VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY

INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

****

**Data Mining Project**

**Course:** IT160IU - Data Mining

**Semester:** Fall 2024

Group Members

1. Nguyễn Minh Đạt ITDSIU22166
2. Dương Ngọc Phương Anh ITDSIU22135
3. Nguyễn Hải Phú ITDSIU22179

**Instructor:** MSc. Nguyen Quang Phu

# **Introduction**

Data mining is the process of extracting valuable and actionable insights from extensive datasets, enabling organizations and researchers to uncover hidden patterns and trends. Machine learning enhances and automates this process by leveraging algorithms like Decision Trees, Naive Bayes, and IBK, which are tailored to solve a variety of classification tasks. These algorithms bring unique strengths to the table: Decision Trees provide transparent and interpretable decision rules, Naive Bayes delivers reliable probabilistic predictions based on simple yet effective assumptions, and IBK offers a flexible, instance-based learning approach. Together, they exemplify the potential of combining data mining with machine learning to tackle complex, real-world challenges. In the context of this assignment, their application underscores the transformative role these technologies play in identifying meaningful insights and making data-driven decisions more efficiently.

**Objective**

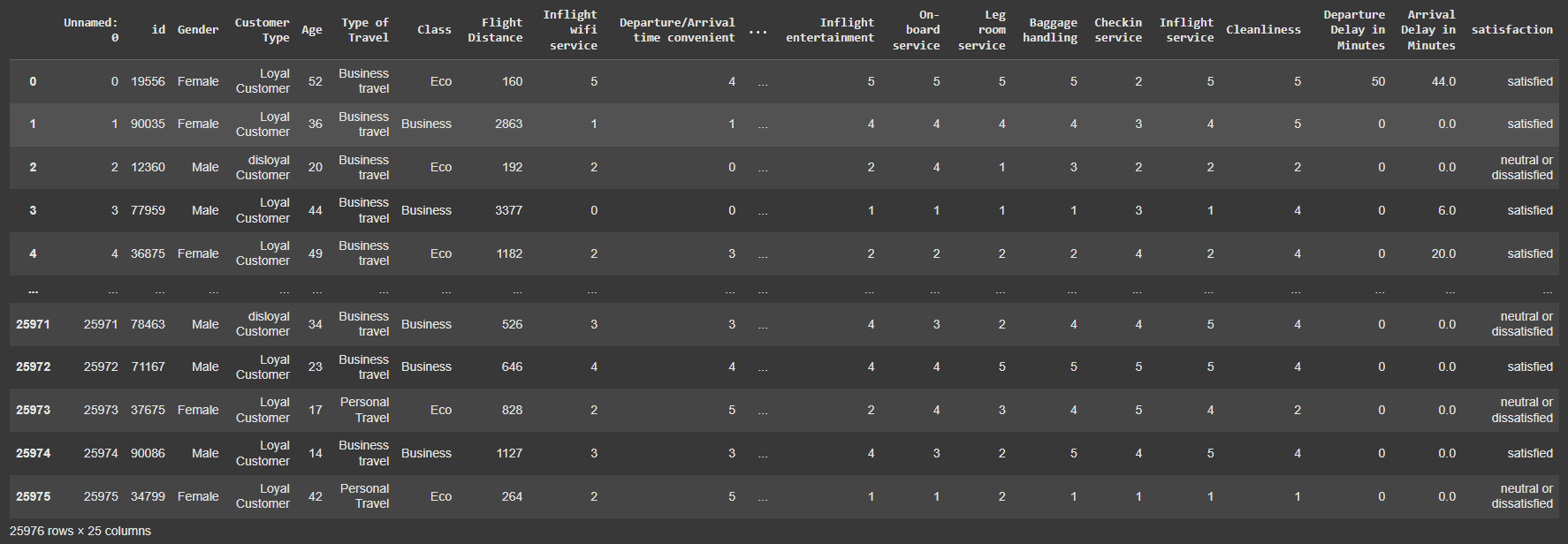
The purpose of this project is to develop an advanced data mining framework composed of two key components, each designed to address specific analytical objectives and provide valuable insights:

* The first component is a classification and prediction model that integrates the strengths of Decision Trees, Naive Bayes, and IBK algorithms. This model will leverage key features such as demographic attributes, ticket class, and other relevant variables to accurately predict passenger survival. By combining these algorithms, the framework ensures a balanced approach, utilizing Decision Trees for their interpretability, Naive Bayes for its probabilistic predictions, and IBK for its adaptability to instance-based learning. Together, these methods aim to deliver robust and reliable predictions.
* The second component is a sequence mining algorithm, which seeks to uncover hidden sequential patterns within the dataset. This algorithm will identify logical relationships and temporal dependencies between events, providing a deeper understanding of how different factors interact over time to influence survival outcomes. By analyzing these patterns, the framework will reveal meaningful insights that go beyond static feature relationships.

The primary goal of this project is to evaluate the effectiveness, accuracy, and relevance of these machine learning models in addressing classification problems, particularly in predicting survival outcomes. Furthermore, this project highlights the adaptability of these techniques to varying types of data and tasks. The integration of both classification and sequence mining approaches not only demonstrates the versatility of data mining but also emphasizes the importance of combining multiple methodologies to achieve comprehensive and insightful results. Ultimately, the framework aims to showcase the practical application of machine learning in solving real-world challenges while providing a foundation for future research and development in data-driven decision-making.

**Dataset Used**

Specify the dataset selected from the provided options (e.g., Production Quality Dataset from Kaggle).

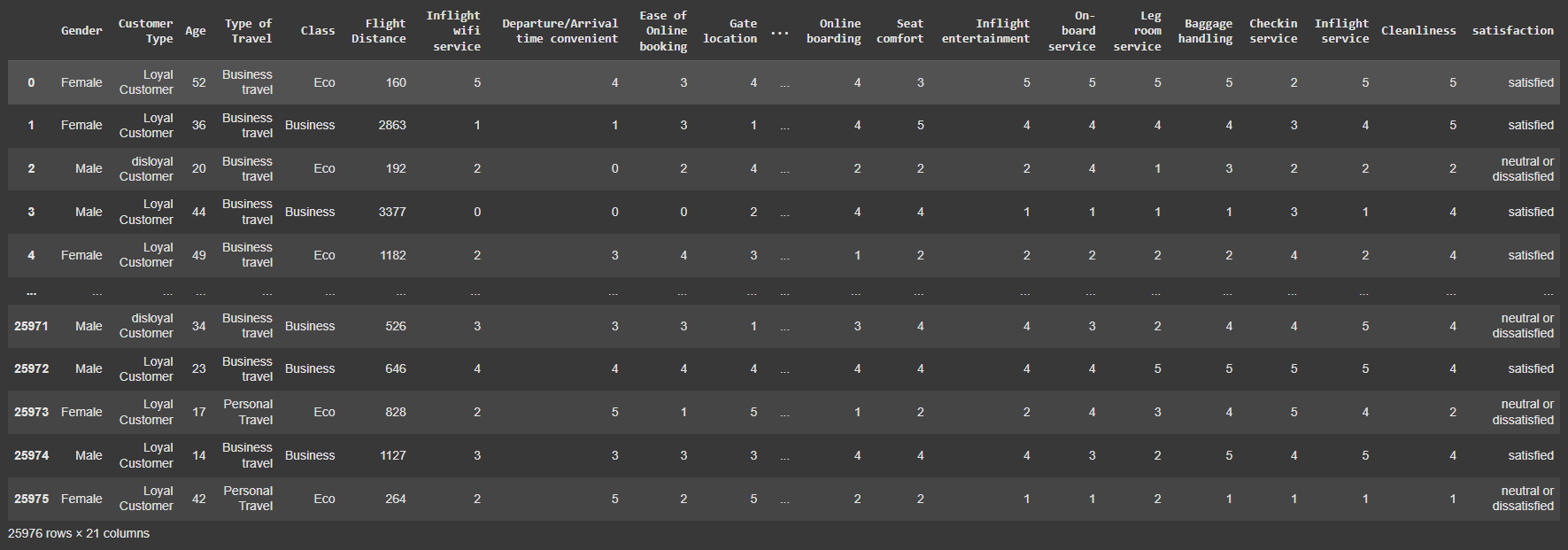
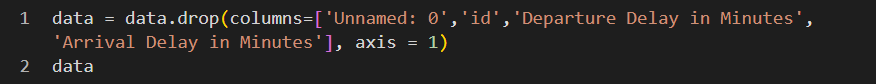


