00 | 21.10 | 695.97 | 1 | 1 | 1 | 16 | 19 |
| pressure | 0.04 | 0.00 | 20.20 | 419.63 | 2 | 2 | 2 | 15 | 21 |
| reset\_rate | 0.08 | 0.00 | 3.01 | 20.31 | 5 | 3 | 5 | 10 | 23 |
| setpoint | 0.14 | 0.00 | 0.96 | 2.90 | 8 | 4 | 9 | 7 | 28 |
| rate | 0.09 | 0.01 | 8.09 | 69.06 | 6 | 6 | 3 | 14 | 29 |
| cycle\_time | 0.08 | 0.03 | 2.49 | 21.71 | 4 | 8 | 6 | 11 | 29 |
| deadband | 0.17 | 0.02 | 0.17 | -0.97 | 9 | 7 | 12 | 5 | 33 |
| function | 0.14 | 0.03 | 5.12 | 27.34 | 7 | 9 | 4 | 13 | 33 |
| system\_mode | 0.40 | 0.25 | 1.16 | -0.42 | 11 | 11 | 8 | 6 | 36 |
| gain | 0.07 | 0.01 | -4.83 | 26.24 | 3 | 5 | 16 | 12 | 36 |
| crc\_rate | 0.42 | 0.28 | 0.09 | -1.97 | 12 | 12 | 13 | 1 | 38 |
| pump | 0.47 | 0.32 | 0.77 | -1.41 | 13 | 13 | 10 | 4 | 40 |
| solenoid | 0.50 | 0.45 | 0.19 | -1.96 | 15 | 15 | 11 | 2 | 43 |
| length | 0.19 | 0.04 | -4.47 | 18.80 | 10 | 10 | 15 | 9 | 44 |
| control\_scheme | 0.47 | 0.34 | -0.67 | -1.54 | 14 | 14 | 14 | 3 | 45 |
| Label | 1.81 | 0.78 | 2.12 | 3.18 | 16 | 16 | 7 | 8 | 47 |

### WFI Using Random Forest

Random Forest assigns importance scores using the Gini impurity criterion, which measures the reduction in node impurity caused by each feature.

**Implementation:**

1. The dataset was split into features (X) and the target variable (y).
2. A RandomForestClassifier was trained on the full dataset.
3. Feature importance scores were extracted from the model’s feature\_importances\_ attribute.
4. Features were ranked based on their importance, with higher values indicating greater predictive power.

**Outputs:**

Tabel 3 shows an example of feature ranking based on Random Forest WFI.

Table 3 Feature importance ranking using Random Forest.

|  |  |
| --- | --- |
| Name | Random Forest Feature Importance |
| function | 0.170343 |
| crc\_rate | 0.14766 |
| length | 0.135827 |
| cycle\_time | 0.094819 |
| reset\_rate | 0.093911 |
| setpoint | 0.088682 |
| gain | 0.074429 |
| deadband | 0.071171 |
| rate | 0.047493 |
| pump | 0.027476 |
| system\_mode | 0.025344 |
| control\_scheme | 0.008638 |
| solenoid | 0.006637 |
| address | 0.006363 |
| pressure\_measurement | 0.001208 |

### WFI Using XGboost

We also applied Weighted Feature Importance (WFI) using XGBoost, a gradient boosting method known for its robustness in feature selection. Unlike Random Forest, XGBoost provides gain-based importance scores, which indicate how much each feature contributes to reducing the model's overall loss function.

**Implementation:**

1. The dataset was split into input features (X) and target variable (y).
2. The class labels were encoded using LabelEncoder to ensure sequential class representation.
3. An XGBoost classifier was trained using the encoded labels.
4. The feature\_importances\_ attribute of the trained model was used to compute importance scores.
5. The features were ranked based on their importance, with higher values indicating greater influence on predictions.

**Outputs:**

The WFI process produced a ranked list of features based on their importance in the XGBoost model. The results were stored in a table, as shown in Table 4:

Table 4 Feature importance ranking using XGBoost.

|  |  |
| --- | --- |
| Name | XGBoost Feature Importance |
| length | 0.349779 |
| function | 0.183473 |
| crc\_rate | 0.175295 |
| rate | 0.154045 |
| pump | 0.032862 |
| gain | 0.021015 |
| cycle\_time | 0.018802 |
| system\_mode | 0.013917 |
| reset\_rate | 0.011271 |
| setpoint | 0.011252 |
| deadband | 0.009006 |
| control\_scheme | 0.008318 |
| solenoid | 0.0062 |
| pressure\_measurement | 0.004765 |
| address | 0 |

These ranked features were used as input for subsequent feature selection methods, ensuring the most relevant attributes were retained.

### Feature Reduction Analysis

After ranking the features using SP and WFI methods, we conducted an in-depth analysis to examine how reducing the number of features affected model performance. The primary objective of this analysis was to determine the smallest subset of features that could maintain high predictive accuracy while minimizing computational complexity.

To achieve this, we first ranked the features using two distinct approaches: the SP (Statistical Properties) method and the WFI method, which utilizes Random Forest and XGBoost for feature importance evaluation. The ranked feature list generated from these methods served as the foundation for selecting different feature subsets for further testing.

Following this, we implemented an iterative feature removal and selection process. We systematically tested various feature set sizes by progressively eliminating the least important features, removing the top one to five features from the ranked list. Additionally, we explored the performance of minimal feature sets by selecting only the top one to five most important features to assess whether a significantly reduced subset could still yield satisfactory predictive results.

For each of these feature subsets, we trained machine learning models using Random Forest and XGBoost. These models were then cross validated using multiple averaging strategies, including micro, and weighted F1-Scores, to ensure a comprehensive assessment of their predictive performance across different evaluation metrics.

The results of this analysis provided valuable insights into the relationship between feature count and classification performance. We identified the optimal number of features necessary to sustain high predictive accuracy and observed how the removal of low-importance features influenced model effectiveness. Furthermore, this process allowed us to quantify the trade-off between feature reduction and computational efficiency, ensuring that the final model maintained strong performance without unnecessary complexity. By systematically refining the feature selection process, we successfully determined the smallest subset of features that could be used without compromising classification accuracy.

## Selective Progressive Feature Selection (SPFS)

The SPFS algorithm was applied as a dynamic feature selection technique, iteratively adding features and evaluating their contributions to model performance, specifically focusing on the F1-Score. The process began with an empty feature set, progressively incorporating features based on their ranking from the previous step. At each iteration, the selected features were used to train either a Random Forest or XGBoost classifier, after which the model was evaluated on a validation set using the F1-Score. If adding a feature improved performance, it was retained; otherwise, it was discarded. This iterative refinement resulted in a final subset of selected features that maximized predictive performance. The SPFS process also generated a comprehensive metrics table detailing F1-Score, precision, recall, false positive rate (FPR), and false negative rate (FNR) at each step.

Below is a flow chart that shows the SPFS process:

A diagram of a performance measurement

AI-generated content may be incorrect.

Figure Flowchart of SPFS process

## Recursive Feature Elimination (RFE)

Recursive Feature Elimination (RFE) is a backward feature elimination algorithm designed to iteratively remove the least significant features until a specified number of features remains. This method is particularly useful for identifying and eliminating features that negatively impact model performance, thereby enhancing predictive accuracy and computational efficiency.

The implementation of RFE began with an initial setup where all features were included in the model. From there, an elimination process was carried out, in which features were removed one at a time, starting with the least important feature as determined by the model’s internal feature importance metric. After each removal, the model was retrained, and its F1-Score was recalculated to evaluate its performance. This iterative process continued until the optimal subset of features was identified based on the highest recorded F1-Score.

The RFE process produced several key outputs. First, it generated a ranking of features based on their contribution to the model’s overall performance. Additionally, it identified a subset of features that yielded the best results, similar in function to the SPFS method.

Figure 3 presents the flow chart of RFE process:

A diagram of a process flow

AI-generated content may be incorrect.

Figure Flowchart of RFE

## Cross-Validation

To ensure the reliability and robustness of the model, we employed k-fold cross-validation using the selected features identified during the feature selection process. This method divides the training dataset into k equally sized folds (with k=10 in this implementation), where the model is trained on k-1 folds and validated on the remaining fold. The process is repeated k times, with each fold used as the validation set exactly once, ensuring that every data point in the training set is used for both training and validation.

The training dataset was generated through an 80/20 split of the original data, where 80% was allocated for training and 20% reserved as a separate test set for final evaluation. During cross-validation, only the training data is used, and the results of all folds are aggregated to evaluate the model's overall performance.

To determine the best-performing model, we selected the model that achieved the highest F1 score across the 10-folds. This model was then tested on the remaining 20% of the data to assess its final performance. After evaluation, the selected model was saved as a .pkl file for future prediction tasks.

## Metrics Evaluation

After cross-validation, the best model's performance was evaluated using a variety of metrics to assess its predictive accuracy and ability to generalize to unseen data. The following metrics were calculated for each fold during the cross-validation process:

* **Accuracy**: The proportion of correctly predicted instances out of the total instances. It provides a general overview of the model's performance.
* **Precision**: The ratio of true positive predictions to all positive predictions. It evaluates the model's ability to avoid false positives.
* **Recall**: The ratio of true positive predictions to all actual positive instances. It measures the model's ability to identify all relevant instances.
* **F1 Score**: The harmonic mean of precision and recall, balancing the trade-off between the two metrics.
* **False Positive Rate (FPR)**: The proportion of negative instances incorrectly classified as positive.
* **False Negative Rate (FNR)**: The proportion of positive instances incorrectly classified as negative.

These metrics were saved to a CSV file for documentation and analysis.

## Averaging Methods for Metrics Evaluation

In multi-class classification tasks, evaluation metrics such as Precision, Recall, and F1-Score are commonly averaged using different strategies to provide a holistic assessment of the model's performance across all classes. The three primary averaging methods used in this study are Micro Averaging, Macro Averaging, and Weighted Averaging. Each method offers a unique perspective on performance, making them essential for analyzing model effectiveness in different contexts.

### Micro Averaging

Micro averaging computes the metrics by aggregating the contributions of all classes before performing the final calculation. It treats every individual instance equally, regardless of class distribution. This is especially useful when class imbalance is present.

The micro-average calculations are given by:

Precision = Recall = F1 =

where:

* TP (True Positives) is the number of correctly classified instances for each class.
* FP (False Positives) is the number of misclassified instances.
* FN (False Negatives) is the number of missed instances.

Micro averaging gives an overall measure of the model's performance and is particularly useful in highly imbalanced datasets, where larger classes would otherwise dominate the results.

### Weighted Averaging

Weighted averaging is similar to macro averaging, but it weighs each class's contribution to the final score based on the number of true instances in that class.

Precision = Recall = F1 =

where:

is the proportion of total samples belonging to class , computed as

Weighted averaging provides a balanced measure that accounts for both class-wise performance and class distribution. It is particularly useful when comparing models on datasets with highly imbalanced class distributions.

### Evaluation of Averaging Methods

For each iteration of model evaluation, we tested both micro and weighted averaging to determine which provided the best performance. Initially, we also considered macro averaging, but its scores were significantly lower than the other two methods, making it unsuitable for this analysis.

During the experiments, micro averaging generally performed well in cases where class imbalance was significant, as it emphasized overall correctness rather than per-class performance. Weighted averaging provided a balanced perspective by incorporating class distributions, making it particularly useful for understanding real-world classification performance

The results of these comparisons are further analyzed in Section 8, where we discuss the impact of each averaging method on final model performance.

# Design/Framework/Testbed

To systematically evaluate different feature selection and ranking techniques, we implemented a structured framework that iterated through various combinations of ranking methods, feature selection strategies, model types, and averaging methods. The objective of this framework was to identify the most effective approach for feature selection and model training, ultimately improving classification performance while maintaining computational efficiency.

The process began by looping through each ranking method, including SP (Statistical Properties) and WFI (Weight-based Feature Importance), to establish an initial ordering of features based on their statistical significance or model-derived importance scores. Following this, we applied different feature selection techniques, such as Recursive Feature Elimination (RFE), Statistical Property-based Feature Selection (SPFS), and manual removal or retention of top-ranked features, to determine the optimal feature subset for training.

For each selected subset of features, two machine learning models—Random Forest and XGBoost—were trained and evaluated using multiple averaging strategies (micro, macro, and weighted F1-Scores) to ensure a robust comparison across different evaluation metrics. Each experiment involved training a model on the selected features and measuring its performance across accuracy, precision, recall, F1-Score, false positive rate (FPR), and false negative rate (FNR).

In addition to testing all possible feature selection techniques, we performed an iterative feature reduction analysis, systematically removing the least important features (from 1 to X) to analyze their impact on model performance. The goal was to determine the minimum number of features required to maintain high classification accuracy while improving computational efficiency. By comparing the performance of models trained on progressively smaller feature subsets, we identified potential trade-offs between feature count and predictive power.

This comprehensive approach allowed us to evaluate the effects of different feature selection and ranking methods, ultimately leading to the identification of the most effective strategy for feature reduction in gas pipeline anomaly detection. The results from these experiments provided insights into which feature selection and ranking combinations yielded the highest F1-Scores while maintaining low false positive and false negative rates.

The diagram below illustrates the automated pipeline used for feature selection and model evaluation in our study. The process begins with loading the dataset, followed by ranking the features using Statistical Properties (SP) or Weighted Feature Importance (WFI) methods, which include Random Forest and XGBoost. After ranking, different feature selection techniques such as Recursive Feature Elimination (RFE) and Statistical Properties Feature Selection (SPFS) are applied to refine the feature subset. Additionally, an alternative approach is employed where the least important *X* features are systematically removed. The selected feature subset is then used to train a machine learning model, after which the model's performance is evaluated using various metrics, including F1-Score, Accuracy, Precision, Recall, False Positive Rate (FPR), False Negative Rate (FNR), and computation time.

To comprehensively analyze the impact of feature selection on model performance, a script was developed to systematically loop through all possible combinations of ranking methods, feature selection techniques, model types (Random Forest and XGBoost), and averaging strategies (micro, macro, weighted). This ensured a thorough evaluation of different configurations, allowing for the identification of the optimal approach that maximizes predictive accuracy while minimizing feature redundancy.

A diagram of a structure

AI-generated content may be incorrect.

Figure 7 Flowchart of model evaluation process

# Results and Discussion

## Overview of Result Metrics

The performance of the models was evaluated using standard classification metrics, including Accuracy, Precision, Recall, F1 Score, False Positive Rate (FPR), and False Negative Rate (FNR). The results were analyzed across different feature selection methods and varying numbers of selected features to assess the trade-off between model complexity and predictive performance.

One of the key findings from this analysis is the impact of feature reduction on model performance. The F1 Score, a balanced measure of precision and recall, was used as the primary performance metric for evaluation. The figure below illustrates how the F1 Score changes as the number of selected features decreases.

