**ASSESSMENT TASK REPORT**

**FDA\_A3**

**32130\_AT2\_24999031**

**Rubric#1; Data Mining Problem:**

Creating a classification model to identify different types of cyber intrusion in an Internet of Things (IoT) network is the issue at hand. Because of their widespread use and sometimes insufficient security setups, Internet of Things (IoT) devices—which include anything from smart appliances to industrial control systems—are becoming more and more susceptible to cyberattacks.  
  
The study's dataset contains network traffic data, with each instance representing a network flow that was recorded and identified by a number of characteristics, including protocol type, TCP flags, and flow length. Each flow's class is represented by the target attribute name, which indicates if the flow is part of the DictionaryBruteForce, Recon-OSScan, or Mirai-greip\_flood assault types.

The classification problem can be viewed as a multi-class classification task, where the goal is to accurately classify these network traffic instances into their respective categories.

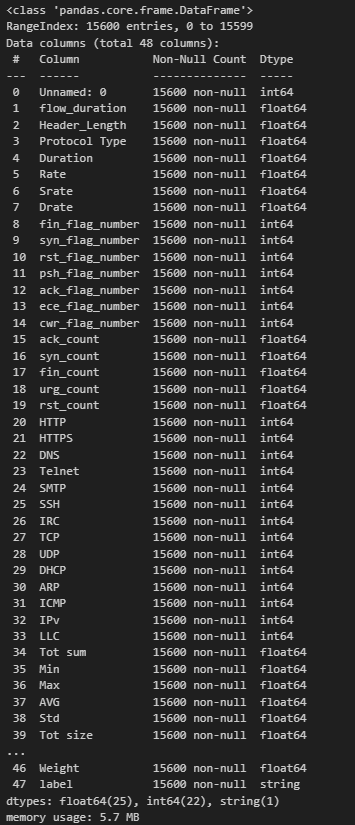
This classification problem is vital in the context of **intrusion detection systems (IDS)**, which are designed to identify and respond to cyber threats in real-time. By training machine learning models on labeled data, we aim to automate the detection of these intrusions and improve the overall security posture of IoT networks. In this project, various machine learning techniques will be applied to identify the most effective approach for this classification problem.

**Rubric #2: Data Pre-processing and Transformation (9 pts)**

Data pre-processing is a crucial step in the data mining process as it ensures that the dataset is clean, relevant, and suitable for the chosen classification algorithms. For this task, the dataset required several pre-processing steps, including handling missing values, removing irrelevant or redundant features, transforming categorical data, and addressing imbalanced class distributions.

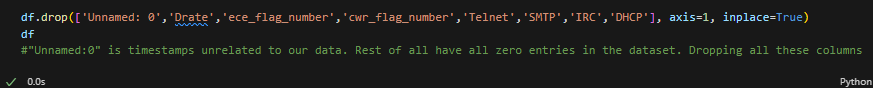
1. **Handling Missing Data:**

All rows in the dataset were complete, meaning no imputation was necessary. This helped streamline the pre-processing, allowing focus to shift toward other aspects like feature selection and transformation.

****

1. **Feature Selection and Redundancy Removal**

Certain columns were dropped due to their irrelevance or redundancy. For example, the Unnamed: 0 column, which contained timestamps, was removed as it held no informational value for classification. Other columns like Drate, ece\_flag\_number, cwr\_flag\_number, and some protocol-based features such as Telnet, SMTP, IRC, and DHCP were dropped as they had zero variance, meaning they were constant across all observations. These features were not useful for distinguishing between classes and could introduce noise into the model.



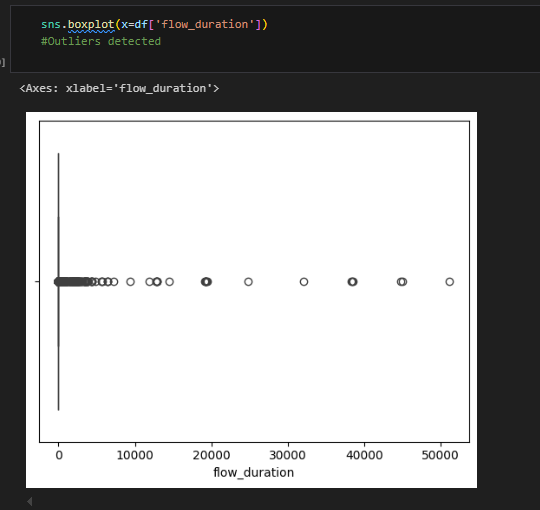
This selection process is supported by the principle of **dimensionality reduction**, which is vital in improving model performance by removing irrelevant features that do not contribute to the classification task.