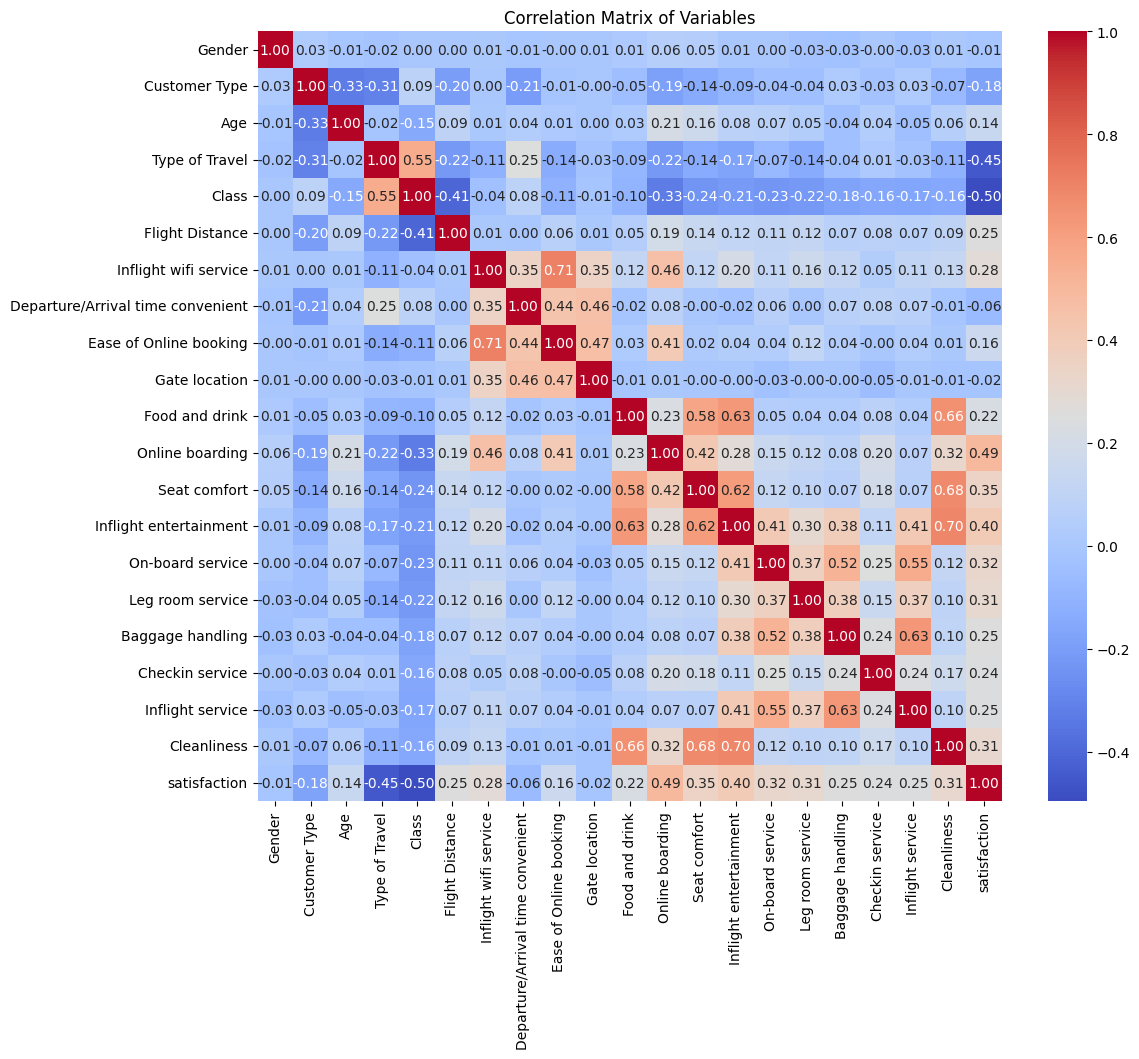
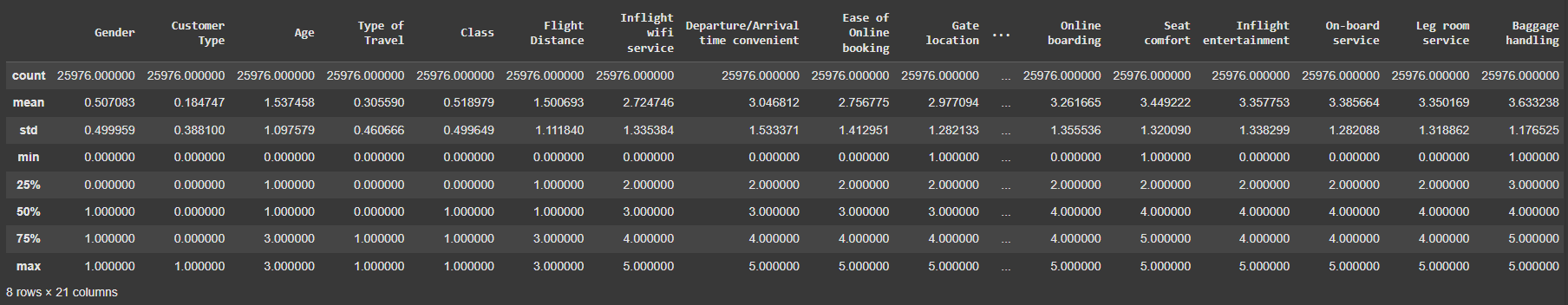
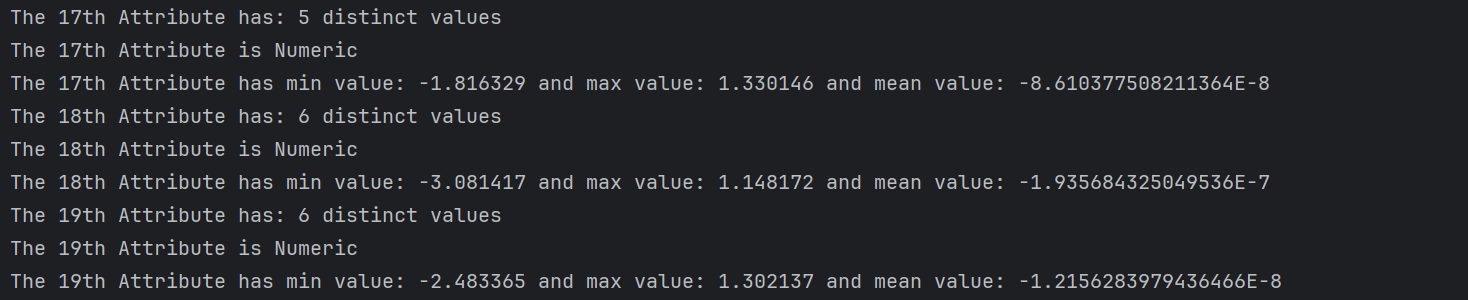
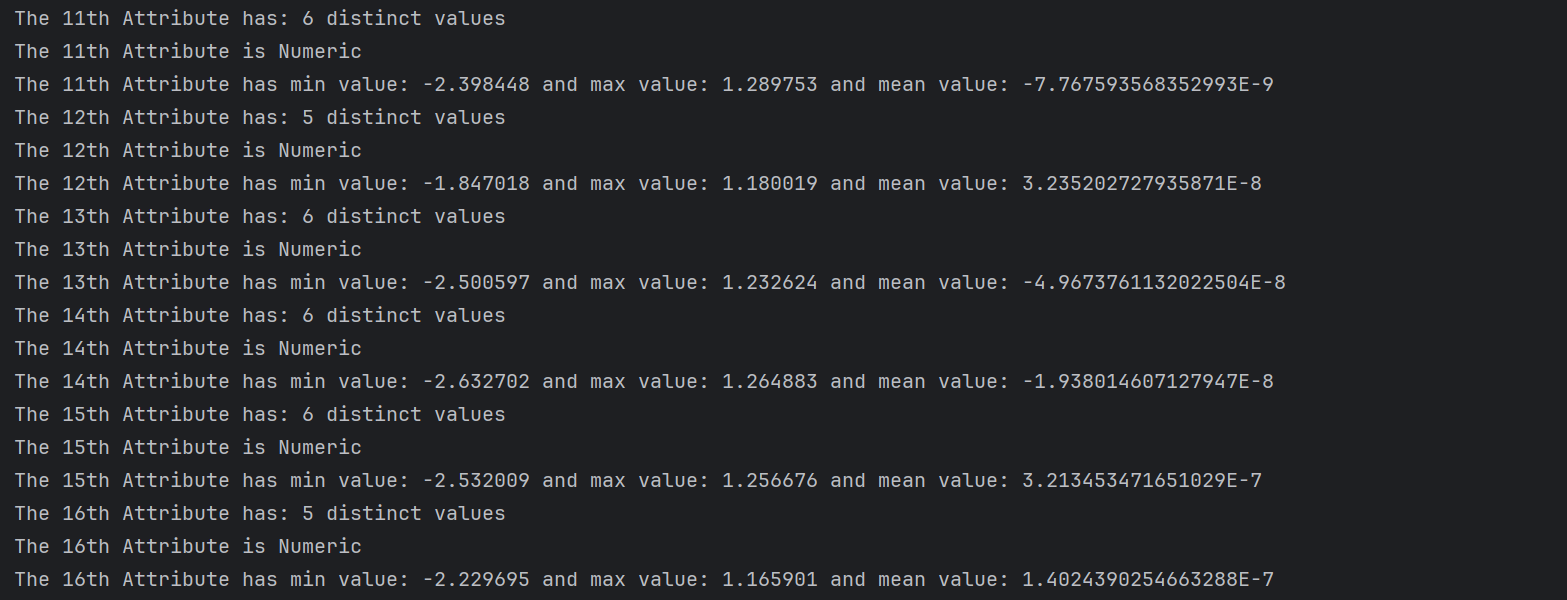
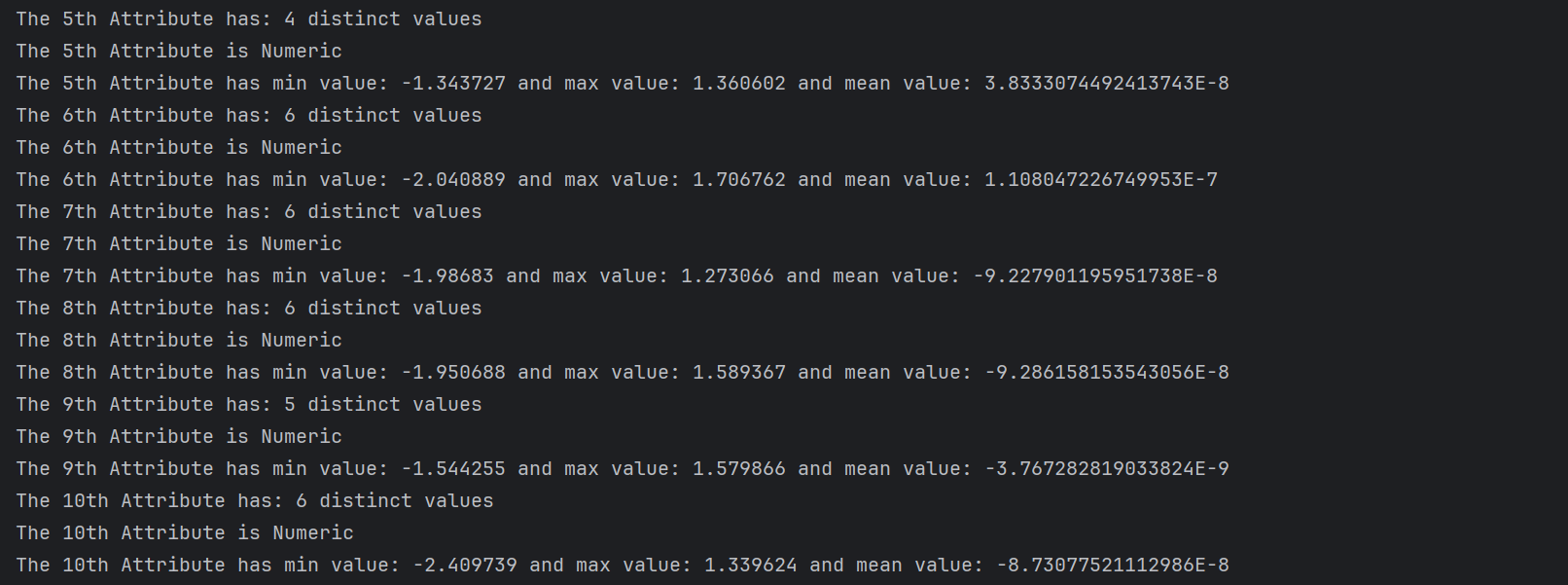
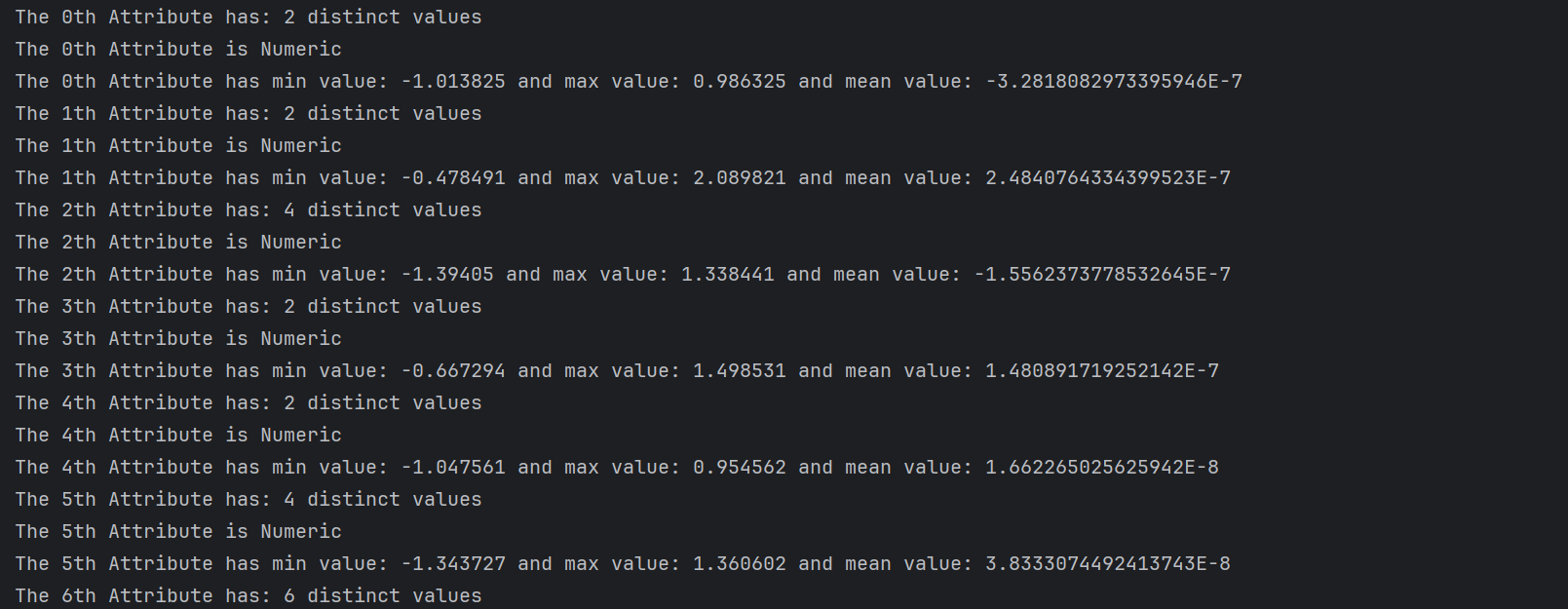
# **DATA PRE-PROCESSING**