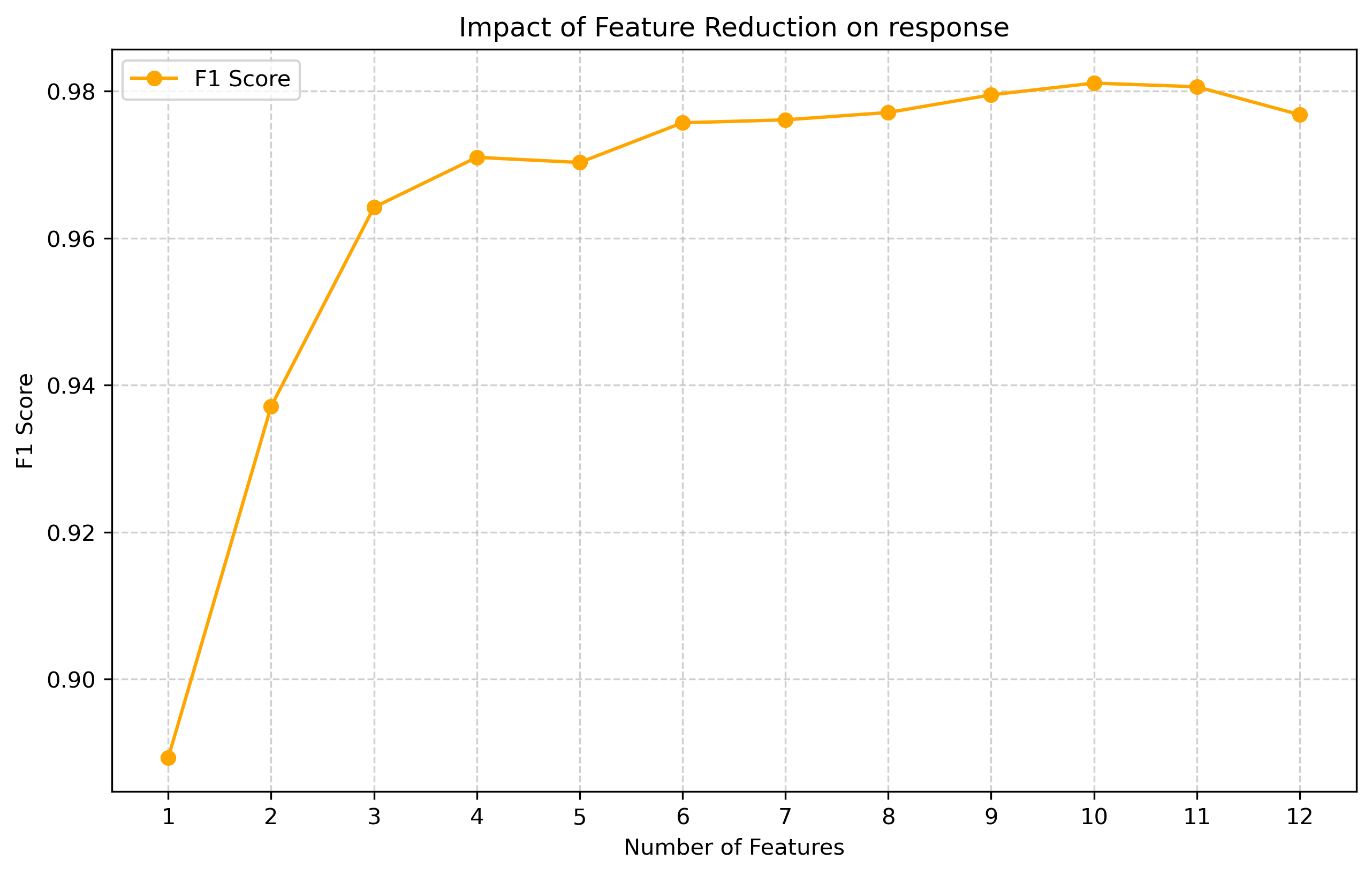


Figure 8 Impact of Feature Reduction on Model Performance (using best of SP and Micro Averaging)

From the graph, we can observe that the F1 Score remains consistently high when using 9 or more features but slightly decreases at 12 features. As the number of selected features decreases from 8 to 4, there is a gradual decline in performance, which suggests that some features, while not critical, still contribute to overall predictive accuracy. However, when fewer than 3 features are used, there is a significant drop in F1 Score, indicating that certain features are essential for maintaining classification performance. This trend highlights the importance of selecting an optimal number of features to balance predictive accuracy with computational efficiency. By carefully applying feature selection techniques, we can ensure minimal loss in classification performance while improving efficiency in model training and inference.

### Key Performance Metrics

The following metrics were used to evaluate model performance:

* **Accuracy**: The percentage of correctly classified instances.
* **Precision**: The ratio of correctly predicted positive observations to total predicted positives.
* **Recall**: The ratio of correctly predicted positive observations to actual positives.
* **F1 Score**: The harmonic mean of precision and recall, providing a balanced assessment of model performance.
* **False Positive Rate (FPR)**: The proportion of false positives among all negative cases.
* **False Negative Rate (FNR)**: The proportion of false negatives among all positive cases.
* **Time Taken**: The computational time required to train and evaluate the model.

### Performance Trends

A deeper analysis of performance trends revealed several key insights. The most notable decline in model performance occurred when the number of selected features fell below seven, emphasizing that while removing redundant features improved efficiency, eliminating too many negatively impacted classification accuracy. Models trained with 8 to 9 features consistently achieved the highest F1 scores, reinforcing the importance of careful and balanced feature selection. Although moderate reduction proved beneficial, removing more than six or seven features resulted in noticeable performance degradation, confirming the necessity of certain key features to maintain predictive success.

A comparison between Random Forest (RF) and XGBoost (XGB) models demonstrated distinct strengths and weaknesses. Random Forest consistently outperformed XGBoost across all feature selection strategies, achieving the highest F1 scores in nearly every scenario and maintaining strong predictive performance even with substantial feature reduction. Notably, an RF model trained with only three features reached an impressive F1 score of 0.9642, surpassing multiple XGBoost models that utilized all nine features. These findings highlight RF's resilience and superior predictive accuracy, even under extreme feature constraints. Conversely, XGBoost, although significantly more computationally efficient, required more features to approach RF's performance and experienced rapid accuracy deterioration as feature counts dropped.

Feature ranking methods significantly impacted model effectiveness by identifying valuable subsets of predictive variables. Weighted Feature Importance (WFI) and Statistical Properties (SP) consistently emerged as the most effective ranking strategies, especially when combined with Recursive Feature Elimination (RFE), which consistently yielded slightly higher F1 scores than Sequential Progressive Feature Selection (SPFS). Removal-based methods (rem\_1 to rem\_4) demonstrated that certain lower-importance features could safely be discarded with minimal accuracy loss; however, removing more than six or seven features led to gradual but clear declines in performance. The highest recorded F1 score (0.9811) was achieved using a WFI-based ranking strategy combined with an RF model, reinforcing the effectiveness of pairing robust ranking approaches with strategic feature selection.

Analyzing feature removal patterns indicated that the features "Control Scheme," "Solenoid," and "Pump" were consistently among the least impactful and frequently removed, suggesting limited contributions to predictive accuracy. Nonetheless, simply retaining only top-ranked features without thoughtful selection caused noticeable degradation, emphasizing the importance of maintaining a balanced feature set that includes strategically selected lower-ranked features.

Overall, these findings clearly establish Random Forest as the optimal classifier for achieving high predictive accuracy in feature-limited scenarios, consistently outperforming XGBoost despite XGB's computational advantages. Maintaining a carefully balanced set of around 8 to 9 key features and utilizing ranking strategies like WFI and SP, combined with effective selection methods such as RFE, provided the strongest overall results, optimizing both performance and computational efficiency.

## Ranking Results

### Statistical Properties Ranking Results

Below are the results of the ranking using the SP method, which evaluates features based on statistical properties such as Standard Deviation, Absolute Difference, Skewness, and Kurtosis. The Total Rank is derived by summing the individual rankings, with a lower total rank indicating higher feature importance.

Table 5 Ranking features in original dataset based on statistical properties

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Standard Deviation | Absolute Difference | Skewness | Kurtosis | Standard Deviation Rank | Absolute Difference Rank | Skewness Rank | Kurtosis Rank | Total Rank |
| reset\_rate | 0.080091 | 0.00287 | 3.027019 | 20.45427 | 1 | 1 | 3 | 8 | 13 |
| rate | 0.086332 | 0.01207 | 8.00789 | 67.64717 | 2 | 3 | 2 | 11 | 18 |
| deadband | 0.172602 | 0.00885 | 0.168879 | -0.96307 | 3 | 2 | 10 | 5 | 20 |
| cycle\_time | 0.347518 | 0.11861 | 2.502339 | 21.90586 | 4 | 4 | 4 | 9 | 21 |
| pump | 0.466281 | 0.319486 | 0.774275 | -1.4005 | 5 | 5 | 7 | 4 | 21 |
| control\_scheme | 0.473897 | 0.340547 | -0.67295 | -1.54714 | 6 | 6 | 11 | 2 | 25 |
| solenoid | 0.497947 | 0.454697 | 0.181959 | -1.96689 | 7 | 8 | 9 | 1 | 25 |
| system\_mode | 0.795119 | 0.492476 | 1.16533 | -0.40903 | 8 | 9 | 5 | 6 | 28 |
| setpoint | 6.720592 | 0.398772 | 0.956644 | 2.879614 | 9 | 7 | 6 | 7 | 29 |
| crc\_rate | 1951.891 | 688.4975 | 0.374177 | -1.44625 | 11 | 11 | 8 | 3 | 33 |
| pressure\_measurement | 1.36E+37 | 7.28E+35 | 20.14181 | 416.988 | 12 | 12 | 1 | 12 | 37 |
| gain | 8.630923 | 1.001917 | -4.86458 | 26.78825 | 10 | 10 | 12 | 10 | 42 |

The results indicate that Reset Rate, Rate, and Deadband are the most influential features in the dataset. Reset Rate ranked first in both Standard Deviation and Absolute Difference, with an overall lowest total rank of 13, making it the most critical feature. Rate followed closely with a total rank of 18, securing top positions in Skewness and a high ranking in Absolute Difference. Deadband, with a total rank of 20, emerged as another essential feature, performing well across multiple statistical criteria. These findings suggest that these three features should be prioritized in feature selection, as they play a significant role in predictive accuracy.

Moderately important features included Cycle Time, Pump, and Setpoint, each achieving total ranks between 21 and 29. While these features contribute to the dataset’s structure, their relative importance is slightly lower than the top-ranked ones. Cycle Time (rank 21) and Pump (rank 21) demonstrated balanced performance across all statistical metrics, whereas Setpoint (rank 29) had strong Standard Deviation and Skewness rankings but performed moderately in other areas. These features remain relevant for modeling but may be considered for removal if computational efficiency is a priority.

The least important features in this ranking were Control Scheme, Solenoid, and Pressure Measurement, which received the highest total ranks (25, 25, and 37, respectively). Control Scheme and Solenoid performed poorly across most statistical properties, particularly in Skewness and Standard Deviation rankings. Pressure Measurement, while ranking first in Skewness, performed the worst across all other ranking criteria, leading to the highest total rank (37). These features contributed the least to overall predictive power, indicating that they may be redundant and potentially removable during feature selection.

These findings offer valuable insights for feature selection and dimensionality reduction. Reset Rate, Rate, and Deadband should be retained across models due to their strong contribution to predictive performance. In contrast, features such as Control Scheme, Solenoid, and Pressure Measurement can be considered for removal when optimizing model efficiency. This ranking highlights the impact of different features on the dataset and provides a data-driven strategy for selecting the most informative features while discarding those with minimal relevance.

### Weighted feature selection ranking results using RandomForest Classifier

This section presents the feature importance results obtained using a RandomForest classifier. Unlike the SP method—which relies on statistical properties—the RandomForest ranking quantifies the importance of each feature based on how much it reduces uncertainty (impurity) in the decision trees. The importance scores are derived from the contribution of each feature to the model's overall predictive performance. A higher feature importance score indicates that the feature plays a more critical role in making accurate predictions. The following table (Table 6) shows the ranked features for the function dataset using the RandomForest classifier.

Table 6 Feature importance rankings for the function dataset using the RandomForest classifier

|  |  |
| --- | --- |
| Name | Random Forest Feature Importance |
| pressure\_measurement | 0.527674 |
| setpoint | 0.08867 |
| reset\_rate | 0.084729 |
| deadband | 0.074899 |
| crc\_rate | 0.057273 |
| cycle\_time | 0.056421 |
| gain | 0.050223 |
| system\_mode | 0.014984 |
| solenoid | 0.013538 |
| pump | 0.012236 |
| control\_scheme | 0.011589 |
| rate | 0.007764 |

The results from the RandomForest classifier reveal that Pressure Measurement (0.5277) is by far the most influential feature in the dataset, with a significantly higher importance score than all other features. Setpoint (0.0887) and Reset Rate (0.0847) follow as the next most important features, suggesting that these three variables contribute the most to predictive accuracy. The strong importance of Setpoint and Reset Rate implies that these features play a crucial role in model decision-making and should be prioritized in feature selection.

A set of moderately important features includes Deadband (0.0749), CRC Rate (0.0573), Cycle Time (0.0564), and Gain (0.0502). These features contribute meaningfully to model performance but to a lesser extent than the top-ranked features. Their importance may vary depending on the specific dataset characteristics and modeling objectives.

The least important features in the RandomForest ranking include System Mode (0.0150), Solenoid (0.0135), Pump (0.0122), Control Scheme (0.0116), and Rate (0.0078). These variables have significantly lower importance scores, indicating that they have minimal impact on model predictions. Since they contribute little to classification accuracy, they could be considered for removal in feature selection processes when optimizing model efficiency.

The RandomForest ranking method provides a different perspective on feature importance compared to the SP method. While SP identified Reset Rate, Rate, and Deadband as highly significant, RandomForest assigned the highest importance to Pressure Measurement, with Reset Rate and Setpoint ranking highly as well. Additionally, SP ranked Control Scheme and Solenoid moderately, whereas RandomForest placed them among the least important features.

### Weighted feature selection ranking results using XGBoost Classifier

This section details the feature importance rankings as determined by an XGBoost classifier. XGBoost is a gradient boosting algorithm that evaluates the contribution of each feature based on the improvement it brings to the model’s loss function. The resulting importance scores reflect the degree to which each feature influences predictive accuracy. In this context, higher scores indicate a more substantial impact on model performance. Table 7 summarizes the feature importance rankings for the function dataset using the XGBoost classifier.

Table 7 Feature importance rankings for the Response dataset using the XGBoost classifier

|  |  |
| --- | --- |
| Name | XGBoost Feature Importance |
| pressure\_measurement | 0.350216 |
| system\_mode | 0.079991 |
| setpoint | 0.069707 |
| cycle\_time | 0.061433 |
| reset\_rate | 0.06046 |
| deadband | 0.059807 |
| control\_scheme | 0.05911 |
| solenoid | 0.057731 |
| pump | 0.056702 |
| gain | 0.055546 |
| rate | 0.050887 |
| crc\_rate | 0.038411 |

The feature importance rankings obtained from the XGBoost model provide a unique perspective on the significance of different features compared to SP ranking and Random Forest ranking. According to XGBoost, the most influential feature in the dataset was Pressure Measurement (Feature Importance = 0.3502). This significant gap in importance compared to other features highlights its dominant role in predictive performance and reinforces its necessity in feature selection.

A group of moderately important features follows, including System Mode (0.0799), Setpoint (0.0697), Cycle Time (0.0614), Reset Rate (0.0605), Deadband (0.0598), and Control Scheme (0.0591). These features still played a substantial role in model predictions and should be prioritized when applying feature selection techniques to maintain classification performance.

Further down, Solenoid (0.0577), Pump (0.0567), Gain (0.0555), and Rate (0.0509) were found to have less impact compared to the top features, but their contributions were still notable. Removing these features may reduce model complexity, but their impact on performance would need to be carefully evaluated to avoid unnecessary drops in accuracy.

The least important feature identified by the XGBoost classifier was CRC Rate (0.0384), which had the lowest feature importance score. This suggests that CRC Rate has minimal influence on predictive accuracy and could be considered for removal during feature selection, particularly in scenarios requiring computational efficiency.