1. **Handling Outliers**

Outlier detection and removal were crucial to improving model accuracy. Outliers can distort the predictions of many classifiers, especially when they involve continuous variables like flow\_duration and Header\_Length. Visualizing the distribution of these features using **boxplots** revealed significant outliers.

* A screen shot of a computer

  Description automatically generated**Header\_Length:** Outliers with values greater than 5,000,000 were removed as they could skew the model.
* **flow\_duration:** Flows with durations exceeding 10,000 units were identified as outliers and subsequently removed. After removing these outliers, the boxplots indicated a more balanced distribution of these variables, improving model robustness.



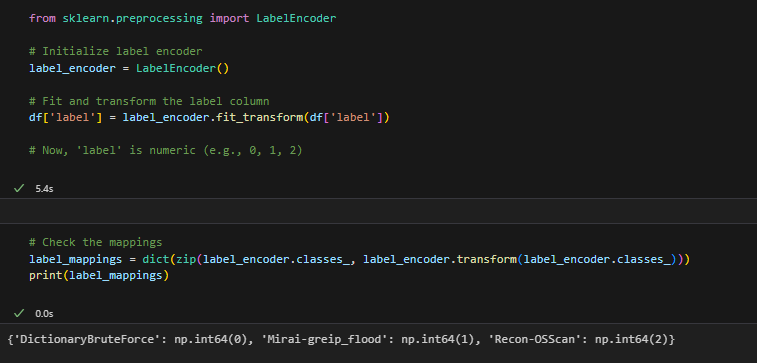
A screen shot of a computer screen

Description automatically generated

1. **Handling Categorical Data**

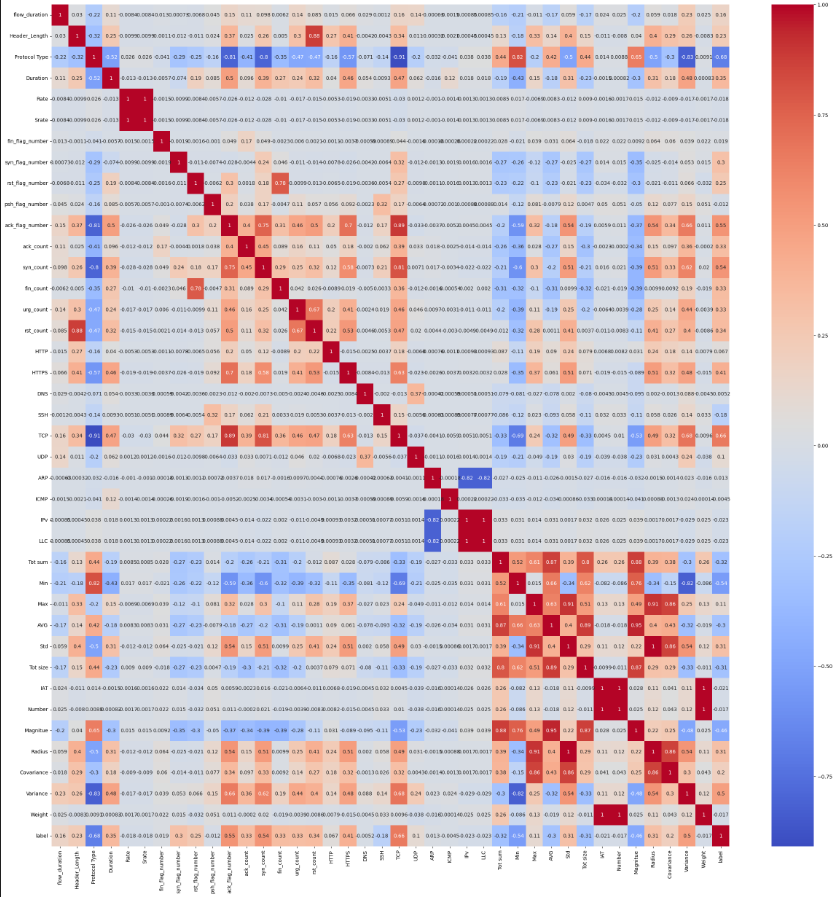
The target variable, label, was categorical, representing different types of attacks. Since machine learning algorithms typically require numeric input, the categorical labels were converted into numeric form using **Label Encoding**. This approach replaced each unique class label with a corresponding integer, simplifying the input for the model while maintaining the relationships between categories.