## **Objective**

* Preparing raw data for analysis by cleaning, transforming, and organizing it into a format suitable for modeling and decision-making.
* Ensuring the data is accurate, consistent, and free from errors or missing values.
* Enabling effective analysis and improving the performance of machine learning models.

## **Raw Data Overview**

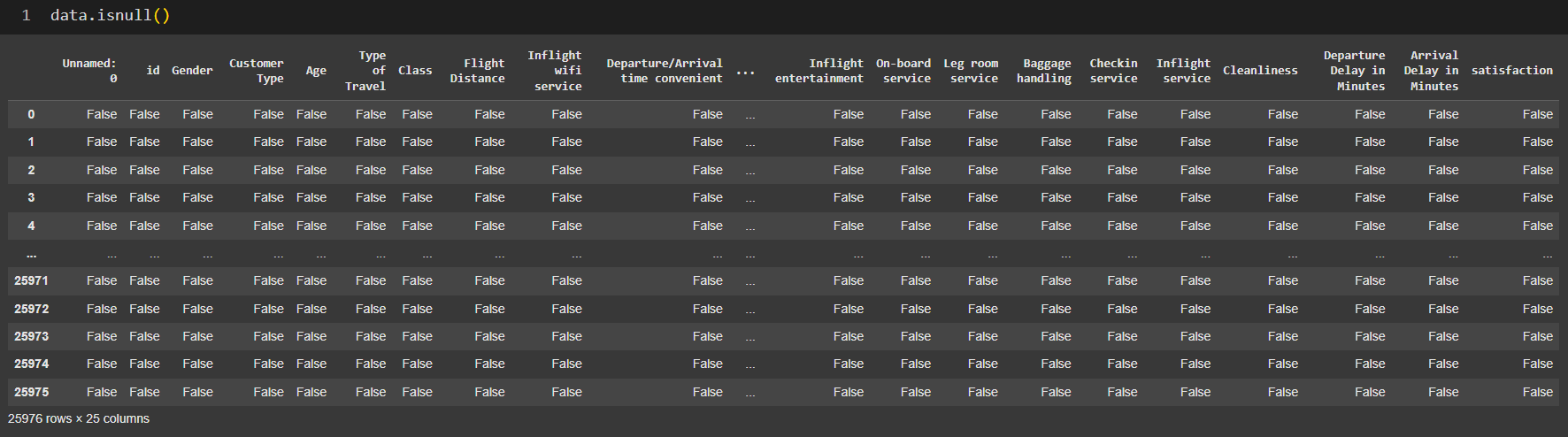




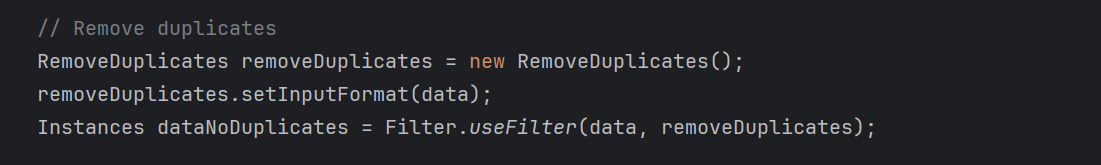
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## **Data Cleaning Process**