The XGBoost feature importance rankings show differences when compared to Random Forest and SP ranking methods. Unlike Random Forest, which ranked CRC Rate, Reset Rate, and Setpoint as the most significant features, XGBoost assigned the highest importance to Pressure Measurement. Additionally, System Mode, which was assigned a lower importance score in Random Forest, was ranked as the second most important feature in XGBoost.

## ****Summary of Best-Performing Models****

### Best F1 Scores

To identify the most effective model configurations, we analyzed various feature selection methods, classifiers, and ranking approaches. Table 8 presents the **top 30 models ranked by F1 score**, highlighting optimal configurations for intrusion detection.

Table 8 Top 30 best preforming combinations based on greatest F1 score

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Rank Method | Feature Selection | Model Type | Average Type | Number of Features | Removed Features | F1 Score | Ranking Time | Feature Selection Time | Evaluation Time | Total Time |
| WFI-XGB | rem\_2 | RF | micro | 10 | rate, crc\_rate | 0.9811 | 0.95 | 7.08 | 57.85 | 65.87 |
| WFI-XGB | rem\_2 | RF | weighted | 10 | rate, crc\_rate | 0.981 | 0.95 | 7.08 | 56.51 | 64.53 |
| WFI-RF | RFE | RF | micro | 11 | crc\_rate | 0.9806 | 7.92 | 85.58 | 51.71 | 145.21 |
| WFI-RF | RFE | RF | weighted | 11 | crc\_rate | 0.9805 | 7.92 | 85.58 | 50.67 | 144.18 |
| WFI-XGB | RFE | RF | micro | 10 | solenoid, crc\_rate | 0.9802 | 0.95 | 76.17 | 53.17 | 130.29 |
| WFI-XGB | RFE | RF | weighted | 10 | solenoid, crc\_rate | 0.9801 | 0.95 | 76.17 | 55.19 | 132.30 |
| SP | RFE | RF | micro | 11 | crc\_rate | 0.9798 | 0.09 | 78.20 | 51.87 | 130.16 |
| WFI-XGB | rem\_1 | RF | micro | 11 | crc\_rate | 0.9798 | 0.95 | 7.08 | 61.04 | 69.06 |
| SP | RFE | RF | weighted | 11 | crc\_rate | 0.9797 | 0.09 | 78.20 | 51.95 | 130.24 |
| WFI-XGB | rem\_1 | RF | weighted | 11 | crc\_rate | 0.9797 | 0.95 | 7.08 | 54.72 | 62.74 |
| WFI-XGB | rem\_3 | RF | micro | 9 | rate, gain, crc\_rate | 0.9795 | 0.95 | 7.08 | 60.48 | 68.51 |
| WFI-XGB | rem\_3 | RF | weighted | 9 | rate, gain, crc\_rate | 0.9794 | 0.95 | 7.08 | 45.36 | 53.38 |
| WFI-XGB | rem\_4 | RF | micro | 8 | rate, pump, gain, crc\_rate | 0.9771 | 0.95 | 7.08 | 34.67 | 42.69 |
| WFI-XGB | rem\_4 | RF | weighted | 8 | rate, pump, gain, crc\_rate | 0.9769 | 0.95 | 7.08 | 48.98 | 57.01 |
| WFI-XGB | rem\_5 | RF | micro | 7 | pump, solenoid, crc\_rate, rate, gain | 0.9761 | 0.95 | 7.08 | 39.13 | 47.15 |
| WFI-XGB | rem\_5 | RF | weighted | 7 | pump, solenoid, crc\_rate, rate, gain | 0.9759 | 0.95 | 7.08 | 40.03 | 48.06 |
| WFI-XGB | rem\_6 | RF | micro | 6 | pump, control\_scheme, solenoid, crc\_rate, rate, gain | 0.9757 | 0.95 | 7.08 | 47.98 | 56.01 |
| WFI-XGB | rem\_6 | RF | weighted | 6 | pump, control\_scheme, solenoid, crc\_rate, rate, gain | 0.9755 | 0.95 | 7.08 | 45.35 | 53.37 |
| WFI-RF | rem\_2 | RF | micro | 10 | rate, control\_scheme | 0.9749 | 7.92 | 7.00 | 66.31 | 81.24 |
| WFI-RF | rem\_1 | RF | micro | 11 | rate | 0.9748 | 7.92 | 7.00 | 65.12 | 80.05 |
| SP | rem\_1 | RF | micro | 11 | gain | 0.9747 | 0.09 | 6.71 | 61.85 | 68.65 |
| WFI-RF | rem\_2 | RF | weighted | 10 | rate, control\_scheme | 0.9746 | 7.92 | 7.00 | 67.45 | 82.38 |
| WFI-RF | rem\_1 | RF | weighted | 11 | rate | 0.9745 | 7.92 | 7.00 | 64.04 | 78.97 |
| SP | rem\_1 | RF | weighted | 11 | gain | 0.9743 | 0.09 | 6.71 | 64.26 | 71.06 |
| WFI-RF | rem\_3 | RF | micro | 9 | rate, pump, control\_scheme | 0.9741 | 7.92 | 7.00 | 71.97 | 86.90 |
| WFI-RF | rem\_3 | RF | weighted | 9 | rate, pump, control\_scheme | 0.9738 | 7.92 | 7.00 | 71.98 | 86.91 |
| WFI-RF | rem\_4 | RF | micro | 8 | rate, pump, solenoid, control\_scheme | 0.9718 | 7.92 | 7.00 | 57.28 | 72.21 |
| WFI-RF | rem\_4 | RF | weighted | 8 | rate, pump, solenoid, control\_scheme | 0.9714 | 7.92 | 7.00 | 57.87 | 72.79 |
| WFI-RF | rem\_8 | RF | micro | 4 | system\_mode, pump, control\_scheme, solenoid, crc\_rate, cycle\_time, rate, gain | 0.971 | 7.92 | 7.00 | 63.74 | 78.67 |
| WFI-RF | rem\_8 | RF | weighted | 4 | system\_mode, pump, control\_scheme, solenoid, crc\_rate, cycle\_time, rate, gain | 0.9707 | 7.92 | 7.00 | 63.06 | 77.98 |

The analysis of the best-performing models revealed that Random Forest (RF) consistently achieved the highest F1 scores, surpassing XGBoost (XGB) across various feature selection methods. The top-ranked model, utilizing RF with the WFI-XGB ranking method and a feature set reduced to ten features, achieved an F1 score of 0.9811, outperforming all other configurations. This model used a removal-based feature selection method, rem\_2, which eliminated rate and crc\_rate, showing that certain features could be discarded without compromising performance.

Recursive Feature Elimination (RFE) also yielded strong results, particularly when used with RF. The highest F1 score recorded under RFE was 0.9806, demonstrating that RF benefits from progressive feature selection techniques. Models using XGB consistently lagged behind in terms of F1 score, with the best XGB model achieving a maximum F1 score of 0.9673, lower than multiple RF models with fewer features.

Feature selection played a crucial role in optimizing model performance. Removing a small number of redundant features, such as solenoid and crc\_rate, led to significant improvements in both efficiency and accuracy. While models with ten or more features performed the best, configurations with as few as six features still maintained competitive F1 scores above 0.975.

Interestingly, RF models demonstrated resilience to feature reduction, maintaining high classification accuracy even when only six or seven features were retained. In contrast, XGB models experienced a sharper decline in performance as the feature count decreased. An RF model with only three features still managed to outperform multiple XGB models retaining up to nine features, highlighting RF's superior adaptability when working with a limited number of predictive attributes.

Overall, the findings indicate that RF is the dominant classifier in this dataset, achieving both the highest F1 scores and superior robustness to feature reduction. While XGB remained competitive, it consistently trailed behind RF in terms of F1 score, even in configurations with optimized feature selection. These results underscore the importance of feature selection strategies and confirm that RF is the preferred classifier for maximizing predictive accuracy in this dataset.

### Best accuracy

Tabel 9 ranks the top 30 model configurations based on accuracy, highlighting the trade-offs between feature selection methods, classifier choice, and computational efficiency.

Table 9 Best 30 accuracy scores (Micro Averaging)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Rank Method | Feature Selection | Model Type | Number of Features | Removed Features | Accuracy | F1 Score | Ranking Time | Feature Selection Time | Evaluation Time | Total Time |
| WFI-XGB | rem\_2 | RF | 10 | rate, crc\_rate | 0.9811 | 0.9811 | 0.95 | 7.08 | 57.85 | 65.87 |
| WFI-RF | RFE | RF | 11 | crc\_rate | 0.9806 | 0.9806 | 7.92 | 85.58 | 51.71 | 145.21 |
| WFI-XGB | RFE | RF | 10 | solenoid, crc\_rate | 0.9802 | 0.9802 | 0.95 | 76.17 | 53.17 | 130.29 |
| SP | RFE | RF | 11 | crc\_rate | 0.9798 | 0.9798 | 0.09 | 78.20 | 51.87 | 130.16 |
| WFI-XGB | rem\_1 | RF | 11 | crc\_rate | 0.9798 | 0.9798 | 0.95 | 7.08 | 61.04 | 69.06 |
| WFI-XGB | rem\_3 | RF | 9 | rate, gain, crc\_rate | 0.9795 | 0.9795 | 0.95 | 7.08 | 60.48 | 68.51 |
| WFI-XGB | rem\_4 | RF | 8 | rate, pump, gain, crc\_rate | 0.9771 | 0.9771 | 0.95 | 7.08 | 34.67 | 42.69 |
| WFI-RF | SPFS | RF | 12 |  | 0.9768 | 0.9768 | 7.92 | 79.50 | 63.25 | 150.67 |
| WFI-RF | None | RF | 12 |  | 0.9766 | 0.9766 | 7.92 | 7.00 | 61.48 | 76.41 |
| SP | None | RF | 12 |  | 0.9766 | 0.9766 | 0.09 | 6.71 | 62.53 | 69.32 |
| WFI-XGB | None | RF | 12 |  | 0.9766 | 0.9766 | 0.95 | 7.08 | 62.88 | 70.91 |
| WFI-XGB | rem\_5 | RF | 7 | pump, solenoid, crc\_rate, rate, gain | 0.9761 | 0.9761 | 0.95 | 7.08 | 39.13 | 47.15 |
| WFI-XGB | rem\_6 | RF | 6 | pump, control\_scheme, solenoid, crc\_rate, rate, gain | 0.9757 | 0.9757 | 0.95 | 7.08 | 47.98 | 56.01 |
| WFI-RF | rem\_2 | RF | 10 | rate, control\_scheme | 0.9749 | 0.9749 | 7.92 | 7.00 | 66.31 | 81.24 |
| WFI-RF | rem\_1 | RF | 11 | rate | 0.9748 | 0.9748 | 7.92 | 7.00 | 65.12 | 80.05 |
| SP | rem\_1 | RF | 11 | gain | 0.9747 | 0.9747 | 0.09 | 6.71 | 61.85 | 68.65 |
| WFI-XGB | SPFS | RF | 12 |  | 0.9746 | 0.9746 | 0.95 | 66.61 | 60.26 | 127.82 |
| WFI-RF | rem\_3 | RF | 9 | rate, pump, control\_scheme | 0.9741 | 0.9741 | 7.92 | 7.00 | 71.97 | 86.90 |
| SP | SPFS | RF | 12 |  | 0.9741 | 0.9741 | 0.09 | 55.56 | 61.00 | 116.65 |
| WFI-RF | rem\_4 | RF | 8 | rate, pump, solenoid, control\_scheme | 0.9718 | 0.9718 | 7.92 | 7.00 | 57.28 | 72.21 |
| WFI-RF | rem\_8 | RF | 4 | system\_mode, pump, control\_scheme, solenoid, crc\_rate, cycle\_time, rate, gain | 0.971 | 0.971 | 7.92 | 7.00 | 63.74 | 78.67 |
| WFI-XGB | rem\_7 | RF | 5 | pump, deadband, control\_scheme, solenoid, crc\_rate, rate, gain | 0.9703 | 0.9703 | 0.95 | 7.08 | 39.51 | 47.54 |
| WFI-RF | rem\_5 | RF | 7 | system\_mode, pump, control\_scheme, solenoid, rate | 0.9696 | 0.9696 | 7.92 | 7.00 | 64.97 | 79.90 |
| WFI-XGB | rem\_2 | XGB | 10 | rate, crc\_rate | 0.9695 | 0.9695 | 0.95 | 1.16 | 8.79 | 10.89 |
| WFI-XGB | RFE | XGB | 9 | rate, solenoid, crc\_rate | 0.9673 | 0.9673 | 0.95 | 11.22 | 7.85 | 20.03 |
| WFI-RF | rem\_6 | RF | 6 | system\_mode, pump, control\_scheme, solenoid, rate, gain | 0.9672 | 0.9672 | 7.92 | 7.00 | 68.45 | 83.38 |
| WFI-RF | RFE | XGB | 10 | rate, crc\_rate | 0.967 | 0.967 | 7.92 | 13.12 | 13.43 | 34.47 |
| WFI-RF | SPFS | XGB | 12 |  | 0.9669 | 0.9669 | 7.92 | 13.64 | 12.40 | 33.97 |
| WFI-RF | None | XGB | 12 |  | 0.9662 | 0.9662 | 7.92 | 1.48 | 10.63 | 20.04 |
| SP | None | XGB | 12 |  | 0.9662 | 0.9662 | 0.09 | 1.25 | 9.95 | 11.29 |

The analysis of model accuracy rankings reveals that Random Forest (RF) consistently achieved the highest accuracy scores, outperforming XGBoost (XGB) in most configurations. The top-ranked model, which utilized RF with the WFI-XGB ranking method and the rem\_2 feature selection strategy, achieved an accuracy of 0.9811. This model retained ten features, removing rate and crc\_rate, demonstrating that certain features could be eliminated without negatively impacting classification accuracy.

Recursive Feature Elimination (RFE) also proved to be an effective selection method, with an RF-based RFE model achieving an accuracy of 0.9806 while retaining eleven features. Other high-ranking models followed a similar trend, with the rem\_3 and rem\_4 feature selection methods producing RF models with eight or nine features while maintaining high accuracy scores above 0.977.

RF demonstrated superior robustness to feature reduction, maintaining high accuracy even when only six or seven features were retained. Notably, an RF model with just four features achieved an accuracy of 0.971, outperforming multiple XGB models with significantly more features. In contrast, XGB models experienced a more noticeable decline in accuracy as the number of selected features decreased. The best XGB model recorded an accuracy of 0.9695, ranking lower than several RF models with fewer features.

These findings indicate that RF is the superior classifier in terms of accuracy, consistently delivering the highest scores across a variety of feature selection strategies. While XGB remained competitive, particularly in terms of computational efficiency, it did not achieve the top accuracy scores. This reinforces the conclusion that RF is the preferred model when maximizing predictive accuracy is the primary objective.

### Performance of Models with Limited Features

The analysis of model performance with a limited number of features focused on identifying configurations that maintain high classification accuracy and efficiency despite a reduced feature set. The results confirm that Random Forest (RF) consistently outperformed XGBoost (XGB), even when working with fewer than nine features. The highest-ranked model in this category used RF with WFI-XGB ranking, which retained nine features and achieved an F1 score of 0.9795, outperforming all other models. In contrast, the best XGB model, which also used nine features, achieved an F1 score of 0.9673, further emphasizing RF’s advantage in this setting.