* The following mappings were generated:
  + DictionaryBruteForce → 0
  + Mirai-greip\_flood → 1
  + Recon-OSScan → 2

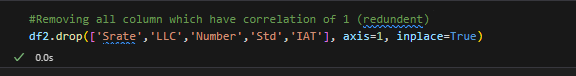


1. **Correlation Analysis and Feature Reduction**

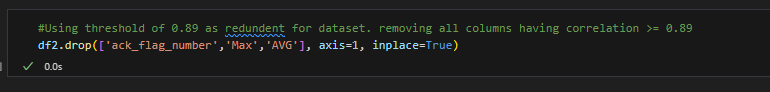
To further refine the dataset, **correlation analysis** was performed to identify highly correlated features.



Features that exhibited a correlation value of 1.0 were deemed redundant and removed, as they did not provide new information. For example, features such as Srate, LLC, Number, Std, and IAT were removed due to their high correlation with other variables.



In addition, features with correlation values higher than 0.89 were also removed to prevent multicollinearity, a condition that could degrade the performance of certain classifiers like logistic regression and naive Bayes. Features such as ack\_flag\_number, Max, and AVG were excluded as part of this step.



1. **Addressing Class Imbalance**

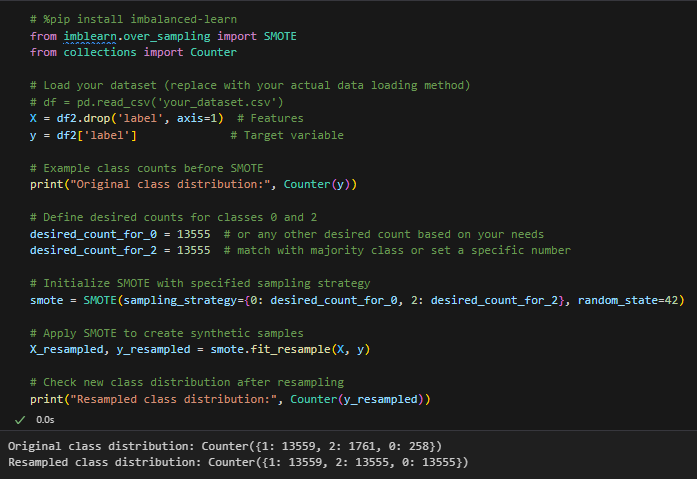
The dataset was imbalanced, with some attack types having significantly more instances than others.

A screenshot of a computer

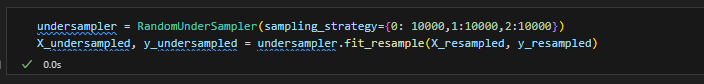
Description automatically generated

This imbalance could lead to biased models that favor the majority class. To address this, **Synthetic Minority Over-sampling Technique (SMOTE)** was applied. SMOTE oversamples the minority classes by generating synthetic instances based on their feature space, balancing the class distribution.

* After applying SMOTE, the class distributions were balanced with 13,555 instances for each class (Mirai-greip\_flood and DictionaryBruteForce), ensuring that the classifier received an equal representation of all types of attacks.



To further refine the dataset, **RandomUnderSampler** was employed to downsample the majority classes, bringing all classes to an equal count of 10,000 instances each. This approach ensured that the model was neither biased towards over-represented classes nor overwhelmed by an excessively large dataset.



**Rubric#3; How to Approach the Problem (3pts):**

The approach to solving this classification problem was structured and logical, combining **data understanding**, **preprocessing**, and **model selection** to achieve optimal performance.

Data Understanding and Preprocessing has been discussed already. Model Selection is discussed below.