### Handling missing values.

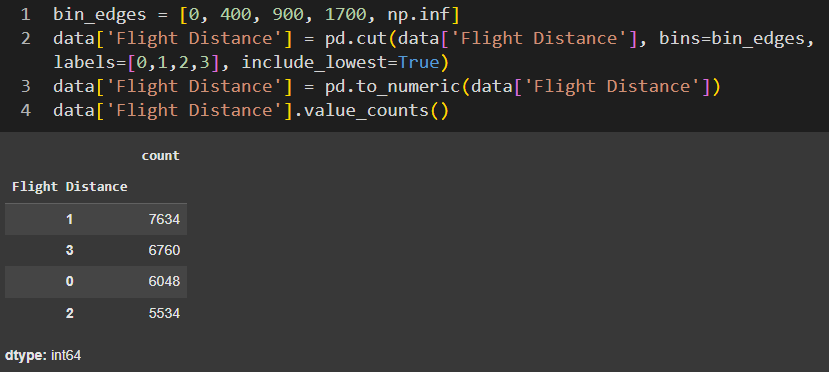
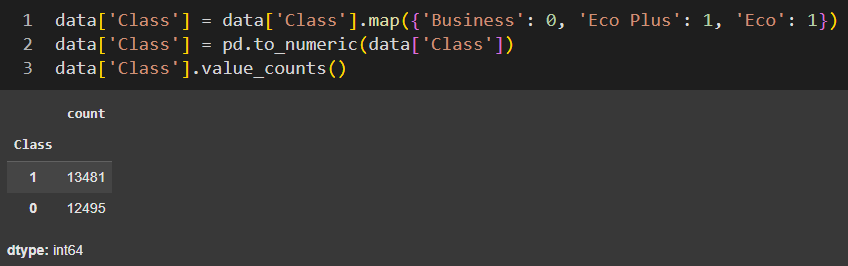
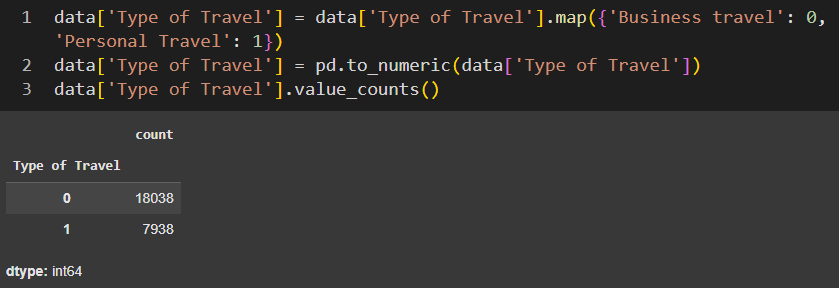
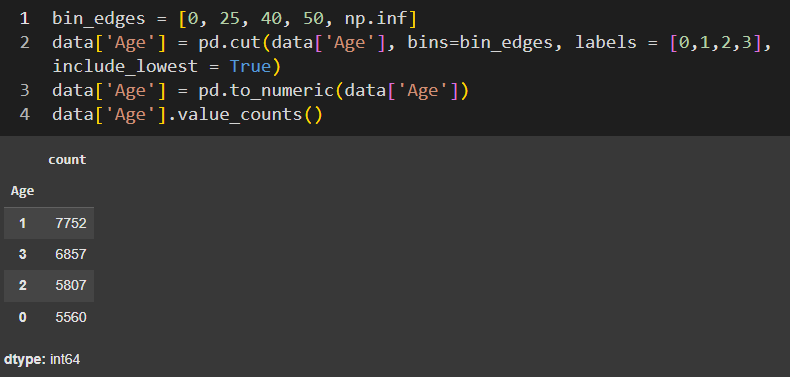
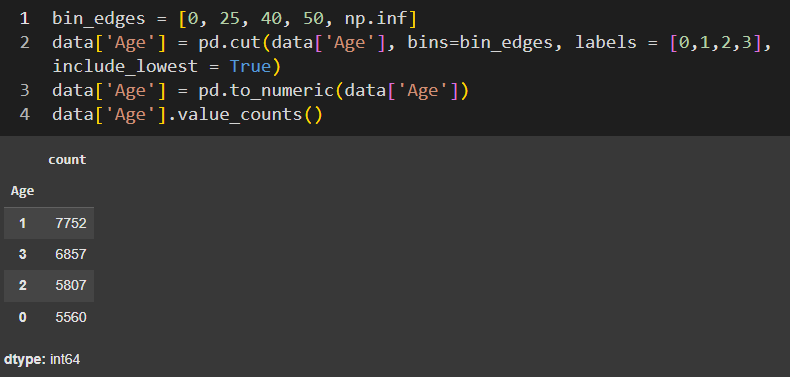
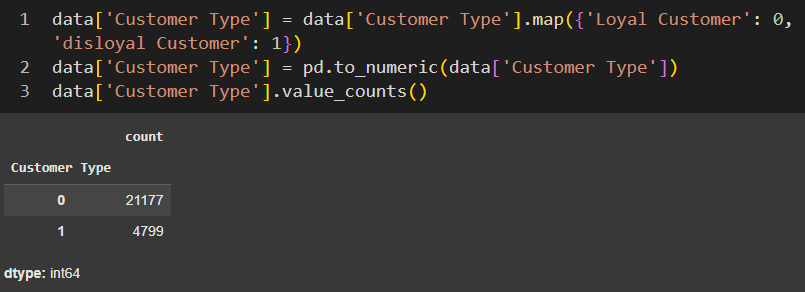
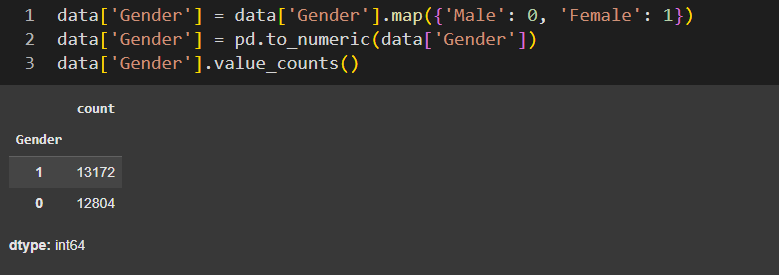


### Removing duplicates.

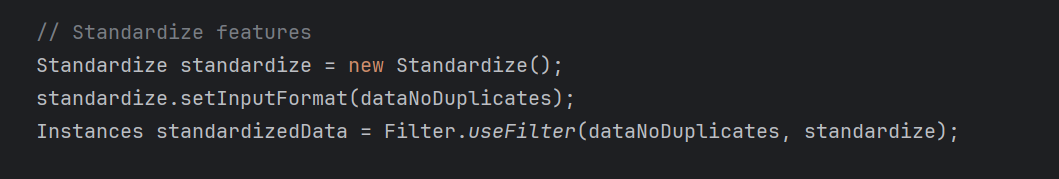


## **Data Transformation**

### Encoding categorical variables:

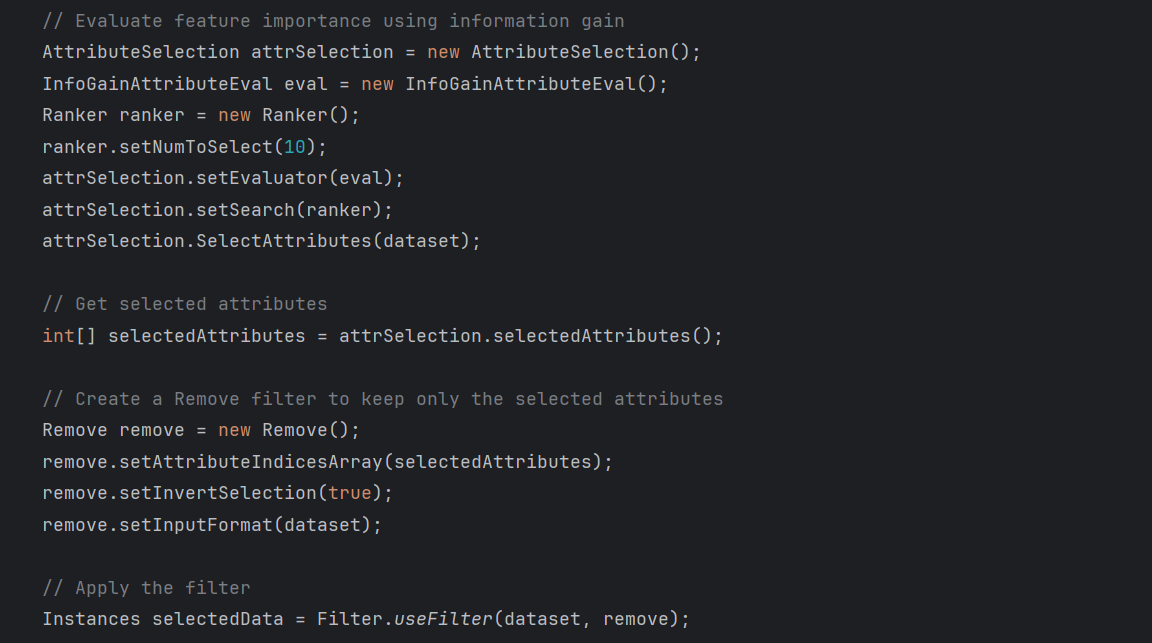


### Standardization:



### **Features selection:**

* **InfoGainAttributeEval:**
* **Calculate Entropy**: Measures the impurity or randomness in the dataset.
* **Compute Information Gain**: Calculates the reduction in entropy after splitting the dataset based on each feature.
* **Rank Features**: Ranks features based on their Information Gain scores.
* **Select Top Features:** Selects the highest-ranking features as the most relevant.



* **ReliefFAttributeEval:**
* **Initialize:** Select a random instance and define the number of nearest neighbors (k).
* **Find Neighbors:** Identify the k nearest instances of the same class (near-hit) and different classes (near-miss).
* **Update Weights:** Calculate how well each feature distinguishes between near-hit and near-miss instances, updating feature weights accordingly.
* **Repeat:** Iterate the process for multiple instances.
* **Select Features:** Rank features based on their weights, with higher weights indicating more important features.



**Output:** Present the final cleaned dataset.

# **CLASSIFICATION/PREDICTION ALGORITHM**

## **Objective**

* Utilizing the Weka library to implement and evaluate a classification or prediction model:
* Selecting an appropriate algorithm, training the model using a prepared dataset.
* Assessing its performance through metrics such as accuracy, precision, recall, and F1 score.
* Involving feature selection, parameter tuning, and cross-validation to optimize the model for robust predictions

## **Model Selection**

**IBk (Instance-Based K-nearest Neighbor) Algorithm:**

The IBk algorithm is chosen because of its simplicity and effectiveness in solving classification problems. The algorithm classifies data depending on the distance between the instances to labeled data points, which is a good choice for our Titanic dataset as relationships between the features and survival outcomes might be non-linear. Moreover, the IBk algorithm does not rely on strict rules about how the data is distributed, making it flexible and easy to handle different types of data.

**J48 Tree Algorithm:**

J48 Tree classifiers are flowchart-like structures that recursively split the data based on feature values to make predictions. Each internal node represents a decision rule on a feature, and each leaf represents a class label, making the model highly interpretable. J48 Tree handle both categorical and numerical data well and can model complex, non-linear relationships without requiring feature scaling. However, they are prone to overfitting, which can be mitigated through pruning or setting constraints like maximum depth. Their transparency and ability to reveal feature importance make them a popular choice for tasks like customer segmentation and medical diagnosis.

**OneRule Algorithm:**

OneR (One Rule) is a simple classification algorithm that generates one rule for prediction based on a single feature. It evaluates all features and selects the one with the lowest error rate, creating an interpretable and straightforward model. While it is less accurate than advanced methods, it is a robust choice for initial exploratory analysis and provides insights into the most informative feature. Its simplicity and quick execution make it a good candidate for small or preliminary datasets. OneR is often used for comparison and to identify key predictors.

**ZeroRule Algorithm:**

ZeroR is a baseline classification algorithm that predicts the majority class or the mean value in the dataset. It does not consider any features and is used primarily as a benchmark to evaluate the performance of more complex models. Its simplicity makes it fast and reliable for initial comparisons. While it cannot capture any patterns or relationships in the data, it is useful to assess whether a model adds value beyond simple majority-rule prediction. ZeroR’s primary purpose is to serve as a reference for evaluating model effectiveness.

**NaiveBayes Algorithm:**

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle. It assumes that features are conditionally independent given the class label, which simplifies computations and makes it highly efficient, even for large datasets. Despite its "naive" assumption, the algorithm often performs well in practice, particularly with categorical data. Naive Bayes is chosen for its efficiency, ability to handle categorical data effectively, and straightforward probability-based outputs, making it ideal for classifying survival outcomes in the Titanic dataset. Its simplicity and robustness with missing values further justify its selection.