Table 10 presents the best-performing models under feature-limited conditions:

Table Best preforming combinations when reducing the number of features to 9 or less (Micro Average)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Rank Method | Model Type | Number of Features | Removed Features | Accuracy | F1 Score | Ranking Time | Feature Selection Time | Evaluation Time | Total Time |
| WFI-XGB | RF | 9 | rate, gain, crc\_rate | 0.9795 | 0.9795 | 0.95 | 7.08 | 60.48 | 68.51 |
| WFI-XGB | RF | 8 | rate, pump, gain, crc\_rate | 0.9771 | 0.9771 | 0.95 | 7.08 | 34.67 | 42.69 |
| WFI-XGB | RF | 7 | pump, solenoid, crc\_rate, rate, gain | 0.9761 | 0.9761 | 0.95 | 7.08 | 39.13 | 47.15 |
| WFI-XGB | RF | 6 | pump, control\_scheme, solenoid, crc\_rate, rate, gain | 0.9757 | 0.9757 | 0.95 | 7.08 | 47.98 | 56.01 |
| WFI-RF | RF | 9 | rate, pump, control\_scheme | 0.9741 | 0.9741 | 7.92 | 7.00 | 71.97 | 86.90 |
| WFI-RF | RF | 8 | rate, pump, solenoid, control\_scheme | 0.9718 | 0.9718 | 7.92 | 7.00 | 57.28 | 72.21 |
| WFI-RF | RF | 4 | system\_mode, pump, control\_scheme, solenoid, crc\_rate, cycle\_time, rate, gain | 0.971 | 0.971 | 7.92 | 7.00 | 63.74 | 78.67 |
| WFI-XGB | RF | 5 | pump, deadband, control\_scheme, solenoid, crc\_rate, rate, gain | 0.9703 | 0.9703 | 0.95 | 7.08 | 39.51 | 47.54 |
| WFI-RF | RF | 7 | system\_mode, pump, control\_scheme, solenoid, rate | 0.9696 | 0.9696 | 7.92 | 7.00 | 64.97 | 79.90 |
| WFI-XGB | XGB | 9 | rate, solenoid, crc\_rate | 0.9673 | 0.9673 | 0.95 | 11.22 | 7.85 | 20.03 |
| WFI-RF | RF | 6 | system\_mode, pump, control\_scheme, solenoid, rate, gain | 0.9672 | 0.9672 | 7.92 | 7.00 | 68.45 | 83.38 |
| WFI-RF | RF | 5 | system\_mode, pump, control\_scheme, solenoid, cycle\_time, rate, gain | 0.9643 | 0.9643 | 7.92 | 7.00 | 74.40 | 89.33 |
| WFI-RF | RF | 3 | system\_mode, pump, deadband, control\_scheme, solenoid, crc\_rate, cycle\_time, rate, gain | 0.9642 | 0.9642 | 7.92 | 7.00 | 43.93 | 58.86 |
| WFI-XGB | XGB | 9 | rate, gain, crc\_rate | 0.9639 | 0.9639 | 0.95 | 1.16 | 8.42 | 10.53 |
| WFI-RF | XGB | 9 | rate, pump, control\_scheme | 0.9628 | 0.9628 | 7.92 | 1.48 | 9.76 | 19.17 |
| WFI-XGB | XGB | 8 | rate, pump, gain, crc\_rate | 0.9627 | 0.9627 | 0.95 | 1.16 | 8.34 | 10.44 |
| WFI-RF | XGB | 8 | rate, pump, solenoid, control\_scheme | 0.9623 | 0.9623 | 7.92 | 1.48 | 9.14 | 18.55 |
| WFI-XGB | XGB | 7 | pump, solenoid, crc\_rate, rate, gain | 0.961 | 0.961 | 0.95 | 1.16 | 7.98 | 10.08 |
| WFI-XGB | XGB | 6 | pump, control\_scheme, solenoid, crc\_rate, rate, gain | 0.9598 | 0.9598 | 0.95 | 1.16 | 7.53 | 9.63 |
| WFI-RF | XGB | 7 | system\_mode, pump, control\_scheme, solenoid, rate | 0.9595 | 0.9595 | 7.92 | 1.48 | 9.27 | 18.68 |

The analysis of model performance with a limited number of features focused on identifying configurations that maintain high classification accuracy and efficiency despite a reduced feature set. The results confirm that Random Forest (RF) consistently outperformed XGBoost (XGB), even when working with fewer than nine features. The highest-ranked model in this category used RF with WFI-XGB ranking, which retained nine features and achieved an F1 score of 0.9795, outperforming all other models. In contrast, the best XGB model, which also used nine features, achieved an F1 score of only 0.9673, further emphasizing RF’s advantage in this setting.

The best-performing models generally retained between eight and nine features, with WFI-XGB ranking proving to be the most effective selection method. This method consistently led to the top-ranked models across both RF and XGB classifiers. Notably, an RF model with only six features achieved an F1 score of 0.9757, demonstrating that some degree of feature reduction can be beneficial without significantly compromising accuracy. However, once the number of features was reduced below seven, performance began to drop sharply, indicating that certain key features are essential for maintaining predictive strength.

A particularly striking result is that an RF model with only three features achieved an F1 score of 0.9642, outperforming multiple XGB models that retained as many as nine features. This highlights RF’s ability to maintain classification performance even under extreme feature constraints, whereas XGB models with similar or greater feature counts failed to achieve comparable scores. This reinforces the conclusion that RF is a more robust classifier in this dataset, particularly when the number of available features is highly limited.

The most frequently removed features among the top-performing models included Control Scheme, Solenoid, and Pump, suggesting that these variables contribute little to predictive accuracy. Their consistent exclusion in high-ranking configurations indicates that they were either redundant or carried minimal useful signal compared to other features.

Although XGB remained competitive, especially in efficiency, RF models consistently achieved higher F1 scores. Even when using only six or seven features, RF models maintained F1 scores above 0.96, whereas XGB models struggled to reach comparable levels even with nine features. Additionally, RF required significantly more computational time, particularly for feature selection and evaluation, with some configurations requiring over 200 seconds, whereas similar XGB models completed in under 70 seconds.

These findings confirm that Random Forest is the dominant model in feature-limited scenarios, delivering superior classification accuracy across all configurations. However, XGB remains a viable option where computational efficiency is a primary concern, as it consistently required less processing time while still maintaining reasonable predictive performance. The results demonstrate that, despite the advantages of XGB’s efficiency, RF’s ability to maintain high accuracy with fewer features makes it the superior choice for maximizing predictive power in constrained feature environments.

### Performance on Time Taken

To evaluate the efficiency of model configurations, we analyzed the time taken for ranking, feature selection, and evaluation. Initially, models were ranked by F1 score, and the top 13 best-performing models were then sorted by the smallest total time taken. This ensures a focus on models that not only deliver high classification performance but also maintain computational efficiency, which is essential for real-time applications.

Table 11 Ranking of Model Configurations by Total Computational Time (Best 13 F1 Scores)

(micro averaging)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Rank Method | Feature Selection | Model Type | Number of Features | Removed Features | F1 Score | Ranking Time | Feature Selection Time | Evaluation Time | Total Time |
| WFI-XGB | rem\_4 | RF | 8 | rate, pump, gain, crc\_rate | 0.9771 | 0.95 | 7.08 | 34.67 | 42.69 |
| WFI-XGB | rem\_5 | RF | 7 | pump, solenoid, crc\_rate, rate, gain | 0.9761 | 0.95 | 7.08 | 39.13 | 47.15 |
| WFI-XGB | rem\_6 | RF | 6 | pump, control\_scheme, solenoid, crc\_rate, rate, gain | 0.9757 | 0.95 | 7.08 | 47.98 | 56.01 |
| WFI-XGB | rem\_2 | RF | 10 | rate, crc\_rate | 0.9811 | 0.95 | 7.08 | 57.85 | 65.87 |
| WFI-XGB | rem\_3 | RF | 9 | rate, gain, crc\_rate | 0.9795 | 0.95 | 7.08 | 60.48 | 68.51 |
| WFI-XGB | rem\_1 | RF | 11 | crc\_rate | 0.9798 | 0.95 | 7.08 | 61.04 | 69.06 |
| SP | None | RF | 12 |  | 0.9766 | 0.09 | 6.71 | 62.53 | 69.32 |
| WFI-XGB | None | RF | 12 |  | 0.9766 | 0.95 | 7.08 | 62.88 | 70.91 |
| WFI-RF | None | RF | 12 |  | 0.9766 | 7.92 | 7.00 | 61.48 | 76.41 |
| SP | RFE | RF | 11 | crc\_rate | 0.9798 | 0.09 | 78.20 | 51.87 | 130.16 |
| WFI-XGB | RFE | RF | 10 | solenoid, crc\_rate | 0.9802 | 0.95 | 76.17 | 53.17 | 130.29 |
| WFI-RF | RFE | RF | 11 | crc\_rate | 0.9806 | 7.92 | 85.58 | 51.71 | 145.21 |
| WFI-RF | SPFS | RF | 12 |  | 0.9768 | 7.92 | 79.50 | 63.25 | 150.67 |

The fastest model overall was WFI-XGB with rem\_4, which removed rate, pump, gain, and crc\_rate, achieving an F1 score of 0.9771 with a total computation time of 42.69 seconds. Similarly, models using rem\_5 and rem\_6 also performed efficiently, keeping the total computation time below 56 seconds while maintaining F1 scores above 0.975. The highest-ranked model in terms of F1 score (WFI-XGB rem\_2, F1 = 0.9811) required 65.87 seconds for full execution, demonstrating that balancing feature selection and model efficiency is crucial.

Feature selection played a significant role in determining total processing time. Simpler feature selection methods, such as rem\_4, rem\_5, and rem\_6, resulted in shorter total computation times, ranging from 42 to 56 seconds, while keeping F1 scores competitive. In contrast, more complex feature selection methods, such as Recursive Feature Elimination (RFE) and Sequential Progressive Feature Selection (SPFS), required significantly longer processing times, often exceeding 130 seconds, despite offering only marginal improvements in F1 scores. Additionally, Random Forest models generally required longer computation times, particularly for feature selection and evaluation. For instance, the WFI-RF model using SPFS took 150.67 seconds, whereas comparable XGBoost (XGB) models completed in under 70 seconds.

The findings confirm that XGBoost consistently outperformed Random Forest in computational efficiency, maintaining strong performance while requiring less time. The best trade-off models combined XGBoost with lightweight feature selection techniques, such as rem\_4 to rem\_6, achieving high accuracy within 42–56 seconds. On the other hand, models utilizing RFE and SPFS delivered slightly higher F1 scores but at the cost of significantly increased processing time, often exceeding 130 seconds.

Overall, feature selection plays a crucial role in determining total computational time. The findings suggest that for real-time applications, avoiding complex feature selection methods like RFE and SPFS can significantly reduce processing time while maintaining competitive accuracy. The most optimal models combined XGBoost with rem\_4, rem\_5, or rem\_6, delivering high F1 scores while keeping processing times under 60 seconds, making them ideal for real-time intrusion detection and similar applications where efficiency is paramount.

### Model Comparison

A direct comparison between XGBoost and Random Forest is shown in Table 12, which highlights the significant difference in computational time. Even though RF models achieved relatively high accuracy, their total computation time was drastically higher than XGBoost models. For instance, an RF model using RFE with 13 features required 576.12 seconds, compared to a similar XGB model requiring only 162.08 seconds—over three times faster.

Table Time comparison between RF and XGBoost

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Rank Method | Feature Selection | Model Type | Average Type | Number of Features | Accuracy | F1 Score | Ranking Time | Feature Selection Time | Evaluation Time | Total Time |
| WFI-XGB | RFE | RF | micro | 10 | 0.9802 | 0.9802 | 0.95 | 76.17 | 53.17 | 130.29 |
| WFI-XGB | SPFS | RF | micro | 12 | 0.9746 | 0.9746 | 0.95 | 66.61 | 60.26 | 127.82 |
| WFI-XGB | RFE | XGB | micro | 9 | 0.9673 | 0.9673 | 0.95 | 11.22 | 7.85 | 20.03 |
| WFI-XGB | SPFS | XGB | micro | 12 | 0.9662 | 0.9662 | 0.95 | 13.14 | 9.95 | 24.04 |

These results confirm that high-performance models can be selected based on both F1 score and efficiency, ensuring an optimal balance between accuracy and computational speed. XGBoost consistently maintained superior efficiency, making it the preferred choice for real-time applications such as intrusion detection systems, where timely responses are as important as predictive performance. While RF models can still provide good accuracy, their much higher computational costs make them less suitable for time-sensitive applications.

## Summary of Findings

This study evaluates the impact of feature selection, classifier choice, and computational efficiency on model performance for intrusion detection in gas pipeline SCADA systems, focusing on the Response dataset, which contains 12 features. The results confirm that models retaining 8 to 10 features achieved the highest F1 scores, balancing predictive accuracy and computational efficiency. Feature reduction below 6 features led to a significant decline in performance, reinforcing the importance of retaining key predictive attributes. While removing redundant features improved efficiency, eliminating too many features degraded classification accuracy.

Feature selection techniques played a critical role in determining model performance. Recursive Feature Elimination (RFE) and Weighted Feature Importance (WFI) consistently yielded the best results, with RFE slightly outperforming other selection methods. Removal-based techniques, such as rem\_2 and rem\_3, demonstrated that some features could be discarded without a major loss in accuracy. However, when more than 6 features were removed, a clear performance decline was observed, confirming the importance of certain key attributes in the dataset.

The results also highlight differences in classifier performance. Random Forest (RF) outperformed XGBoost (XGB) across most feature selection methods, particularly in scenarios where fewer than 9 features were retained. However, XGB demonstrated superior computational efficiency, completing training and evaluation in significantly less time than RF models. While XGB remained competitive in terms of F1 score, its best configuration (F1 = 0.9673) did not match the top RF models (F1 = 0.9811), indicating that RF maintains superior classification accuracy when working with limited features.

### Key Observations

The feature ranking analysis revealed that Weighted Feature Importance (WFI) and Statistical Properties (SP) were among the most effective methods for identifying the most informative features. Models utilizing WFI-based feature selection consistently achieved high F1 scores, particularly when paired with Random Forest, demonstrating the effectiveness of ranking-guided feature elimination. The best-performing model recorded an F1 score of 0.9811 using WFI-XGB with RF, reinforcing the importance of strategic feature selection in optimizing predictive performance.

Computational efficiency varied significantly between XGBoost and Random Forest. XGBoost consistently required less processing time, particularly when working with fewer than 9 features. In contrast, RF models took significantly longer to train and evaluate, with some high-F1 configurations requiring over 80 seconds, whereas comparable XGB models completed in under 20 seconds. This highlights the trade-off between RF’s superior classification accuracy and XGB’s faster computation times, making XGB more suitable for time-sensitive applications.

Feature removal analysis showed that eliminating low-importance features such as Control Scheme, Solenoid, and Pump had minimal impact on model performance. These features were consistently discarded in high-ranking models, suggesting they contributed little to predictive accuracy. However, removing more than 6 features led to a gradual decline in classification performance, confirming that certain attributes play a crucial role in maintaining model effectiveness.

An important finding was RF’s resilience to extreme feature constraints. A Random Forest model with only 3 features (F1 = 0.9642) still outperformed multiple XGBoost models retaining 9 features, highlighting RF’s ability to maintain strong classification accuracy despite aggressive feature reduction. On the other hand, XGBoost models struggled to achieve comparable performance even when more features were retained, reinforcing RF’s robustness in feature-limited scenarios. These findings confirm that while XGBoost offers efficiency, Random Forest remains the superior choice for maintaining high classification accuracy in this dataset.