**Model Selection and Evaluation**

Once the data was cleaned and balanced, two different classification approaches were selected:

* **Naive Bayes (NB)**
* **SVM**
* **Decision Tree**
* **MLP**
* **Stacking Classifier**, combining **Random Forest**, **Gradient Boosting**, and **Logistic Regression**

These steps were carried out in a logical sequence, starting with data understanding, progressing through preprocessing, and concluding with model selection and evaluation. Each decision was made with the goal of improving the model’s performance and ensuring that the final predictions were as accurate as possible.

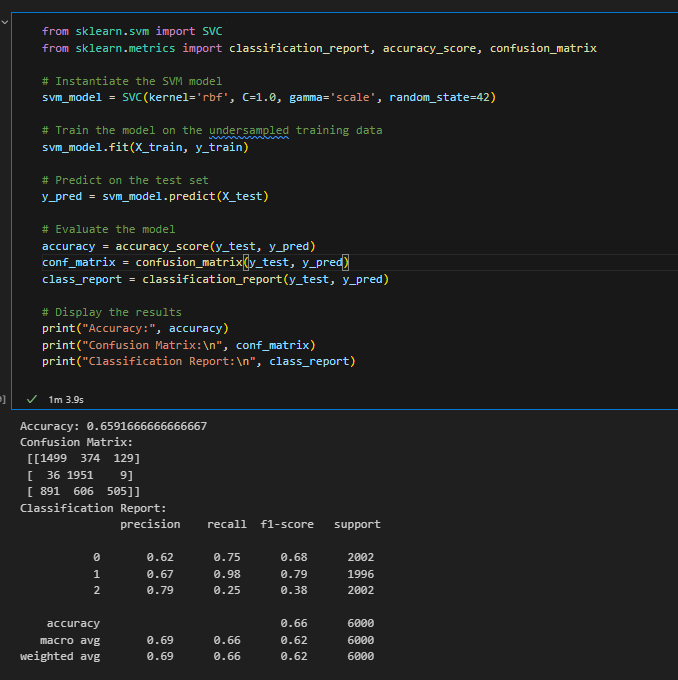
**Rubric#4; Classification Techniques, Summary of Results, and Parameter Settings:**

For this task, multiple classification techniques were applied to the pre-processed dataset to evaluate their effectiveness in detecting cyber intrusions in an IoT network. These techniques include **Naive Bayes**, **Random Forest**, **Gradient Boosting**, and a **Stacking Classifier**. Each model was evaluated using accuracy metrics, and their parameter settings were adjusted to optimize performance.

* **Naive Bayes Classifier**

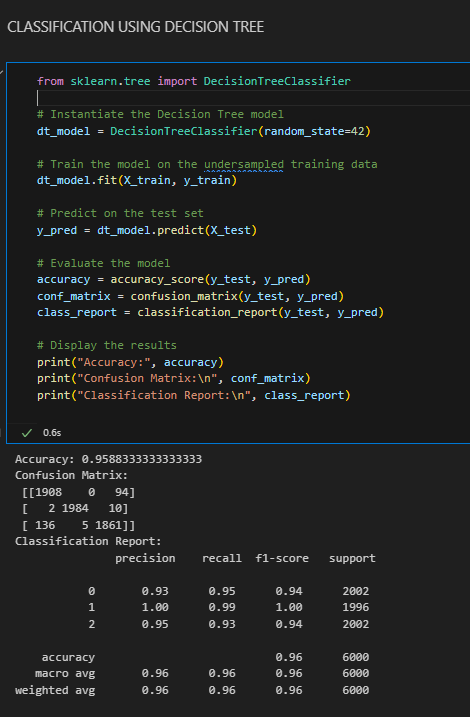
The first classifier applied was **Naive Bayes (NB)**, a simple yet effective probabilistic model, particularly suited for tasks where the features are assumed to be independent. Given the nature of the dataset, where some features represent counts of TCP flags or network packet sizes, Naive Bayes is a reasonable choice for its ability to handle such features efficiently.