**Logistic Algorithm:**

Logistic is a statistical method for binary or multi-class classification that models the probability of class membership using a sigmoid function. It assumes a linear relationship between the input features and the log-odds of the target class, making it interpretable and easy to implement. Despite its simplicity, it performs well with linearly separable data and is widely used in applications like spam detection and medical diagnosis. Logistic Regression also provides probability-based outputs, which are useful for decision-making. Its efficiency and effectiveness with small to medium-sized datasets make it a popular choice for initial modeling.

**Support Vector Machines Algorithm:**

Support Vector Machines are powerful classification algorithms that aim to find the hyperplane that best separates classes in the feature space. SVM works well with high-dimensional data and handles linear and non-linear relationships using kernels like polynomial or radial basis function (RBF). It is robust to overfitting, especially in cases where the number of dimensions exceeds the number of data points. SVMs are chosen for their ability to create complex decision boundaries and perform well with structured datasets. However, they can be computationally intensive, especially with large datasets, and may require careful tuning of hyperparameters like the kernel and regularization parameter.

Random Forest Algorithm:

**AdaBoostM1 Algorithm:**

**Implementation Process**

Detail the steps to convert data to ARFF format and integrate Weka into the program. Mention any challenges faced during implementation.

**Results**Share initial results, including accuracy, precision, recall, and runtime.

# **Improvement of Results**

## **Objective**

Enhance the model's performance using clustering, different algorithms, or advanced data analysis techniques.

## **Methodology**

Explain the additional algorithm or improvement method used

## **Comparison of Results**

[COMPARISON TABLE FOR DATA MINING](https://docs.google.com/spreadsheets/d/1sKG1N3GNLXkOYt-oRhvd8jmecnDNiZZSq-kASfRD0b8/edit?usp=sharing)

# **Model Evaluation**

## **Objective**

Evaluate the final models using 10-fold cross-validation.

## **Performance Metrics**

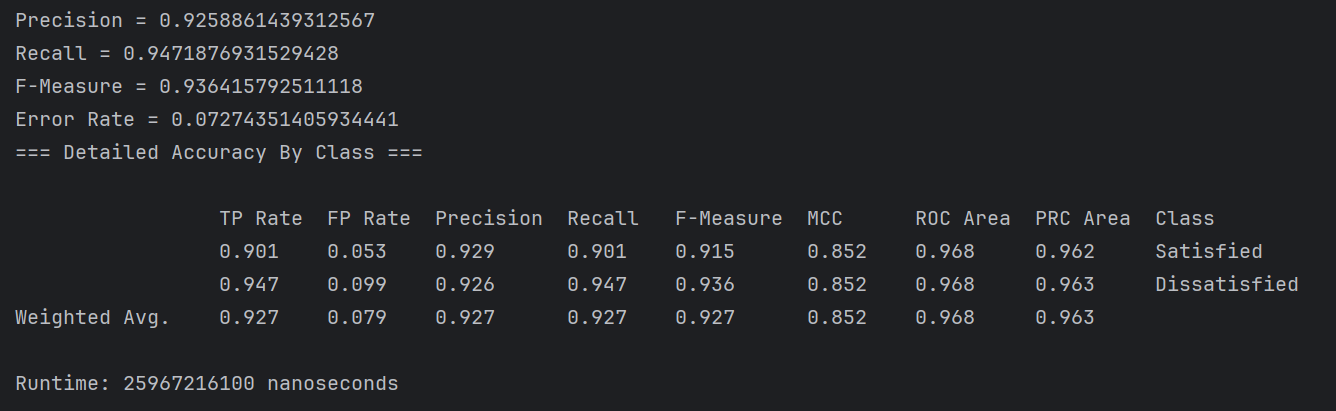
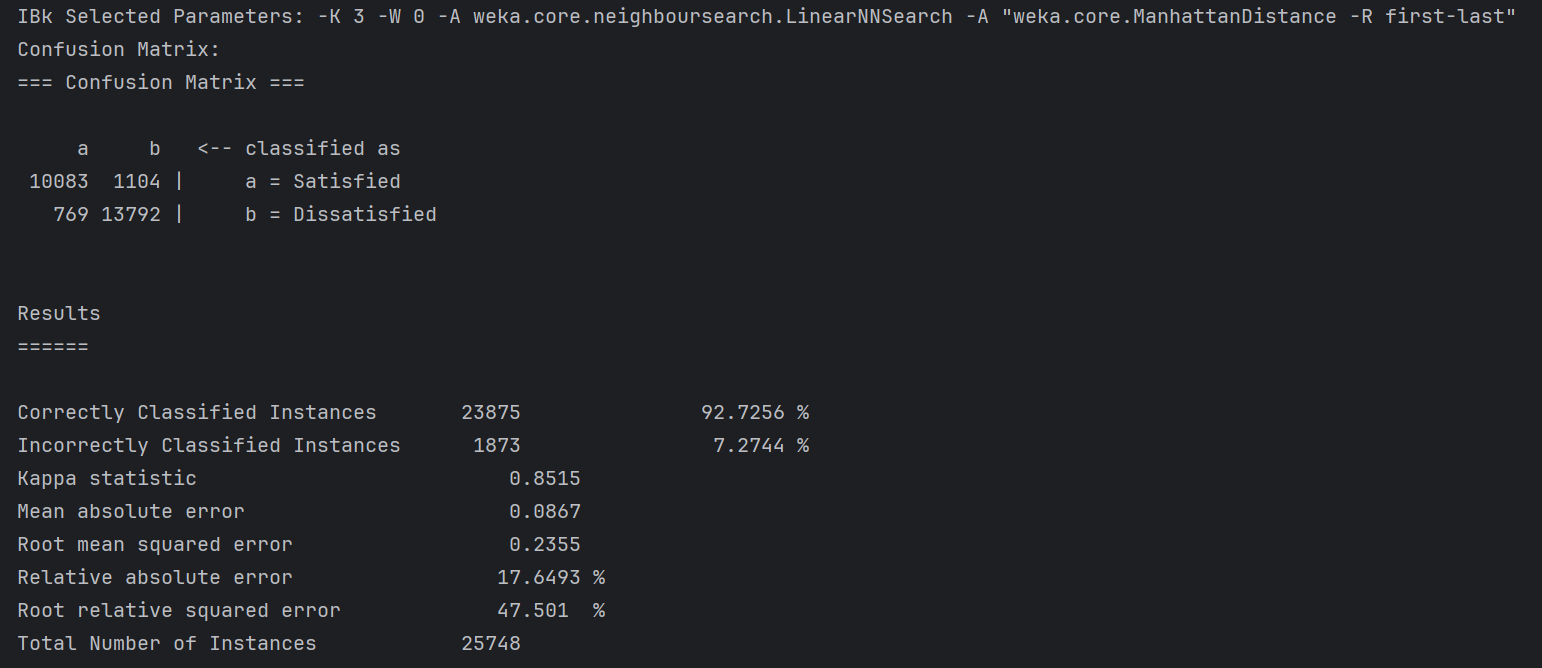
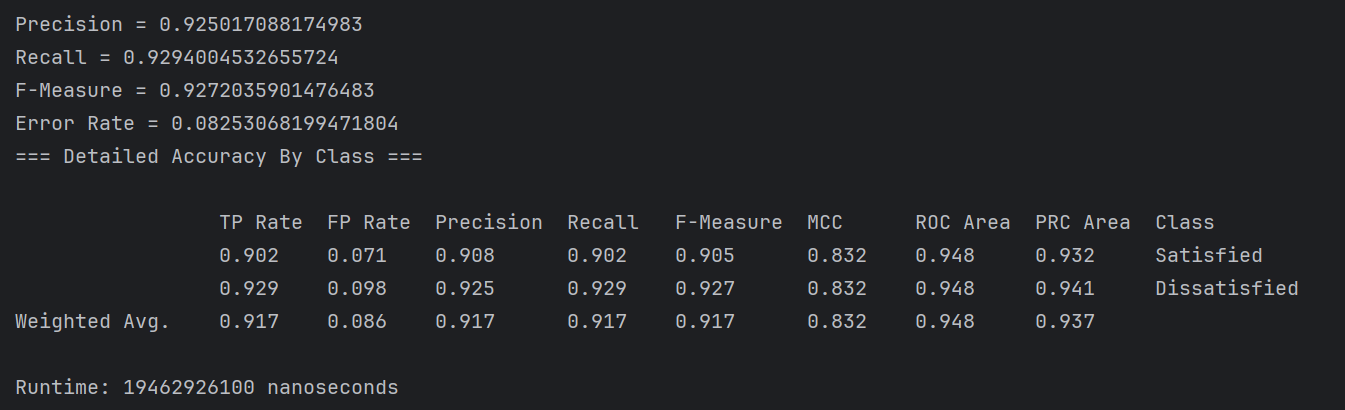
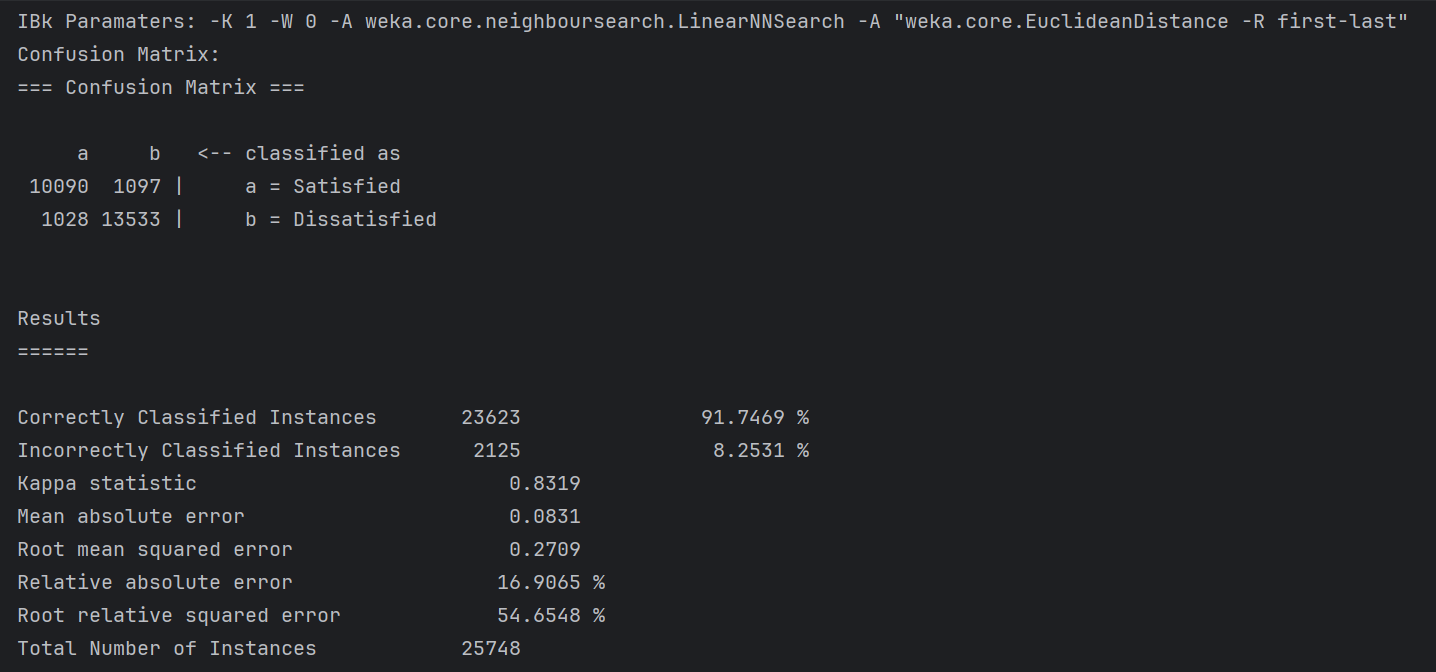
Present metrics such as accuracy, F1-score, and runtime for all models.