### Summary of Best-Performing Configurations

The best-performing configurations focused on using RFE and SPFS feature selection methods with the Random Forest model. The top-ranked models consistently achieved F1 scores around 0.98, with the highest F1 score at 0.9806 obtained using WFI-RF combined with RFE and a Random Forest model. The graph below visualizes the best-performing configurations, illustrating that models retaining 10 to 11 features generally performed the best.

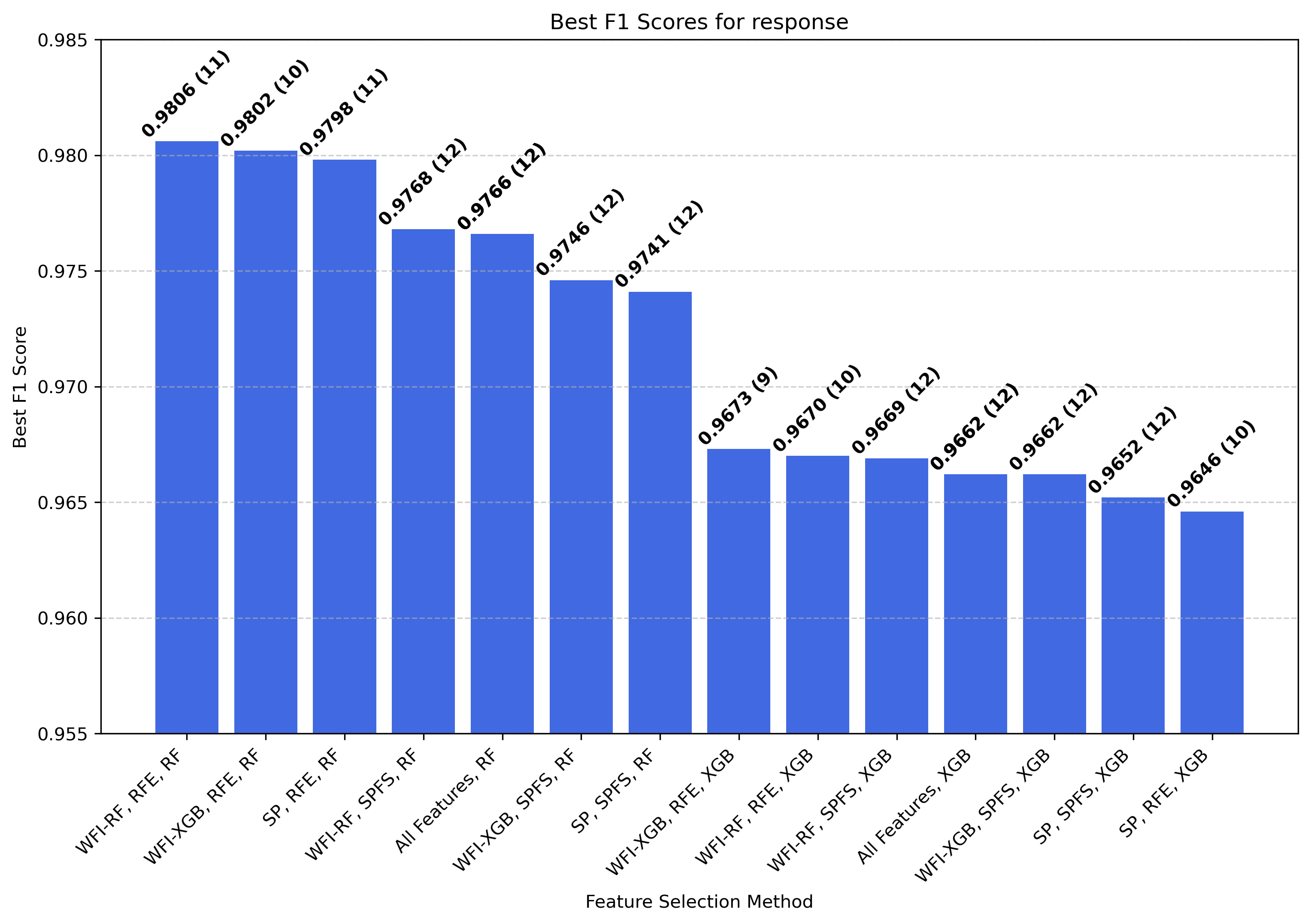


Figure Graph showing all combinations best F1-Score

The most effective configurations were those that balanced feature selection and ranking methods while maintaining computational efficiency. Models that retained 10 or 11 features using WFI-RF and RFE consistently outperformed others, achieving both high accuracy and efficiency. While RF-based models achieved a greater F1 score on average, their significantly higher computational time makes them less suitable for real-time applications.

The results validate the effectiveness of a structured feature selection approach, confirming that an optimal balance of features, ranking methods, and classifiers leads to superior model performance. These findings serve as a guideline for designing efficient machine learning models in SCADA intrusion detection systems, ensuring both accuracy and computational feasibility.

# Conclusion

This study applied the RS-BMLMS (Robust Strategy for Best Machine Learning Model Selection) methodology to enhance intrusion detection in gas pipeline SCADA systems. Key findings emphasize the importance of effectively combining feature selection, ranking methods, and classifiers to optimize accuracy and computational efficiency. Optimal performance was achieved using models retaining 8 to 10 features, with Recursive Feature Elimination (RFE) and Sequential Progressive Feature Selection (SPFS) proving most effective. Removal-based methods (e.g., rem\_2 to rem\_4) demonstrated that some features could be safely removed; however, eliminating more than six features caused significant accuracy drops.

Classifier comparisons showed Random Forest (RF) consistently achieved higher predictive accuracy and F1 scores compared to XGBoost (XGB), even under substantial feature reduction. RF displayed remarkable resilience, maintaining high accuracy with as few as three features, whereas XGB experienced notable performance declines with reduced features. Nevertheless, XGB demonstrated significantly better computational efficiency, making it ideal for real-time detection scenarios.

Feature ranking methods such as Weighted Feature Importance (WFI) and Statistical Properties (SP) were highly effective, particularly when integrated with RFE-based feature selection techniques, leading to the highest F1 score (0.9811). Features like Control Scheme, Solenoid, and Pump consistently had minimal predictive impact and were frequently removed without harming accuracy.

Overall, the RS-BMLMS methodology provided a structured, repeatable framework that optimized intrusion detection model performance, highlighting RF’s accuracy advantage and robustness in feature-limited scenarios, and XGB’s strength in computational speed. Future research could further refine these approaches by exploring additional feature-ranking and optimization techniques.

# Scope of Future Researche

<Discuss the scope of future research and identify how this particular research work can be extended in future>

# References

The following is an example of References for this document – replace with the appropriate list of References that have been referred to in this document.

1. Akyildiz, I. F., W. Su, Y. Sankarasubramaniam and E. Cayirci. “A survey on sensor networks”. *IEEE Commun. Mag*., 40 (8), 2002, pp. 102-114.

2. Bartolini, N. T. et al. “Snap and spread: a self-deployment algorithm for sensor networks”. *Lecture Notes on Comput. Sci.,* 5067, 2008, pp. 451-456.

3. Fletcher, G., “Placement and relocation of wireless sensor nodes by a team of robots”, Masters Thesis, University of Ottawa, 2010.

4. Ma, M. and Yang, Y. “Adaptive triangular deployment algorithm for unattended mobile sensor networks”. *IEEE Trans. Comput.,* 56 (7), 2007, pp. 946-958.

# Appendix A: xxx

Link to GitHub: <https://github.com/DanielGoel/FeatureSelectionML.git>

main.py: Loops through all possible combinations of Ranking, Feature selection and model testing using ranking.py, feature\_selection.py and model\_testing.py.

|  |
| --- |
| import os  import pandas as pd  from ranking import DataAnalysisRanker  from feature\_selection import FeatureSelector  from model\_testing import ModelEvaluator  # List of CSV files to process  csv\_files = {      "function": "GaspipelineDatasets/NewGasFilteredFunctionMinMax.csv",      "command": "GaspipelineDatasets/NewGasFilteredCommandMinMax.csv",      "all": "GaspipelineDatasets/NewGasFilteredAllMinMax.csv",      "response": "GaspipelineDatasets/NewGasFilteredResponseNNNoOHEMulti.csv"  }  model\_dir = 'results/models/'  ranking\_methods = ["SP", "WFI-RF", "WFI-XGB"]  # Added WFI-RF and WFI-XGB  use\_std\_dev = use\_abs\_diff = use\_skewness = use\_kurtosis = True  for dataset\_name, input\_file in csv\_files.items():      for ranking\_method in ranking\_methods:          ranking\_file = f'results/Rankings/Ranking\_{ranking\_method}\_{dataset\_name}.csv'            # Step 1: Rank features          ranker = DataAnalysisRanker(ranking\_file, input\_file, ranking\_file)          ranker.analyze(ranking\_method, use\_std\_dev, use\_abs\_diff, use\_skewness, use\_kurtosis)          ranker.save\_results()            # Load ranked features          rankings\_df = pd.read\_csv(ranking\_file)          ranked\_features = rankings\_df.iloc[:, 0].tolist()          total\_features = len(ranked\_features)          print(f"Processing dataset: {dataset\_name} with {total\_features} features using {ranking\_method}.")            for model\_type in ["RF", "XGB"]:              for feature\_selection\_method in ["RFE", "SPFS", "None"]:                  # Step 2: Select Features                  selector = FeatureSelector(dataset\_name, ranking\_file, input\_file)                  selector.perform\_feature\_selection(model\_type, feature\_selection\_method)                    model\_filename = f"{dataset\_name}\_{model\_type}\_{feature\_selection\_method}.pkl"                  model\_path = os.path.join(model\_dir, model\_filename)                    for average\_type in ["micro", "macro", "weighted"]:                      evaluator = ModelEvaluator(input\_file, average\_type, selector.selected\_features, model\_type, feature\_selection\_method, model\_path, ranking\_method)                      evaluator.train\_and\_evaluate()                # Step 3: Remove 1 to (total\_features - 1) features iteratively              for num\_remove in range(1, total\_features):                  selected\_features = ranked\_features[:-num\_remove]                  feature\_selection\_method = f"rem\_{num\_remove}"                  model\_filename = f"{model\_type}\_{feature\_selection\_method}.pkl"                  model\_path = os.path.join(model\_dir, model\_filename)                    selector.selected\_features = selected\_features                  selector.train\_and\_save\_model(selected\_features, model\_type, model\_path)                    for average\_type in ["micro", "macro", "weighted"]:                      evaluator = ModelEvaluator(input\_file, average\_type, selected\_features, model\_type, feature\_selection\_method, model\_path, ranking\_method)                      evaluator.train\_and\_evaluate()        print(f"Completed processing for {dataset\_name} dataset.\n") |

ranking.py: Ranks feature importance using weighted feature importance and statistical properties

|  |
| --- |
| import pandas as pd  import numpy as np  from scipy.stats import skew, kurtosis  from xgboost import XGBClassifier  from sklearn.ensemble import RandomForestClassifier  from sklearn.preprocessing import LabelEncoder  class DataAnalysisRanker:      def \_\_init\_\_(self, ranking\_file, input\_file, output\_file):          self.ranking\_file = ranking\_file          self.input\_file = input\_file          self.output\_file = output\_file          self.df = pd.read\_csv(input\_file)          self.results\_df = None      def compute\_standard\_deviation(self):          return self.df.std()      def compute\_abs\_diff\_mean\_median(self):          return np.abs(self.df.mean() - self.df.median())      def compute\_skewness(self):          return self.df.apply(skew)      def compute\_kurtosis(self):          return self.df.apply(kurtosis)      def analyze(self, rank\_method, use\_std\_dev, use\_abs\_diff, use\_skewness, use\_kurtosis):          metrics = {}          ranks = {}          if rank\_method == 'WFI-RF':              feature\_importances\_df = self.wfi\_RandomForest()              feature\_importances\_df = feature\_importances\_df.set\_index('Name')              self.df = self.df[feature\_importances\_df.index]            elif rank\_method == 'WFI-XGB':              feature\_importances\_df = self.wfi\_XGBoost()              feature\_importances\_df = feature\_importances\_df.set\_index('Name')              self.df = self.df[feature\_importances\_df.index]            else:              if 'Label' in self.df.columns:                  self.df = self.df.drop(columns=['Label'])              if use\_std\_dev:                  metrics['Standard Deviation'] = self.compute\_standard\_deviation()                  ranks['Standard Deviation Rank'] = metrics['Standard Deviation'].rank(ascending=True)              if use\_abs\_diff:                  metrics['Absolute Difference'] = self.compute\_abs\_diff\_mean\_median()                  ranks['Absolute Difference Rank'] = metrics['Absolute Difference'].rank(ascending=True)              if use\_skewness:                  metrics['Skewness'] = self.compute\_skewness()                  ranks['Skewness Rank'] = metrics['Skewness'].rank(ascending=False)              if use\_kurtosis:                  metrics['Kurtosis'] = self.compute\_kurtosis()                  ranks['Kurtosis Rank'] = metrics['Kurtosis'].rank(ascending=True)              # Combine all metrics and ranks into a single DataFrame              OutputData = {\*\*metrics, \*\*ranks}              self.results\_df = pd.DataFrame({                  'Name': self.df.columns,                  \*\*OutputData              })              if ranks:                  total\_rank = sum(ranks.values())                  self.results\_df['Total Rank'] = total\_rank                  self.feature\_importances\_df = self.results\_df.sort\_values('Total Rank', ascending=True)          self.feature\_importances\_df = self.feature\_importances\_df.reset\_index()          #print(feature\_importances\_df)          self.feature\_importances\_df = self.feature\_importances\_df.set\_index('Name')          self.save\_results()        def wfi\_RandomForest(self):          target\_column = 'Label'          y = self.df[target\_column]          X = self.df.drop(columns=[target\_column])          print(X.columns)          model = RandomForestClassifier(random\_state=42)          model.fit(X, y)          self.feature\_importances = model.feature\_importances\_          self.feature\_importances\_df = pd.DataFrame({              'Name': X.columns,              'Random Forest Feature Importance': self.feature\_importances          }).sort\_values(by='Random Forest Feature Importance', ascending=False)  # Sorting by importance          return self.feature\_importances\_df        def wfi\_XGBoost(self):          target\_column = 'Label'          y = self.df[target\_column]          X = self.df.drop(columns=[target\_column])            # Convert class labels to a sequential range          label\_encoder = LabelEncoder()          y\_encoded = label\_encoder.fit\_transform(y)  # Converts [0, 3, 4, 5, 6, 7] → [0, 1, 2, 3, 4, 5]          model = XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss', random\_state=42)          model.fit(X, y\_encoded)  # Use encoded labels          self.feature\_importances = model.feature\_importances\_          self.feature\_importances\_df = pd.DataFrame({              'Name': X.columns,              'XGBoost Feature Importance': self.feature\_importances          }).sort\_values(by='XGBoost Feature Importance', ascending=False)          return self.feature\_importances\_df.sort\_values(by='XGBoost Feature Importance', ascending=False)      def save\_results(self):          df\_to\_save = self.feature\_importances\_df          df\_to\_save.to\_csv(self.ranking\_file, index=True)          print(f"Results saved to {self.ranking\_file}") |