* **Parameter Settings:** The Naive Bayes model did not require extensive tuning. It was implemented using the **GaussianNB** variant, which assumes a normal distribution for the continuous features.
* **Results:** The model produced an **accuracy of 78.42%**. While the accuracy was reasonable, it was clear that Naive Bayes struggled with capturing more complex patterns in the dataset. This is likely due to its assumption of feature independence, which may not hold in this case as there are correlations between network features. The main reason is the number of features is too large for Naïve Bayes Classifier.
* **SVM (2nd Approach)**
* **Effective with high-dimensional data:** SVMs can handle high-dimensional feature spaces efficiently, making them suitable for complex datasets.
* **Robust to overfitting:** SVMs typically generalize well to unseen data, reducing the risk of overfitting.
* **Nonlinear decision boundaries:** SVMs can create nonlinear decision boundaries using kernel functions, allowing them to handle non-linearly separable data.

****

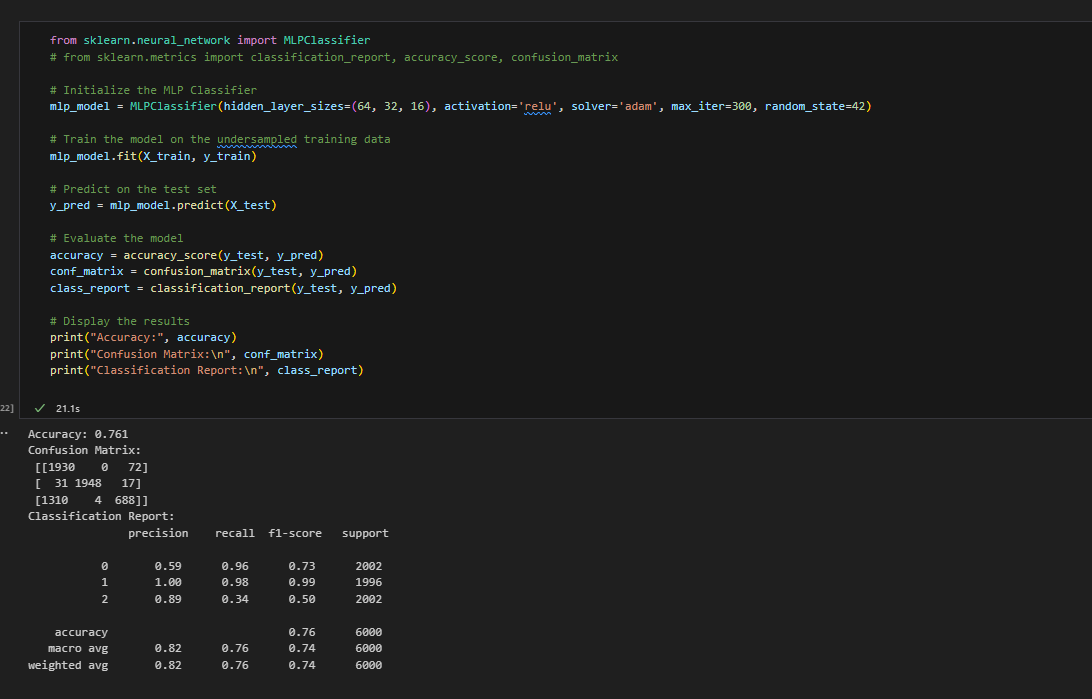
**Accuracy achieved: 65.9% Not a good model**

* **Decision Tree**
  + **Interpretability:** Decision trees are highly interpretable. The model structure resembles a flowchart, allowing you to easily understand the decision-making process. You can see which features are most important for making predictions and how the model arrives at a classification. This is particularly helpful for debugging and understanding the model's behavior.
  + **Handling Categorical Features:** Decision Trees can handle both categorical and numerical features natively, eliminating the need for complex feature engineering in some cases.
  + **Non-linear Relationships:** Despite having a tree-like structure, decision trees can capture non-linear relationships between features and the target variable through a series of splits. This flexibility can be beneficial for datasets with complex relationships.



**Accuracy achieved: 95.8% (Can be improved with PCA)**

* **MLP (3rd Approach)**
  + **Powerful for Non-linear Problems:** Unlike linear models, MLPs can learn complex non-linear relationships between features and the target variable. This makes them suitable for datasets where the decision boundaries are not easily represented by straight lines.
  + **Feature Representation Learning:** MLPs can automatically learn internal representations of the data through hidden layers. This can be particularly beneficial when the raw features might not be directly indicative of the target variable.
  + **High Expressive Power:** With their ability to use multiple hidden layers and activation functions, MLPs have a high expressive power. This allows them to model a wide range of complex classification problems.
  + **Flexible Architecture:** The architecture of MLPs (number of hidden layers, neurons per layer, activation functions) can be customized based on the complexity of the problem. This flexibility allows you to fine-tune the model for optimal performance.