**Naive Bayes Algorithm:**

### 

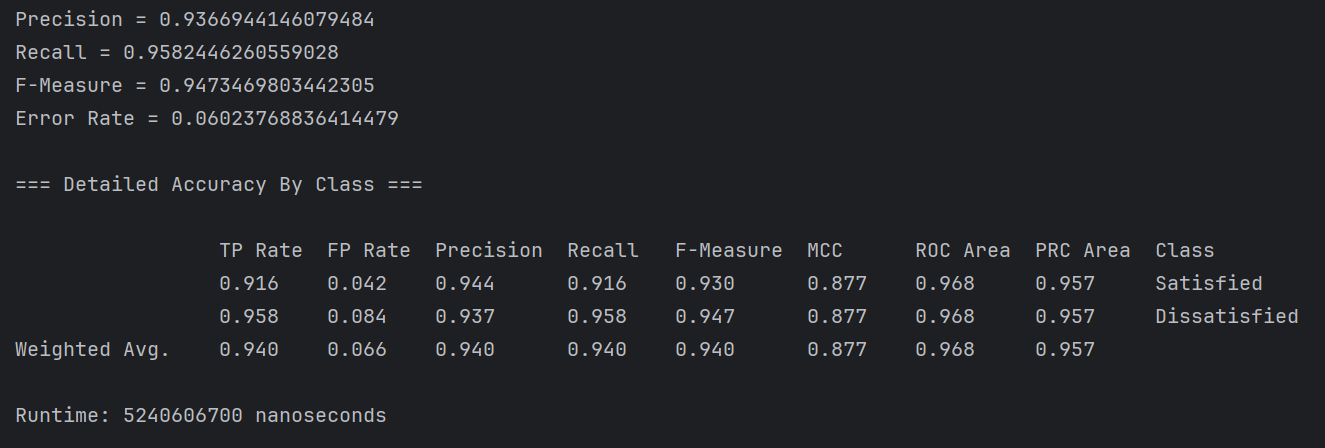
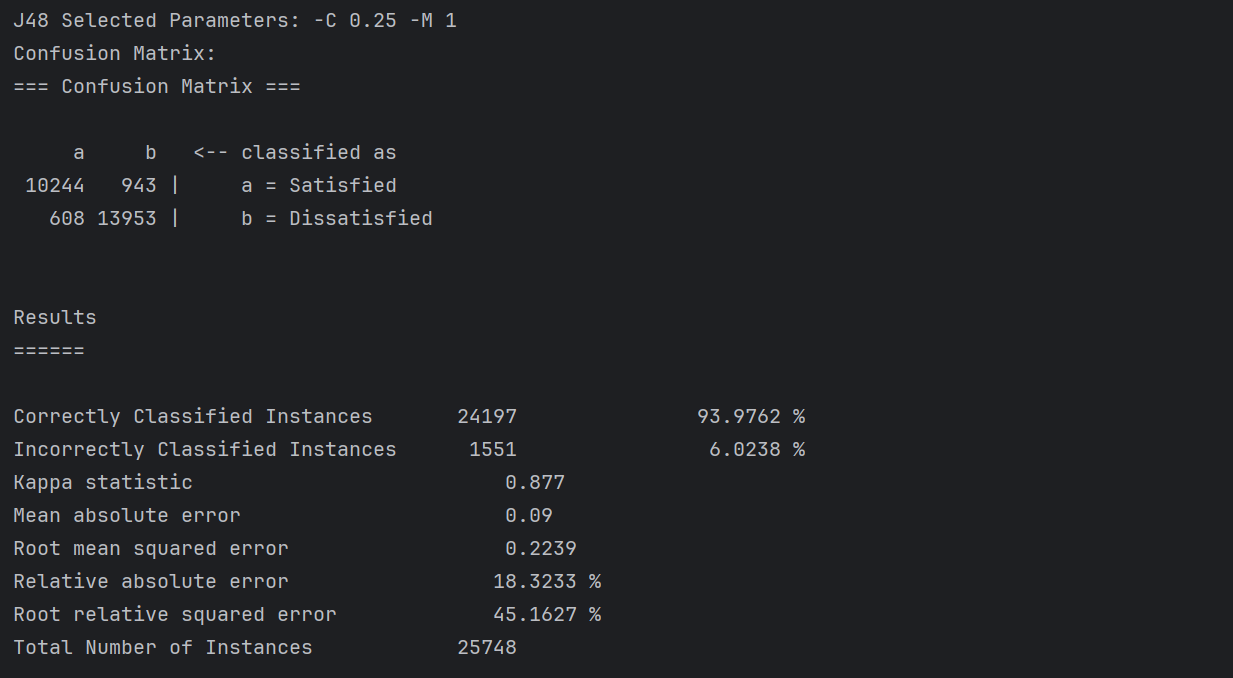
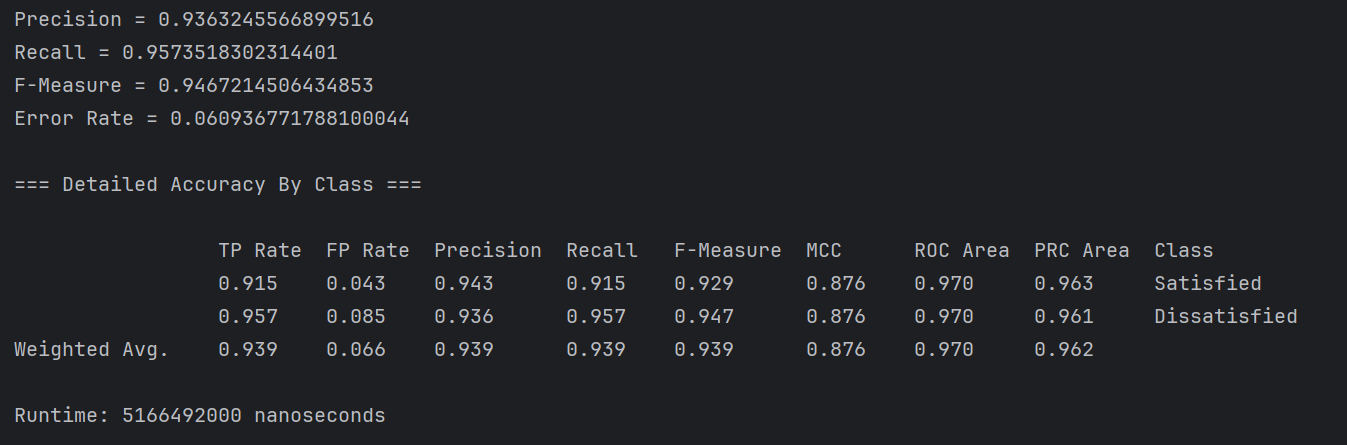
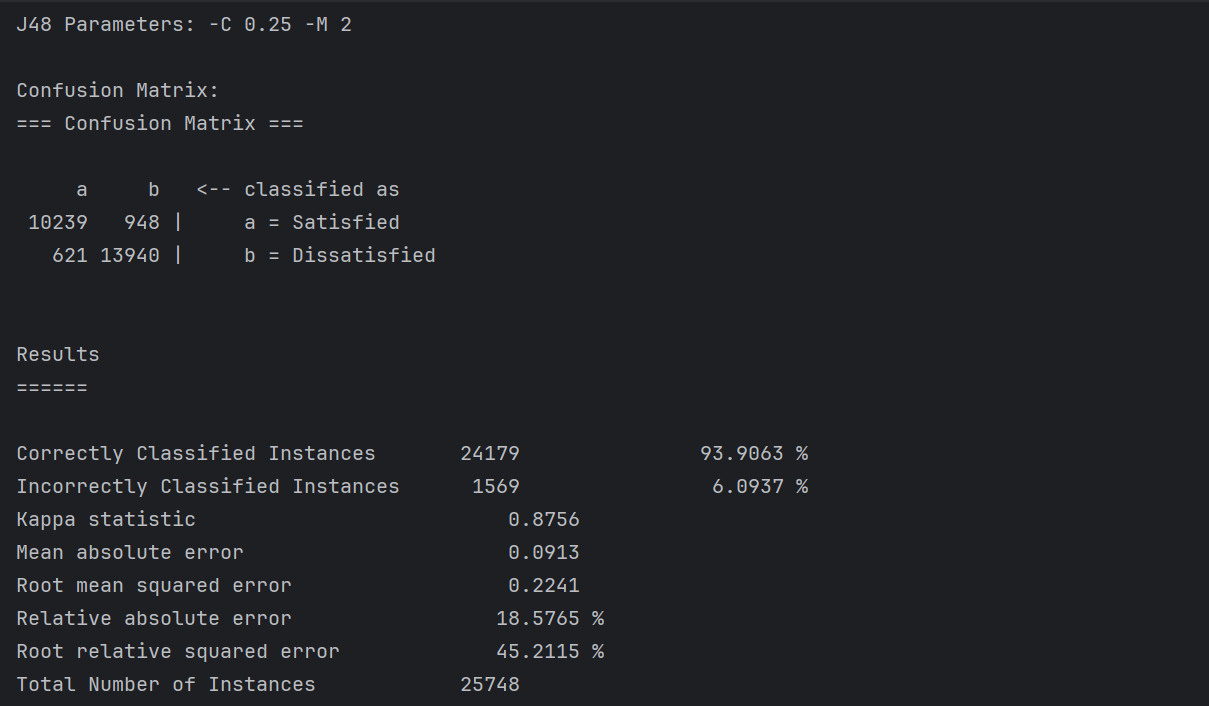
### 