feature\_selection.py: Selects important features using RFE and SPFS

|  |
| --- |
| import pandas as pd  import os  import xgboost as xgb  from sklearn.ensemble import RandomForestClassifier  from sklearn.model\_selection import train\_test\_split  import time  from sklearn.metrics import f1\_score  import pickle  class FeatureSelector:      def \_\_init\_\_(self, dataset\_name, ranking\_file, input\_file):          self.ranking\_file = ranking\_file          self.input\_file = input\_file          self.dataset\_name = dataset\_name          self.df = pd.read\_csv(input\_file)          self.selected\_features = []          self.metrics\_dir = "results/metrics/"          self.model\_dir = "results/models/"          os.makedirs(self.metrics\_dir, exist\_ok=True)          os.makedirs(self.model\_dir, exist\_ok=True)          # Load feature ranking order          self.ranked\_features = self.load\_ranked\_features()      def load\_ranked\_features(self):          rankings\_df = pd.read\_csv(self.ranking\_file)          ranked\_features = rankings\_df.iloc[:, 0].tolist()  # First column contains feature names          return [feature for feature in ranked\_features if feature in self.df.columns]  # Keep only valid features      def perform\_feature\_selection(self, model\_type, method):          if method == "RFE":              self.perform\_rfe(model\_type)          elif method == "SPFS":              self.perform\_spfs(model\_type)          else:              print("Skipped Feature Selection. Using all features.")              self.perform\_baseline(model\_type)      def perform\_rfe(self, model\_type):          target\_column = 'Label'            if target\_column not in self.df.columns:              raise KeyError(f"Column '{target\_column}' not found in dataset!")          y = self.df[target\_column]          X = self.df.drop(columns=[target\_column])  # Use ranked feature order          # Select Model Type          if model\_type == 'XGB':              y = y.astype('category').cat.codes              model = xgb.XGBClassifier(seed=42)          else:              model = RandomForestClassifier(random\_state=42)          selected\_features = self.ranked\_features.copy()  # Start with all ranked features          best\_f1\_score = 0          print("Starting Recursive Feature Elimination...")          # Evaluate the model with all features first          X\_train, X\_test, y\_train, y\_test = train\_test\_split(X[selected\_features], y, test\_size=0.2, random\_state=42)          model.fit(X\_train, y\_train)          y\_pred = model.predict(X\_test)          best\_f1\_score = self.get\_f1\_score(y\_test, y\_pred)['F1 Score']          print(f"Initial F1 Score with all features: {best\_f1\_score:.4f}")          for feature in reversed(self.ranked\_features):              if feature in selected\_features:                  temp\_features = selected\_features.copy()                  temp\_features.remove(feature)                  print(f"Testing without feature: {feature}")                  # Train with updated feature set                  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X[temp\_features], y, test\_size=0.2, random\_state=42)                  model.fit(X\_train, y\_train)                  y\_pred = model.predict(X\_test)                  current\_f1\_score = self.get\_f1\_score(y\_test, y\_pred)['F1 Score']                  if current\_f1\_score >= best\_f1\_score:                      best\_f1\_score = current\_f1\_score                      selected\_features = temp\_features                      print(f"Removing {feature} improved or maintained F1 Score to {current\_f1\_score:.4f}")                  else:                      print(f"Keeping {feature} as removing it decreased F1 Score to {current\_f1\_score:.4f}")          # Save final RFE model          self.selected\_features = selected\_features          self.save\_final\_model(model, model\_type, "RFE")          print(f"Final selected features: {self.selected\_features}")          print(f"Final F1 Score: {best\_f1\_score:.4f}")      def perform\_spfs(self, model\_type):          self.selected\_features = []          target\_column = 'Label'            if target\_column not in self.df.columns:              raise KeyError(f"Column '{target\_column}' not found in dataset!")          y = self.df[target\_column]          X = self.df.drop(columns=[target\_column])          metrics\_data = []          print("Starting SPFS...")          self.ranked\_features = [feature for feature in self.ranked\_features if feature != target\_column]          for feature in self.ranked\_features:              self.selected\_features.append(feature)              X\_selected = X[self.selected\_features]              X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=42)              if model\_type == 'XGB':                  y\_train = y\_train.astype('category').cat.codes                  y\_test = y\_test.astype('category').cat.codes                  model = xgb.XGBClassifier(seed=42)              else:                  model = RandomForestClassifier(random\_state=42)              model.fit(X\_train, y\_train)              y\_pred = model.predict(X\_test)              f1\_score = self.get\_f1\_score(y\_test, y\_pred)['F1 Score']              if f1\_score > max([m['F1 Score'] for m in metrics\_data] or [0]):                  print(f"Feature {feature} kept. New best F1 score = {f1\_score:.4f}\n")              else:                  self.selected\_features.remove(feature)                  print(f"Feature {feature} removed. F1 score did not improve.")          # Save final SPFS model          self.save\_final\_model(model, model\_type, "SPFS")          print(f"Final selected features: {self.selected\_features}")        def perform\_baseline(self, model\_type):          # Use all features except 'Label'          self.selected\_features = self.df.columns.tolist()          self.selected\_features.remove('Label')          # Extract features and target          X = self.df[self.selected\_features]          y = self.df['Label']          # Encode y if using XGBoost          if model\_type.lower() == "xgb":              y = y.astype('category').cat.codes  # Convert labels to numerical codes              model = xgb.XGBClassifier(seed=42)          else:              model = RandomForestClassifier(random\_state=42)          # Split data (80% train, 20% test)          X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)          # Train the model          model.fit(X\_train, y\_train)          # Save the trained model          model\_path = f"results/models/{self.dataset\_name}\_{model\_type}\_None.pkl"          os.makedirs(os.path.dirname(model\_path), exist\_ok=True)            with open(model\_path, 'wb') as model\_file:              pickle.dump(model, model\_file)          print(f"✅ Baseline Model ({model\_type}) saved to {model\_path}")        def get\_f1\_score(self, y\_true, y\_pred):          f1 = f1\_score(y\_true, y\_pred, average='weighted')          return {'F1 Score': f1}      def save\_final\_model(self, model, model\_type, method):          model\_filename = f"{self.dataset\_name}\_{model\_type}\_{method}.pkl"          model\_path = os.path.join(self.model\_dir, model\_filename)          with open(model\_path, 'wb') as f:              pickle.dump(model, f)          print(f"Model saved to {model\_path}")      def train\_and\_save\_model(self, selected\_features, model\_type, model\_path):          print(f"Training model using selected features: {selected\_features}")          target\_column = 'Label'            y = self.df[target\_column]          X = self.df[selected\_features]  # Use selected features!            # Select model type          if model\_type.lower() == "xgb":              y = y.astype('category').cat.codes              model = xgb.XGBClassifier(seed=42)          else:              model = RandomForestClassifier(random\_state=42)            # Split dataset          X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)            # Train model          model.fit(X\_train, y\_train)            # Save model          os.makedirs(os.path.dirname(model\_path), exist\_ok=True)          with open(model\_path, 'wb') as f:              pickle.dump(model, f)            print(f"✅ Model saved to {model\_path}") |

model\_testing.py: Cross-validation to evaluate the model efficiency

|  |
| --- |
| import pandas as pd  import os  import pickle  import time  from sklearn.model\_selection import train\_test\_split, KFold  from sklearn.metrics import f1\_score, precision\_score, recall\_score, accuracy\_score, confusion\_matrix  class ModelEvaluator:      def \_\_init\_\_(self, input\_file, average\_type, selected\_features, model\_type, feature\_selection\_method, model\_path, rank\_method):          self.rank\_method = rank\_method          self.input\_file = input\_file          self.feature\_selection\_method = feature\_selection\_method          self.selected\_features = selected\_features          self.average\_type = average\_type          self.model\_type = model\_type          self.model\_path = model\_path          self.df = pd.read\_csv(input\_file)          self.target\_column = "Label"          self.metrics\_log\_file = "results/metrics/evaluation\_log.csv"          self.removed\_features = list(set(self.df.columns) - set(self.selected\_features)- {self.target\_column})        def load\_model(self):          """Load pre-trained model from a file"""          if not os.path.exists(self.model\_path):              raise FileNotFoundError(f"Model file not found at {self.model\_path}")          with open(self.model\_path, 'rb') as model\_file:              model = pickle.load(model\_file)          return model      def evaluate\_model(self, y\_true, y\_pred):          accuracy = accuracy\_score(y\_true, y\_pred)          precision = precision\_score(y\_true, y\_pred, average=self.average\_type, zero\_division=0)          recall = recall\_score(y\_true, y\_pred, average=self.average\_type, zero\_division=0)          f1 = f1\_score(y\_true, y\_pred, average=self.average\_type)          cm = confusion\_matrix(y\_true, y\_pred)          fpr = cm[0][1] / cm[0].sum() if cm[0].sum() > 0 else 0          fnr = cm[1][0] / cm[1].sum() if cm[1].sum() > 0 else 0          return {              'Accuracy': accuracy,              'Precision': precision,              'Recall': recall,              'F1 Score': f1,              'FPR': fpr,              'FNR': fnr          }      def train\_and\_evaluate(self):          """Perform evaluation using 10-Fold Cross-Validation"""          start\_time = time.time()            if self.feature\_selection\_method == "none":              self.selected\_features = self.df.columns.tolist()              self.selected\_features.remove(self.target\_column)  # Ensure target is removed            X = self.df[self.selected\_features]          y = self.df[self.target\_column]          # Ensure y is encoded correctly for XGBoost          if "xgboost" in self.model\_type.lower():              y = y.astype('category').cat.codes  # Convert labels to numerical format          kf = KFold(n\_splits=10, shuffle=True, random\_state=42)          model = self.load\_model()          metrics\_data = []          print("Performing 10-Fold Cross-Validation...")            for fold, (train\_index, val\_index) in enumerate(kf.split(X, y), start=1):              X\_train, X\_val = X.iloc[train\_index], X.iloc[val\_index]              y\_train, y\_val = y.iloc[train\_index], y.iloc[val\_index]                # Ensure feature names match the ones used during training              if hasattr(model, "feature\_names\_in\_"):                  X\_val = X\_val[model.feature\_names\_in\_]              # Use pre-trained model for prediction              y\_pred = model.predict(X\_val)              metrics = self.evaluate\_model(y\_val, y\_pred)              metrics\_data.append(metrics)              print(f"Fold {fold}: F1 Score = {metrics['F1 Score']:.4f}")          # Compute average metrics          avg\_metrics = pd.DataFrame(metrics\_data).mean(numeric\_only=True)          total\_time\_taken = time.time() - start\_time          # Create a log entry          log\_entry = pd.DataFrame({              'Input File': [os.path.basename(self.input\_file)],              'Rank Method': [self.rank\_method],              'Feature Selection': [self.feature\_selection\_method],              'Model Type': [self.model\_type],              'Average Type': [self.average\_type],              'Number of Features': [len(self.selected\_features)],              'Removed Features': [", ".join(self.removed\_features)],              'Accuracy': [avg\_metrics['Accuracy']],              'Precision': [avg\_metrics['Precision']],              'Recall': [avg\_metrics['Recall']],              'F1 Score': [avg\_metrics['F1 Score']],              'FPR': [avg\_metrics['FPR']],              'FNR': [avg\_metrics['FNR']],              'Time Taken': [total\_time\_taken]          })          # Append results to the log file          os.makedirs(os.path.dirname(self.metrics\_log\_file), exist\_ok=True)          if not os.path.exists(self.metrics\_log\_file):              log\_entry.to\_csv(self.metrics\_log\_file, index=False)          else:              log\_entry.to\_csv(self.metrics\_log\_file, mode='a', header=False, index=False)          print(f"✅ Results appended to {self.metrics\_log\_file}") |

Full evaluation\_log.csv Results:

*Table 13 Full Evaluation Log csv file*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Rank Method | Feature Selection | Model Type | Average Type | Number of Features | Accuracy | F1 Score | Ranking Time | Feature Selection Time | Evaluation Time | Total Time |
| WFI-XGB | RFE | XGB | micro | 15 | 0.9745 | 0.9745 | 8.1783 | 98.5319 | 55.3733 | 162.0835 |
| WFI-XGB | SPFS | XGB | micro | 15 | 0.9745 | 0.9745 | 8.1783 | 103.5423 | 61.2083 | 172.929 |
| WFI-XGB | RFE | XGB | weighted | 15 | 0.9745 | 0.9738 | 8.1783 | 98.5319 | 77.4295 | 184.1397 |
| WFI-XGB | SPFS | XGB | weighted | 15 | 0.9745 | 0.9738 | 8.1783 | 103.5423 | 57.1957 | 168.9164 |
| WFI-RF | rem\_1 | XGB | micro | 14 | 0.9735 | 0.9735 | 28.1665 | 7.3364 | 54.622 | 90.1249 |
| WFI-RF | SPFS | XGB | micro | 15 | 0.9731 | 0.9731 | 28.1665 | 86.1164 | 54.4091 | 168.692 |
| WFI-RF | None | XGB | micro | 15 | 0.9731 | 0.9731 | 28.1665 | 7.3364 | 56.1417 | 91.6446 |
| SP | SPFS | XGB | micro | 15 | 0.9731 | 0.9731 | 0.5398 | 104.1627 | 58.4338 | 163.1363 |
| SP | None | XGB | micro | 15 | 0.9731 | 0.9731 | 0.5398 | 9.7235 | 73.2484 | 83.5117 |
| WFI-XGB | None | XGB | micro | 15 | 0.9731 | 0.9731 | 8.1783 | 7.2277 | 71.6919 | 87.0979 |
| SP | RFE | XGB | micro | 14 | 0.9729 | 0.9729 | 0.5398 | 103.6918 | 57.0397 | 161.2713 |
| WFI-RF | rem\_1 | XGB | weighted | 14 | 0.9735 | 0.9727 | 28.1665 | 7.3364 | 55.2696 | 90.7725 |
| WFI-RF | RFE | XGB | micro | 14 | 0.9726 | 0.9726 | 28.1665 | 98.9481 | 57.5878 | 184.7024 |
| SP | rem\_1 | XGB | micro | 14 | 0.9726 | 0.9726 | 0.5398 | 9.7235 | 60.3108 | 70.5741 |
| WFI-RF | SPFS | XGB | weighted | 15 | 0.9731 | 0.9724 | 28.1665 | 86.1164 | 60.3455 | 174.6284 |
| WFI-RF | None | XGB | weighted | 15 | 0.9731 | 0.9724 | 28.1665 | 7.3364 | 60.0114 | 95.5143 |
| SP | SPFS | XGB | weighted | 15 | 0.9731 | 0.9724 | 0.5398 | 104.1627 | 55.8374 | 160.5399 |
| SP | None | XGB | weighted | 15 | 0.9731 | 0.9724 | 0.5398 | 9.7235 | 67.3089 | 77.5723 |
| WFI-XGB | None | XGB | weighted | 15 | 0.9731 | 0.9724 | 8.1783 | 7.2277 | 65.7008 | 81.1068 |
| SP | RFE | XGB | weighted | 14 | 0.9729 | 0.9721 | 0.5398 | 103.6918 | 55.5296 | 159.7612 |
| WFI-XGB | rem\_1 | XGB | micro | 14 | 0.9719 | 0.9719 | 8.1783 | 7.2277 | 61.0427 | 76.4487 |
| WFI-RF | RFE | XGB | weighted | 14 | 0.9726 | 0.9718 | 28.1665 | 98.9481 | 61.5052 | 188.6198 |
| SP | rem\_1 | XGB | weighted | 14 | 0.9726 | 0.9718 | 0.5398 | 9.7235 | 56.1641 | 66.4274 |
| WFI-XGB | rem\_1 | XGB | weighted | 14 | 0.9719 | 0.971 | 8.1783 | 7.2277 | 56.8606 | 72.2666 |
| SP | rem\_2 | XGB | micro | 13 | 0.9707 | 0.9707 | 0.5398 | 9.7235 | 53.381 | 63.6443 |
| SP | rem\_2 | XGB | weighted | 13 | 0.9707 | 0.9697 | 0.5398 | 9.7235 | 63.5611 | 73.8244 |
| SP | rem\_3 | XGB | micro | 12 | 0.9672 | 0.9672 | 0.5398 | 9.7235 | 59.2953 | 69.5586 |
| SP | rem\_4 | XGB | micro | 11 | 0.9671 | 0.9671 | 0.5398 | 9.7235 | 52.7595 | 63.0228 |
| SP | rem\_3 | XGB | weighted | 12 | 0.9672 | 0.9659 | 0.5398 | 9.7235 | 54.6473 | 64.9106 |
| SP | rem\_4 | XGB | weighted | 11 | 0.9671 | 0.9658 | 0.5398 | 9.7235 | 52.3542 | 62.6175 |
| WFI-XGB | rem\_2 | XGB | micro | 13 | 0.965 | 0.965 | 8.1783 | 7.2277 | 56.9024 | 72.3084 |
| WFI-XGB | rem\_3 | XGB | micro | 12 | 0.9641 | 0.9641 | 8.1783 | 7.2277 | 51.6252 | 67.0312 |
| WFI-XGB | rem\_2 | XGB | weighted | 13 | 0.965 | 0.9637 | 8.1783 | 7.2277 | 52.764 | 68.17 |
| WFI-XGB | rem\_3 | XGB | weighted | 12 | 0.9641 | 0.9627 | 8.1783 | 7.2277 | 51.7998 | 67.2057 |
| SP | RFE | RF | micro | 14 | 0.9623 | 0.9623 | 0.5398 | 388.1504 | 191.8283 | 580.5184 |
| WFI-RF | rem\_2 | RF | micro | 13 | 0.9621 | 0.9621 | 28.1665 | 23.6636 | 198.406 | 250.2361 |
| SP | rem\_5 | XGB | micro | 10 | 0.9619 | 0.9619 | 0.5398 | 9.7235 | 69.7961 | 80.0594 |
| WFI-RF | RFE | RF | micro | 14 | 0.9618 | 0.9618 | 28.1665 | 404.8508 | 226.6316 | 659.6489 |
| WFI-RF | rem\_1 | RF | micro | 14 | 0.9618 | 0.9618 | 28.1665 | 23.6636 | 222.0824 | 273.9125 |
| WFI-XGB | RFE | RF | micro | 13 | 0.9618 | 0.9618 | 8.1783 | 371.0178 | 196.9263 | 576.1224 |
| WFI-RF | None | RF | micro | 15 | 0.9617 | 0.9617 | 28.1665 | 23.6636 | 216.9796 | 268.8097 |
| SP | None | RF | micro | 15 | 0.9617 | 0.9617 | 0.5398 | 24.1702 | 215.05 | 239.76 |
| WFI-XGB | None | RF | micro | 15 | 0.9617 | 0.9617 | 8.1783 | 22.4982 | 215.966 | 246.6425 |
| SP | SPFS | RF | micro | 15 | 0.9615 | 0.9615 | 0.5398 | 338.4803 | 214.9426 | 553.9626 |
| WFI-RF | SPFS | RF | micro | 15 | 0.9614 | 0.9614 | 28.1665 | 341.6488 | 220.5252 | 590.3405 |
| SP | RFE | RF | weighted | 14 | 0.9623 | 0.9613 | 0.5398 | 388.1504 | 193.2668 | 581.957 |
| WFI-XGB | SPFS | RF | micro | 15 | 0.9612 | 0.9612 | 8.1783 | 214.3641 | 216.3696 | 438.9121 |
| WFI-RF | rem\_2 | RF | weighted | 13 | 0.9621 | 0.9611 | 28.1665 | 23.6636 | 198.8509 | 250.681 |
| WFI-XGB | RFE | RF | weighted | 13 | 0.9618 | 0.9608 | 8.1783 | 371.0178 | 196.1758 | 575.3719 |
| WFI-RF | RFE | RF | weighted | 14 | 0.9618 | 0.9607 | 28.1665 | 404.8508 | 222.2565 | 655.2738 |
| WFI-RF | rem\_1 | RF | weighted | 14 | 0.9618 | 0.9607 | 28.1665 | 23.6636 | 223.2803 | 275.1104 |
| WFI-XGB | SPFS | RF | weighted | 15 | 0.9618 | 0.9607 | 8.1783 | 214.3641 | 215.8557 | 438.3982 |
| WFI-RF | None | RF | weighted | 15 | 0.9617 | 0.9606 | 28.1665 | 23.6636 | 215.9117 | 267.7417 |
| SP | None | RF | weighted | 15 | 0.9617 | 0.9606 | 0.5398 | 24.1702 | 212.8829 | 237.5929 |
| WFI-XGB | None | RF | weighted | 15 | 0.9617 | 0.9606 | 8.1783 | 22.4982 | 215.5306 | 246.2072 |
| SP | SPFS | RF | weighted | 15 | 0.9615 | 0.9605 | 0.5398 | 338.4803 | 216.6426 | 555.6626 |
| WFI-RF | SPFS | RF | weighted | 15 | 0.9614 | 0.9604 | 28.1665 | 341.6488 | 218.278 | 588.0933 |
| SP | rem\_1 | RF | micro | 14 | 0.96 | 0.96 | 0.5398 | 24.1702 | 223.3047 | 248.0147 |
| SP | rem\_1 | RF | weighted | 14 | 0.961 | 0.9599 | 0.5398 | 24.1702 | 224.066 | 248.776 |
| SP | rem\_5 | XGB | weighted | 10 | 0.9619 | 0.9596 | 0.5398 | 9.7235 | 62.8392 | 73.1025 |
| WFI-XGB | rem\_1 | RF | micro | 14 | 0.9589 | 0.9589 | 8.1783 | 22.4982 | 223.4063 | 254.0829 |
| WFI-RF | rem\_3 | RF | micro | 12 | 0.9588 | 0.9588 | 28.1665 | 23.6636 | 206.2593 | 258.0894 |
| WFI-RF | rem\_3 | RF | weighted | 12 | 0.9588 | 0.9576 | 28.1665 | 23.6636 | 207.8593 | 259.6893 |
| WFI-XGB | rem\_1 | RF | weighted | 14 | 0.9589 | 0.9576 | 8.1783 | 22.4982 | 223.375 | 254.0516 |
| WFI-RF | rem\_4 | RF | micro | 11 | 0.9574 | 0.9574 | 28.1665 | 23.6636 | 213.8135 | 265.6436 |
| SP | rem\_2 | RF | micro | 13 | 0.9573 | 0.9573 | 0.5398 | 24.1702 | 230.227 | 254.937 |
| SP | rem\_2 | RF | weighted | 13 | 0.9573 | 0.956 | 0.5398 | 24.1702 | 232.1167 | 256.8267 |
| WFI-RF | rem\_4 | RF | weighted | 11 | 0.9566 | 0.9551 | 28.1665 | 23.6636 | 211.9905 | 263.8206 |
| WFI-RF | rem\_5 | RF | micro | 10 | 0.9548 | 0.9548 | 28.1665 | 23.6636 | 223.3571 | 275.1872 |
| SP | rem\_6 | XGB | micro | 9 | 0.9548 | 0.9548 | 0.5398 | 9.7235 | 57.1208 | 67.3841 |
| SP | rem\_3 | RF | micro | 12 | 0.9544 | 0.9544 | 0.5398 | 24.1702 | 239.3821 | 264.0921 |
| SP | rem\_4 | RF | micro | 11 | 0.9533 | 0.9533 | 0.5398 | 24.1702 | 250.9743 | 275.6843 |
| WFI-RF | rem\_5 | RF | weighted | 10 | 0.9548 | 0.9532 | 28.1665 | 23.6636 | 224.6456 | 276.4757 |
| SP | rem\_3 | RF | weighted | 12 | 0.9544 | 0.9528 | 0.5398 | 24.1702 | 238.7612 | 263.4712 |
| SP | rem\_6 | XGB | weighted | 9 | 0.9548 | 0.9518 | 0.5398 | 9.7235 | 53.8096 | 64.0729 |
| WFI-XGB | rem\_2 | RF | micro | 13 | 0.9517 | 0.9517 | 8.1783 | 22.4982 | 207.2949 | 237.9715 |
| WFI-XGB | rem\_4 | XGB | micro | 11 | 0.9514 | 0.9514 | 8.1783 | 7.2277 | 60.9729 | 76.3789 |
| SP | rem\_4 | RF | weighted | 11 | 0.9533 | 0.9512 | 0.5398 | 24.1702 | 251.2735 | 275.9835 |
| WFI-XGB | rem\_2 | RF | weighted | 13 | 0.9517 | 0.9501 | 8.1783 | 22.4982 | 208.6028 | 239.2794 |
| WFI-XGB | rem\_3 | RF | micro | 12 | 0.9499 | 0.9499 | 8.1783 | 22.4982 | 214.4513 | 245.1278 |
| WFI-RF | rem\_6 | RF | micro | 9 | 0.9493 | 0.9493 | 28.1665 | 23.6636 | 237.3647 | 289.1947 |
| WFI-XGB | rem\_4 | XGB | weighted | 11 | 0.9514 | 0.9488 | 8.1783 | 7.2277 | 56.1641 | 71.5701 |
| SP | rem\_5 | RF | micro | 10 | 0.9487 | 0.9487 | 0.5398 | 24.1702 | 269.1365 | 293.8465 |
| WFI-XGB | rem\_3 | RF | weighted | 12 | 0.9497 | 0.9476 | 8.1783 | 22.4982 | 217.598 | 248.2746 |
| WFI-RF | rem\_6 | RF | weighted | 9 | 0.9484 | 0.9464 | 28.1665 | 23.6636 | 234.7604 | 286.5905 |
| SP | rem\_5 | RF | weighted | 10 | 0.9487 | 0.9463 | 0.5398 | 24.1702 | 271.0955 | 295.8055 |
| WFI-RF | rem\_7 | RF | micro | 8 | 0.9441 | 0.9441 | 28.1665 | 23.6636 | 191.7316 | 243.5617 |
| WFI-RF | rem\_7 | RF | weighted | 8 | 0.9441 | 0.9421 | 28.1665 | 23.6636 | 195.1189 | 246.949 |
| SP | rem\_7 | XGB | micro | 8 | 0.9398 | 0.9398 | 0.5398 | 9.7235 | 48.4909 | 58.7542 |
| SP | rem\_6 | RF | micro | 9 | 0.9383 | 0.9383 | 0.5398 | 24.1702 | 254.6673 | 279.3773 |
| WFI-XGB | rem\_4 | RF | micro | 11 | 0.9379 | 0.9379 | 8.1783 | 22.4982 | 227.991 | 258.6676 |
| SP | rem\_6 | RF | weighted | 9 | 0.9383 | 0.9351 | 0.5398 | 24.1702 | 253.089 | 277.799 |
| WFI-XGB | rem\_4 | RF | weighted | 11 | 0.9379 | 0.935 | 8.1783 | 22.4982 | 194.7546 | 225.4312 |
| SP | rem\_7 | XGB | weighted | 8 | 0.9398 | 0.9349 | 0.5398 | 9.7235 | 56.7268 | 66.9901 |
| WFI-RF | rem\_2 | XGB | micro | 13 | 0.9343 | 0.9343 | 28.1665 | 7.3364 | 53.9466 | 89.4495 |
| WFI-RF | rem\_3 | XGB | micro | 12 | 0.9327 | 0.9327 | 28.1665 | 7.3364 | 53.5156 | 89.0185 |
| WFI-RF | rem\_4 | XGB | micro | 11 | 0.9321 | 0.9321 | 28.1665 | 7.3364 | 55.06 | 90.5629 |
| WFI-RF | rem\_8 | RF | micro | 7 | 0.9321 | 0.9321 | 28.1665 | 23.6636 | 202.0066 | 253.8367 |
| WFI-XGB | rem\_5 | XGB | micro | 10 | 0.9304 | 0.9304 | 8.1783 | 7.2277 | 50.9003 | 66.3063 |
| WFI-RF | rem\_8 | RF | weighted | 7 | 0.9321 | 0.9288 | 28.1665 | 23.6636 | 203.1551 | 254.9852 |
| WFI-RF | rem\_5 | XGB | micro | 10 | 0.9266 | 0.9266 | 28.1665 | 7.3364 | 51.0363 | 86.5392 |
| WFI-RF | rem\_2 | XGB | weighted | 13 | 0.9343 | 0.9257 | 28.1665 | 7.3364 | 58.2988 | 93.8017 |
| WFI-XGB | rem\_5 | XGB | weighted | 10 | 0.9304 | 0.9248 | 8.1783 | 7.2277 | 50.7896 | 66.1956 |
| WFI-RF | rem\_3 | XGB | weighted | 12 | 0.9327 | 0.924 | 28.1665 | 7.3364 | 53.5023 | 89.0052 |
| SP | rem\_7 | RF | micro | 8 | 0.9233 | 0.9233 | 0.5398 | 24.1702 | 211.847 | 236.557 |
| WFI-RF | rem\_4 | XGB | weighted | 11 | 0.9321 | 0.9232 | 28.1665 | 7.3364 | 51.8556 | 87.3585 |
| WFI-RF | rem\_6 | XGB | micro | 9 | 0.9217 | 0.9217 | 28.1665 | 7.3364 | 52.2315 | 87.7344 |
| WFI-RF | rem\_9 | RF | micro | 6 | 0.9198 | 0.9198 | 28.1665 | 23.6636 | 191.834 | 243.664 |
| WFI-XGB | rem\_6 | XGB | micro | 9 | 0.9191 | 0.9191 | 8.1783 | 7.2277 | 47.1635 | 62.5695 |
| WFI-RF | rem\_10 | RF | micro | 5 | 0.9182 | 0.9182 | 28.1665 | 23.6636 | 208.8816 | 260.7117 |
| SP | rem\_7 | RF | weighted | 8 | 0.9233 | 0.9175 | 0.5398 | 24.1702 | 210.9994 | 235.7094 |
| WFI-RF | rem\_5 | XGB | weighted | 10 | 0.9266 | 0.9168 | 28.1665 | 7.3364 | 56.7804 | 92.2833 |
| WFI-RF | rem\_9 | RF | weighted | 6 | 0.9198 | 0.915 | 28.1665 | 23.6636 | 192.6771 | 244.5072 |
| WFI-RF | rem\_10 | RF | weighted | 5 | 0.9182 | 0.9129 | 28.1665 | 23.6636 | 208.8889 | 260.719 |
| WFI-RF | rem\_7 | XGB | micro | 8 | 0.9103 | 0.9103 | 28.1665 | 7.3364 | 56.7201 | 92.223 |
| WFI-RF | rem\_6 | XGB | weighted | 9 | 0.9217 | 0.9101 | 28.1665 | 7.3364 | 50.5891 | 86.092 |
| WFI-XGB | rem\_6 | XGB | weighted | 9 | 0.9191 | 0.91 | 8.1783 | 7.2277 | 47.517 | 62.923 |
| WFI-RF | rem\_11 | RF | micro | 4 | 0.9039 | 0.9039 | 28.1665 | 23.6636 | 216.8949 | 268.725 |
| WFI-XGB | rem\_5 | RF | micro | 10 | 0.9039 | 0.9039 | 8.1783 | 22.4982 | 176.3528 | 207.0294 |
| WFI-RF | rem\_7 | XGB | weighted | 8 | 0.9103 | 0.8978 | 28.1665 | 7.3364 | 51.3897 | 86.8926 |
| SP | rem\_8 | RF | micro | 7 | 0.8976 | 0.8976 | 0.5398 | 24.1702 | 221.9051 | 246.6151 |
| WFI-XGB | rem\_5 | RF | weighted | 10 | 0.9039 | 0.8976 | 8.1783 | 22.4982 | 172.7294 | 203.406 |
| WFI-RF | rem\_11 | RF | weighted | 4 | 0.9039 | 0.8964 | 28.1665 | 23.6636 | 215.8312 | 267.6613 |
| WFI-RF | rem\_8 | XGB | micro | 7 | 0.8957 | 0.8957 | 28.1665 | 7.3364 | 48.6596 | 84.1625 |
| WFI-XGB | rem\_6 | RF | micro | 9 | 0.8916 | 0.8916 | 8.1783 | 22.4982 | 175.3519 | 206.0284 |
| WFI-XGB | rem\_7 | XGB | micro | 8 | 0.8914 | 0.8914 | 8.1783 | 7.2277 | 46.843 | 62.249 |
| SP | rem\_8 | RF | weighted | 7 | 0.8976 | 0.8879 | 0.5398 | 24.1702 | 222.8318 | 247.5418 |
| SP | rem\_8 | XGB | micro | 7 | 0.8871 | 0.8871 | 0.5398 | 9.7235 | 55.0001 | 65.2635 |
| WFI-RF | rem\_12 | RF | micro | 3 | 0.885 | 0.885 | 28.1665 | 23.6636 | 184.7289 | 236.559 |
| SP | rem\_9 | RF | micro | 6 | 0.8844 | 0.8844 | 0.5398 | 24.1702 | 218.8254 | 243.5354 |
| WFI-RF | rem\_10 | XGB | micro | 5 | 0.8827 | 0.8827 | 28.1665 | 7.3364 | 53.3513 | 88.8543 |
| WFI-RF | rem\_9 | XGB | micro | 6 | 0.8815 | 0.8815 | 28.1665 | 7.3364 | 45.9867 | 81.4896 |
| WFI-XGB | rem\_6 | RF | weighted | 9 | 0.8916 | 0.8815 | 8.1783 | 22.4982 | 172.8454 | 203.522 |
| SP | rem\_10 | RF | micro | 5 | 0.8808 | 0.8808 | 0.5398 | 24.1702 | 234.9464 | 259.6565 |
| WFI-RF | rem\_8 | XGB | weighted | 7 | 0.8957 | 0.8801 | 28.1665 | 7.3364 | 47.4788 | 82.9818 |
| SP | rem\_9 | XGB | micro | 6 | 0.8776 | 0.8776 | 0.5398 | 9.7235 | 50.3023 | 60.5656 |
| WFI-RF | rem\_12 | RF | weighted | 3 | 0.8857 | 0.876 | 28.1665 | 23.6636 | 182.7389 | 234.569 |
| SP | rem\_10 | XGB | micro | 5 | 0.8747 | 0.8747 | 0.5398 | 9.7235 | 52.8612 | 63.1245 |
| SP | rem\_8 | XGB | weighted | 7 | 0.8871 | 0.8746 | 0.5398 | 9.7235 | 52.9088 | 63.1721 |
| WFI-XGB | rem\_8 | XGB | micro | 7 | 0.8734 | 0.8734 | 8.1783 | 7.2277 | 49.6168 | 65.0228 |
| WFI-XGB | rem\_7 | XGB | weighted | 8 | 0.8914 | 0.8729 | 8.1783 | 7.2277 | 46.1808 | 61.5868 |
| SP | rem\_9 | RF | weighted | 6 | 0.8844 | 0.8711 | 0.5398 | 24.1702 | 219.9098 | 244.6198 |
| SP | rem\_10 | RF | weighted | 5 | 0.8808 | 0.8684 | 0.5398 | 24.1702 | 236.6167 | 261.3267 |
| WFI-RF | rem\_11 | XGB | micro | 4 | 0.8656 | 0.8656 | 28.1665 | 7.3364 | 46.457 | 81.9599 |
| SP | rem\_9 | XGB | weighted | 6 | 0.8776 | 0.8625 | 0.5398 | 9.7235 | 52.8954 | 63.1587 |
| WFI-RF | rem\_10 | XGB | weighted | 5 | 0.8827 | 0.8611 | 28.1665 | 7.3364 | 50.0003 | 85.5032 |
| WFI-RF | rem\_9 | XGB | weighted | 6 | 0.8815 | 0.8588 | 28.1665 | 7.3364 | 45.1721 | 80.675 |
| WFI-XGB | rem\_7 | RF | micro | 8 | 0.8574 | 0.8574 | 8.1783 | 22.4982 | 126.4491 | 157.1257 |
| SP | rem\_10 | XGB | weighted | 5 | 0.8729 | 0.8554 | 0.5398 | 9.7235 | 50.432 | 60.6953 |
| SP | rem\_11 | XGB | micro | 4 | 0.852 | 0.852 | 0.5398 | 9.7235 | 47.499 | 57.7623 |
| WFI-RF | rem\_13 | RF | micro | 2 | 0.8481 | 0.8481 | 28.1665 | 23.6636 | 200.846 | 252.6761 |
| SP | rem\_11 | RF | micro | 4 | 0.848 | 0.848 | 0.5398 | 24.1702 | 222.5105 | 247.2205 |
| WFI-XGB | rem\_8 | XGB | weighted | 7 | 0.8734 | 0.844 | 8.1783 | 7.2277 | 46.4764 | 61.8824 |
| WFI-RF | rem\_12 | XGB | micro | 3 | 0.8421 | 0.8421 | 28.1665 | 7.3364 | 54.8822 | 90.3851 |
| WFI-RF | rem\_11 | XGB | weighted | 4 | 0.8656 | 0.836 | 28.1665 | 7.3364 | 46.2119 | 81.7148 |
| WFI-XGB | rem\_8 | RF | micro | 7 | 0.8329 | 0.8329 | 8.1783 | 22.4982 | 114.2919 | 144.9685 |
| WFI-RF | rem\_13 | RF | weighted | 2 | 0.8481 | 0.8324 | 28.1665 | 23.6636 | 199.6825 | 251.5126 |
| WFI-XGB | rem\_9 | RF | micro | 6 | 0.8312 | 0.8312 | 8.1783 | 22.4982 | 81.5407 | 112.2173 |
| WFI-XGB | rem\_7 | RF | weighted | 8 | 0.8574 | 0.8301 | 8.1783 | 22.4982 | 126.5991 | 157.2756 |
| WFI-XGB | rem\_9 | XGB | micro | 6 | 0.8291 | 0.8291 | 8.1783 | 7.2277 | 44.6675 | 60.0735 |
| SP | rem\_11 | RF | weighted | 4 | 0.848 | 0.8283 | 0.5398 | 24.1702 | 222.0488 | 246.7588 |
| WFI-XGB | rem\_10 | XGB | micro | 5 | 0.8264 | 0.8264 | 8.1783 | 7.2277 | 47.194 | 62.5999 |
| WFI-XGB | rem\_11 | XGB | micro | 4 | 0.8264 | 0.8264 | 8.1783 | 7.2277 | 41.4423 | 56.8483 |
| WFI-XGB | rem\_10 | RF | micro | 5 | 0.8264 | 0.8264 | 8.1783 | 22.4982 | 39.4699 | 70.1465 |
| WFI-XGB | rem\_11 | RF | micro | 4 | 0.8263 | 0.8263 | 8.1783 | 22.4982 | 34.2725 | 64.9491 |
| SP | rem\_11 | XGB | weighted | 4 | 0.852 | 0.8257 | 0.5398 | 9.7235 | 47.5882 | 57.8515 |
| WFI-RF | rem\_13 | XGB | micro | 2 | 0.82 | 0.82 | 28.1665 | 7.3364 | 45.7813 | 81.2842 |
| WFI-XGB | rem\_12 | XGB | micro | 3 | 0.8136 | 0.8136 | 8.1783 | 7.2277 | 49.8336 | 65.2396 |
| WFI-XGB | rem\_12 | RF | micro | 3 | 0.8136 | 0.8136 | 8.1783 | 22.4982 | 29.1751 | 59.8517 |
| WFI-XGB | rem\_13 | XGB | micro | 2 | 0.807 | 0.807 | 8.1783 | 7.2277 | 44.3367 | 59.7427 |
| WFI-XGB | rem\_13 | RF | micro | 2 | 0.807 | 0.807 | 8.1783 | 22.4982 | 26.9387 | 57.6153 |
| WFI-XGB | rem\_14 | XGB | micro | 1 | 0.8048 | 0.8048 | 8.1783 | 7.2277 | 37.8282 | 53.2342 |
| WFI-XGB | rem\_14 | RF | micro | 1 | 0.8048 | 0.8048 | 8.1783 | 22.4982 | 25.0521 | 55.7287 |
| WFI-RF | rem\_14 | RF | micro | 1 | 0.8012 | 0.8012 | 28.1665 | 23.6636 | 69.5501 | 121.3802 |
| WFI-RF | rem\_12 | XGB | weighted | 3 | 0.8421 | 0.799 | 28.1665 | 7.3364 | 51.2851 | 86.788 |
| WFI-RF | rem\_14 | XGB | micro | 1 | 0.7955 | 0.7955 | 28.1665 | 7.3364 | 47.3624 | 82.8653 |
| SP | rem\_12 | XGB | micro | 3 | 0.786 | 0.786 | 0.5398 | 9.7235 | 46.2971 | 56.5605 |
| SP | rem\_13 | XGB | micro | 2 | 0.7849 | 0.7849 | 0.5398 | 9.7235 | 48.1331 | 58.3964 |
| SP | rem\_14 | XGB | micro | 1 | 0.7849 | 0.7849 | 0.5398 | 9.7235 | 28.7305 | 38.9938 |
| SP | rem\_14 | RF | micro | 1 | 0.7849 | 0.7849 | 0.5398 | 24.1702 | 21.9576 | 46.6676 |
| SP | rem\_13 | RF | micro | 2 | 0.7847 | 0.7847 | 0.5398 | 24.1702 | 60.6919 | 85.4019 |
| SP | rem\_12 | RF | micro | 3 | 0.7833 | 0.7833 | 0.5398 | 24.1702 | 104.3669 | 129.0769 |
| WFI-XGB | rem\_8 | RF | weighted | 7 | 0.8329 | 0.7762 | 8.1783 | 22.4982 | 117.5742 | 148.2507 |
| WFI-XGB | rem\_9 | RF | weighted | 6 | 0.8312 | 0.7727 | 8.1783 | 22.4982 | 82.8946 | 113.5712 |
| WFI-XGB | rem\_9 | XGB | weighted | 6 | 0.8292 | 0.7661 | 8.1783 | 7.2277 | 48.9212 | 64.3272 |
| WFI-RF | rem\_13 | XGB | weighted | 2 | 0.82 | 0.7599 | 28.1665 | 7.3364 | 44.8663 | 80.3692 |
| WFI-XGB | rem\_10 | RF | weighted | 5 | 0.8264 | 0.7596 | 8.1783 | 22.4982 | 37.9052 | 68.5818 |
| WFI-XGB | rem\_11 | RF | weighted | 4 | 0.8263 | 0.7595 | 8.1783 | 22.4982 | 34.451 | 65.1276 |
| WFI-XGB | rem\_10 | XGB | weighted | 5 | 0.8265 | 0.7594 | 8.1783 | 7.2277 | 43.5392 | 58.9452 |
| WFI-XGB | rem\_11 | XGB | weighted | 4 | 0.8263 | 0.7591 | 8.1783 | 7.2277 | 42.7077 | 58.1136 |
| WFI-XGB | rem\_12 | XGB | weighted | 3 | 0.8136 | 0.7306 | 8.1783 | 7.2277 | 48.752 | 64.158 |
| WFI-XGB | rem\_12 | RF | weighted | 3 | 0.8136 | 0.7306 | 8.1783 | 22.4982 | 29.0807 | 59.7573 |
| WFI-XGB | rem\_13 | XGB | weighted | 2 | 0.807 | 0.7231 | 8.1783 | 7.2277 | 40.5419 | 55.9479 |
| WFI-XGB | rem\_13 | RF | weighted | 2 | 0.807 | 0.7231 | 8.1783 | 22.4982 | 27.4431 | 58.1197 |
| WFI-RF | rem\_14 | RF | weighted | 1 | 0.8012 | 0.7211 | 28.1665 | 23.6636 | 70.1062 | 121.9363 |
| WFI-XGB | rem\_14 | XGB | weighted | 1 | 0.8048 | 0.7199 | 8.1783 | 7.2277 | 37.3 | 52.706 |
| WFI-XGB | rem\_14 | RF | weighted | 1 | 0.8048 | 0.7199 | 8.1783 | 22.4982 | 24.1353 | 54.8119 |
| WFI-RF | rem\_14 | XGB | weighted | 1 | 0.7942 | 0.7153 | 28.1665 | 7.3364 | 46.8033 | 82.3062 |
| SP | rem\_12 | XGB | weighted | 3 | 0.786 | 0.7107 | 0.5398 | 9.7235 | 46.6887 | 56.952 |
| SP | rem\_12 | RF | weighted | 3 | 0.7833 | 0.6985 | 0.5398 | 24.1702 | 101.9762 | 126.6863 |
| SP | rem\_13 | XGB | weighted | 2 | 0.7849 | 0.692 | 0.5398 | 9.7235 | 46.4174 | 56.6807 |
| SP | rem\_14 | XGB | weighted | 1 | 0.7849 | 0.692 | 0.5398 | 9.7235 | 27.8071 | 38.0704 |
| SP | rem\_13 | RF | weighted | 2 | 0.7847 | 0.692 | 0.5398 | 24.1702 | 63.7639 | 88.474 |
| SP | rem\_14 | RF | weighted | 1 | 0.7849 | 0.692 | 0.5398 | 24.1702 | 21.7626 | 46.4726 |