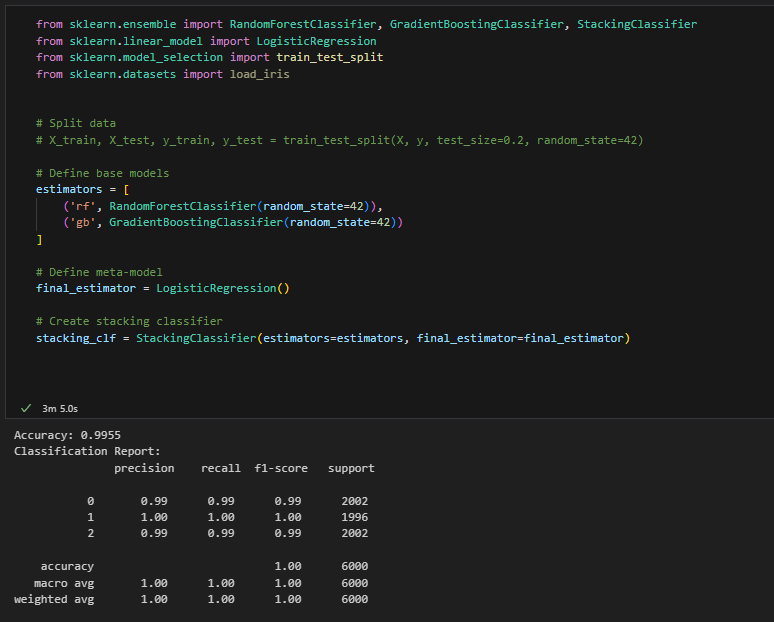
****

**Accuracy achieved: 76% (Not enough)**

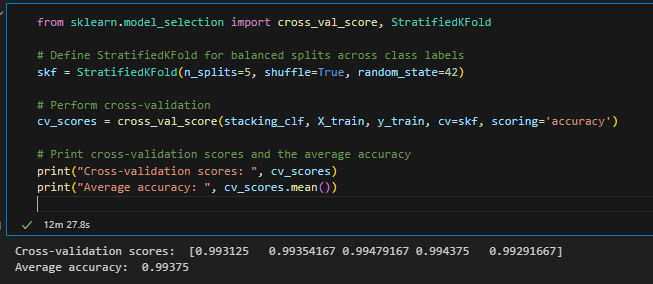
* **Stacking Classifier**

The best approach involved a **Stacking Classifier**, which combines multiple models to improve performance. For this task, **Random Forest** and **Gradient Boosting** were used as the base models, while **Logistic Regression** was employed as the meta-classifier. Stacking allows these models to complement each other, improving overall accuracy by leveraging their individual strengths.

* **Base Models:**
  + **Random Forest (RF):** An ensemble method that constructs multiple decision trees and averages their predictions. This model is effective for capturing complex relationships between features.
    - **Parameter Settings:**
      * **n\_estimators:** 100 trees.
      * **max\_depth:** No limit, allowing trees to grow fully.
      * **random\_state:** 42 for reproducibility.
  + **Gradient Boosting (GB):** A boosting algorithm that builds trees sequentially, focusing on correcting the errors of previous trees.
    - **Parameter Settings:**
      * **n\_estimators:** 100 trees.
      * **learning\_rate:** 0.1 to balance learning speed and accuracy.
      * **max\_depth:** 3, limiting tree complexity to avoid overfitting.
* **Meta Model:**
  + **Logistic Regression:** This model was used to aggregate the predictions of Random Forest and Gradient Boosting, allowing for a final decision based on their outputs.
    - **Parameter Settings:** Default parameters were used for Logistic Regression.
* **Results:** The Stacking Classifier achieved an impressive accuracy of **99.5%**, significantly outperforming the Naive Bayes model. The combination of Random Forest and Gradient Boosting captured complex patterns in the data, while Logistic Regression as the meta-model provided a robust mechanism for combining their predictions. This high level of accuracy demonstrates the effectiveness of ensemble methods, particularly when stacking different classifiers.