### **IBk Algorithm:**

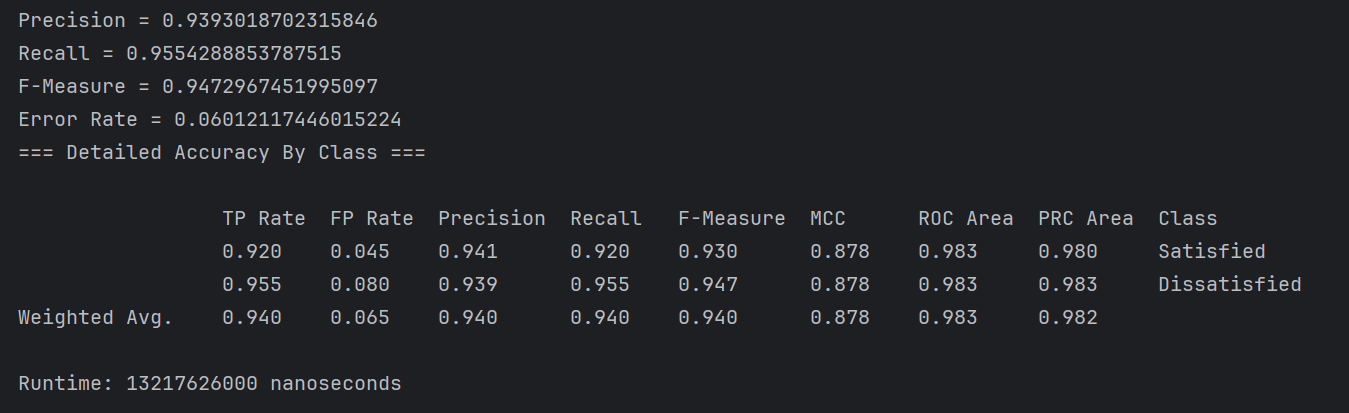
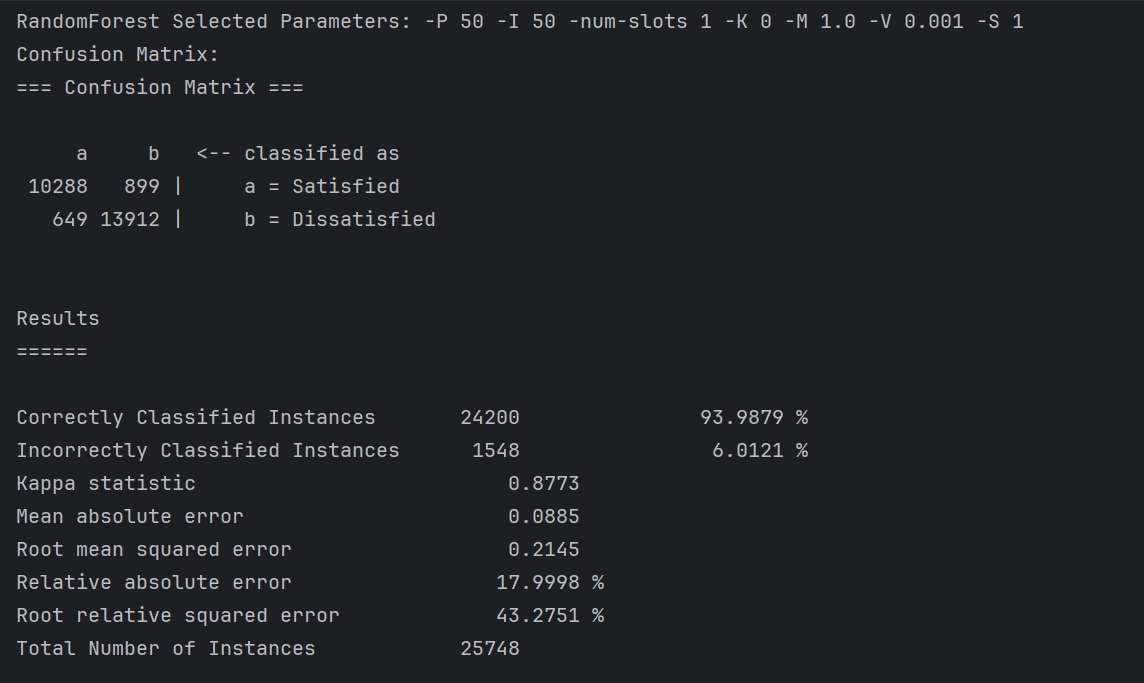
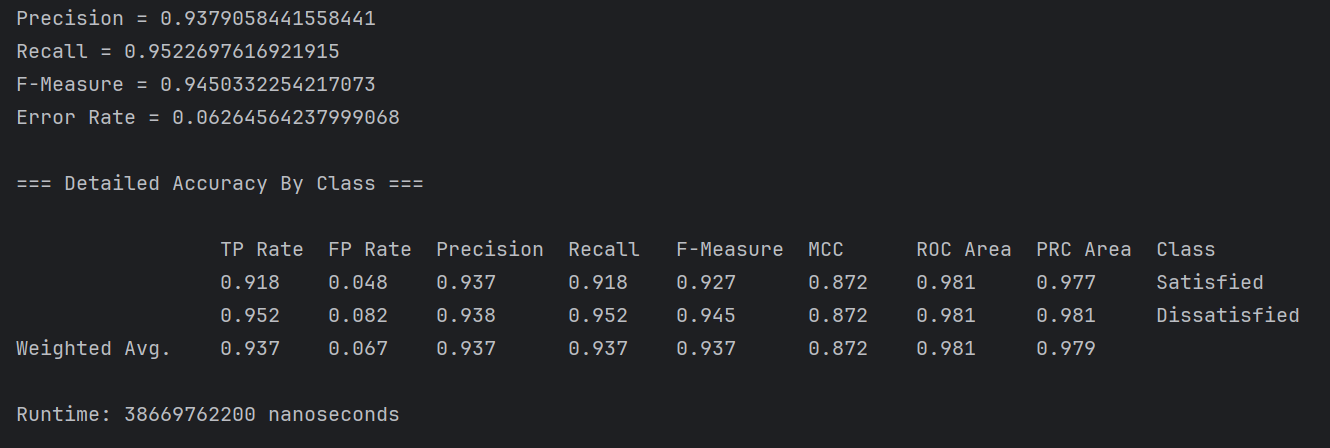
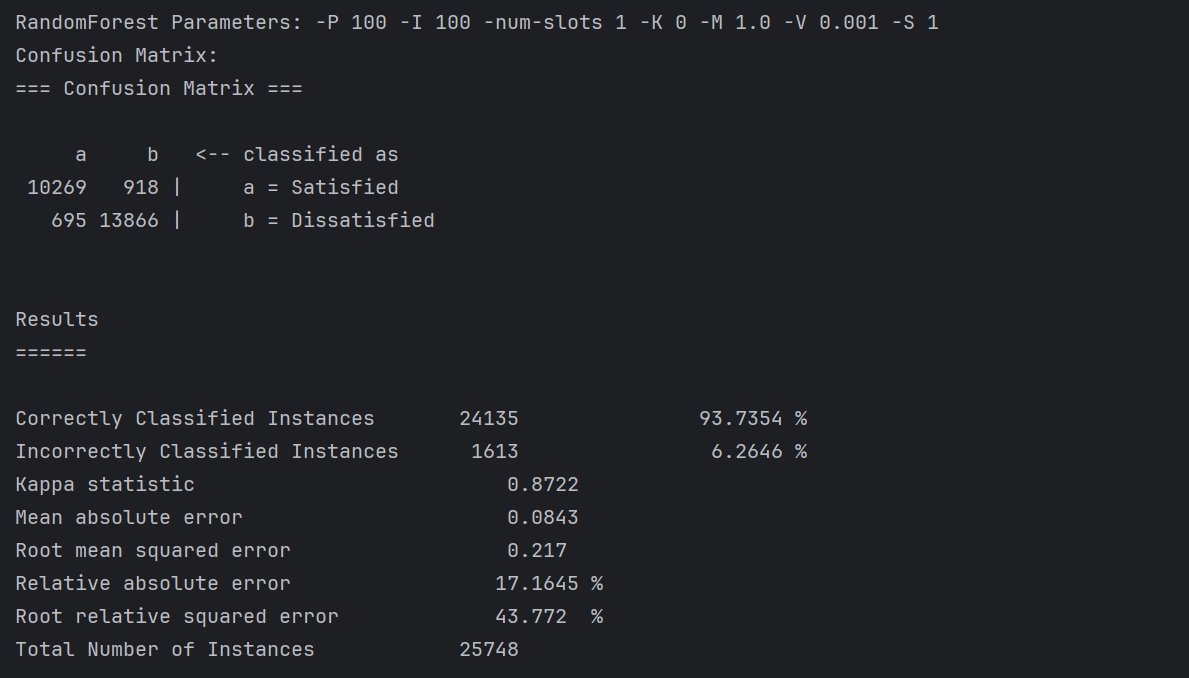