**Formatting Guideline for Sanstream Technical Reports**

**(Delete this section before submission)**

Follow guidelines as described in Table 1 consistently throughout the report.

Table 14: Formatting Guideline

|  |  |  |
| --- | --- | --- |
| **MS Word Feature** | **Where to set** | **Expected Value** |
| Style |  | **Header 1-4** for headings as appropriate.  **Normal** for the body |
| Font |  | **Arial 12 pt** for the body of the report  **Arial 11 pt and Italics** for Figure and Table captions  Example:  *Figure 1: Caption of figure 1* |
| Alignment |  | **Justify** for the body of the report  **Left** for text in Table  **Centre** for figure and table captions |
| Spacing |  | **1.5** for the body of the report |
| Captions |  | All figures and tables must have captions.  **Table** captions should be on **top** of the table  **Figure** captions should be at the **bottom** of the figure |

**Content Guidelines**

|  |
| --- |
| Header Types:  See the superscript on the Section Headers  a - Mandatory for all reports  b - Mandatory for survey reports  c - Mandatory for documenting a new methodology/approach proposed by you  d - Mandatory if describing a design of a software system, testbed, framework, etc.  e - Mandatory for reports on experimental work  *1. You are encouraged to use more sections (Heading 1) than listed here and subsections (Heading 2,3,4) as deemed appropriate for your report.*  *2. You can also slightly modify the title of the Section especially for the Design section to be more specific*  *3. Remove the superscripts from the headers and the “Formatting Guideline for Cistel Technical Report” section once the report is completed and ready for review.*  *4. Perform an Update Field on the Table of Contents, List of Figures and List of Tables.* |

Figure 10 Content Guidelines