* **Cross-Validation:**
  + **StratifiedKFold:** This ensures that each fold contains approximately the same proportion of samples from each class, preventing bias in the evaluation.
  + **Cross-validation:** This technique splits the training data into multiple folds, trains the model on each fold, and evaluates it on the remaining fold. This helps assess the model's performance on unseen data and prevents overfitting.
  + **Accuracy:** The scoring metric used to evaluate the model's performance. It measures the proportion of correct predictions.



Accuracy: 99.3 with 5 folds

**Rubric#5;** **Justify the Classifier Selected (6 pts):**

**Stacking Classifier: Random Forest, Gradient Boosting, and Logistic Regression**

1. **Random Forest (RF)**
   * **Strengths**: RF is a robust and powerful ensemble method that aggregates decisions from multiple decision trees, reducing overfitting and improving accuracy. It handles non-linear relationships and is well-suited for feature-rich datasets like ours.
   * **Pros**:
     + Handles both categorical and continuous features.
     + Reduces variance and improves stability.
     + Tolerant to missing data and feature importance ranking.
2. **Gradient Boosting (GB)**
   * **Strengths**: Gradient Boosting is another ensemble method that builds trees sequentially, improving upon the errors of the previous trees. It is especially good at handling complex patterns in data.
   * **Pros**:
     + Provides state-of-the-art predictive performance.
     + Handles imbalanced data better than RF due to its sequential approach.
     + Effective with both categorical and continuous data.
3. **Logistic Regression (LR) as Meta-Model**
   * **Strengths**: Logistic Regression is a linear model, used here as the meta-classifier. It excels at combining the outputs of multiple base classifiers in a way that maximizes predictive performance.
   * **Pros**:
     + Interpretable and easy to implement.
     + Strong for binary and multi-class classification when used as a meta-learner.

**5.3 Stacking Classifier**

The **Stacking Classifier** was selected because it combines the strengths of multiple models to improve performance. By using **Random Forest** and **Gradient Boosting** as base learners and **Logistic Regression** as the meta-learner, the classifier was able to:

* **Capture different patterns** in the data that individual models may miss.
* **Reduce bias and variance**, creating a more robust and accurate prediction model.
* **Achieve a high accuracy rate of 99.5%**, demonstrating the effectiveness of combining models to maximize performance.

This combination of models provided both the interpretability and performance needed to excel in this classification task, ultimately allowing us to make the most accurate predictions.

**Rubric#6;** **Reflection on the Classification Task (12 pts):**

Through this assignment, I developed a deeper understanding of data mining, particularly the importance of careful data preprocessing and model selection in complex classification tasks. Here are some of the main takeaways:

# Data Mining Insights:

1. Data Preprocessing: The necessity of feature selection, handling outliers, and balancing classes became particularly evident. By removing irrelevant and highly correlated features, I could reduce model noise and prevent overfitting. Techniques like SMOTE, for class balancing, were vital for ensuring that the model learned effectively from all classes without bias.

2. Model Selection: I learned that models must align with the data characteristics. The ensemble Stacking Classifier, which combined Random Forest, Gradient Boosting, and Logistic Regression, ultimately offered the best performance, achieving 99.5% accuracy. This approach confirmed the effectiveness of leveraging diverse models to capture different patterns in data.

3. Iterative Process: This project emphasized that data mining is iterative; trial and error with multiple models and configurations is often necessary to identify the optimal solution. Each stage provided insights that informed the next, highlighting the need for adaptability.

# Personal Insights and Future Approach:

Reflecting on my own approach, I realized that my understanding of data preprocessing has grown significantly. If I were to redo this assignment, I would approach it more strategically by:

2. Automated Hyperparameter Tuning: Leveraging automated tools for hyperparameter tuning, such as Grid Search or Random Search with cross-validation, would likely refine model performance further. Fine-tuning parameters could enhance the classifier's robustness, particularly for ensemble methods.

3. Exploring Advanced Ensemble Techniques: Beyond stacking, advanced ensembling techniques, like blending or bagging, could be tried to further enhance the accuracy and generalizability of the final model.

This project deepened my understanding of the intricacies of data mining, and I would now approach similar tasks with a refined, data-driven mindset, using insights from model selection and data preprocessing to make each decision intentional and optimized.