### **Decision Tree Algorithm:**

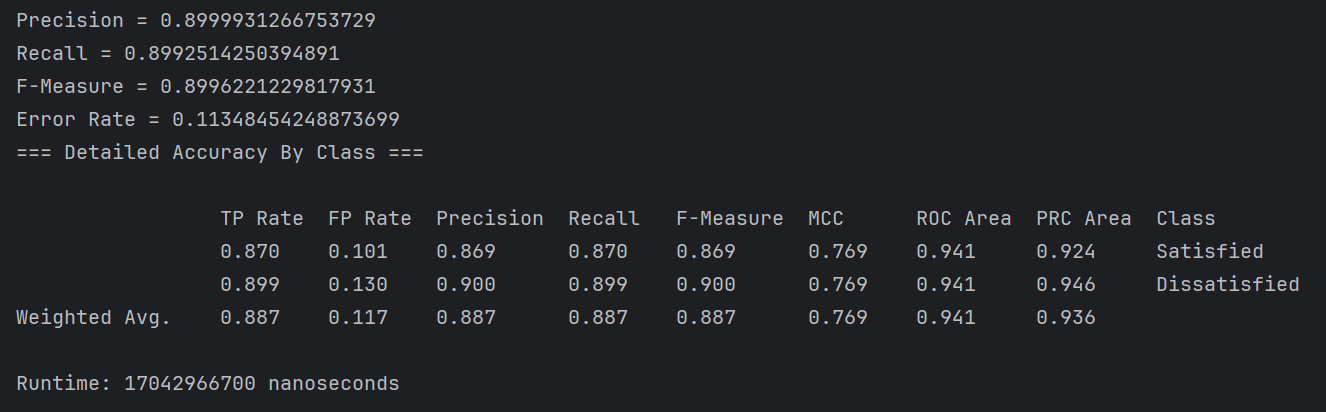
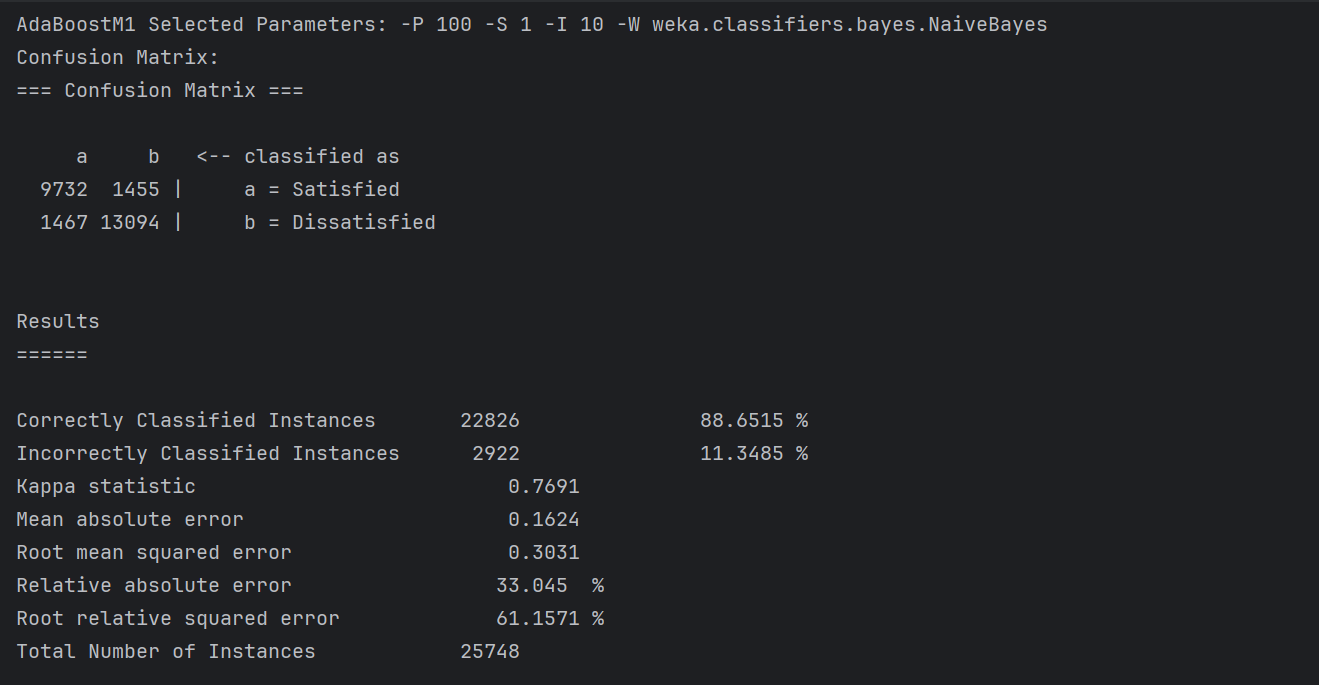
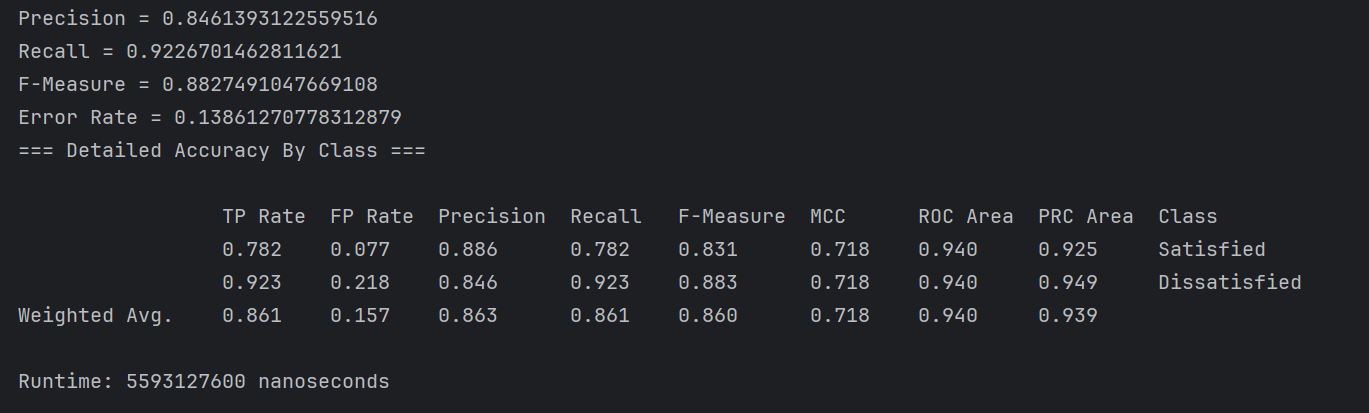
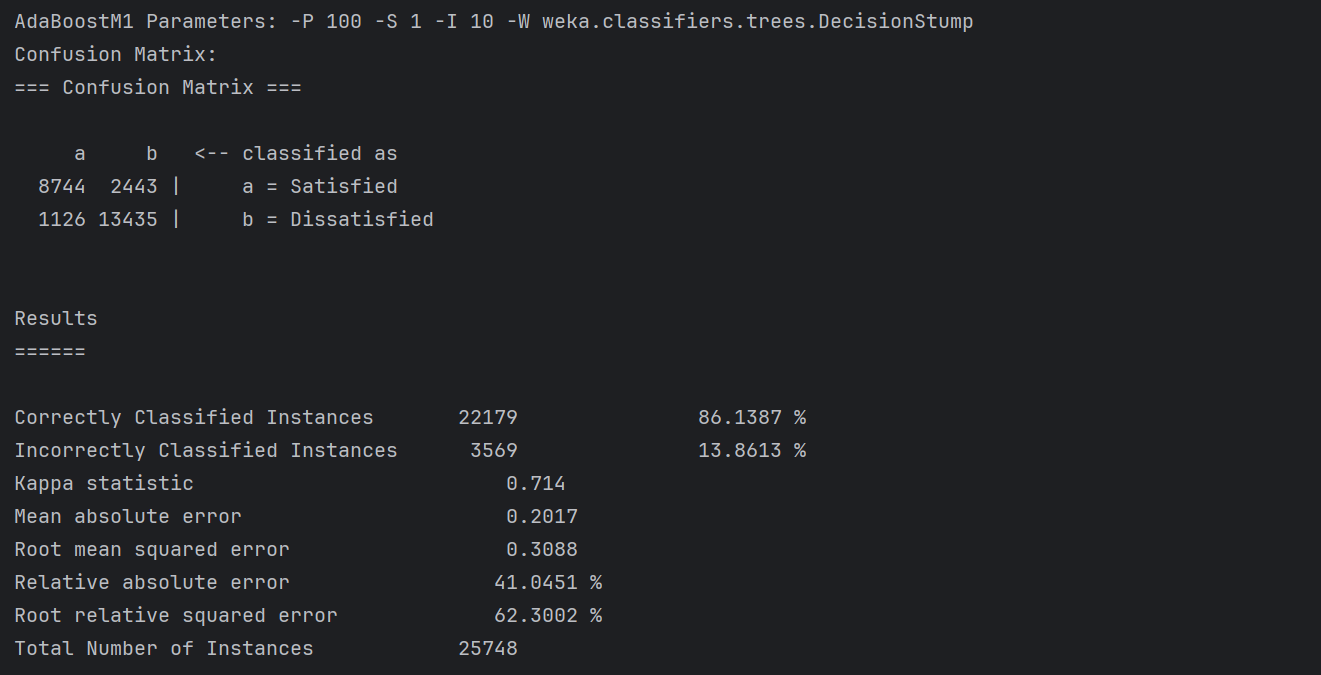
#### J48 Algorithm:



#### Random Forest Algorithm:

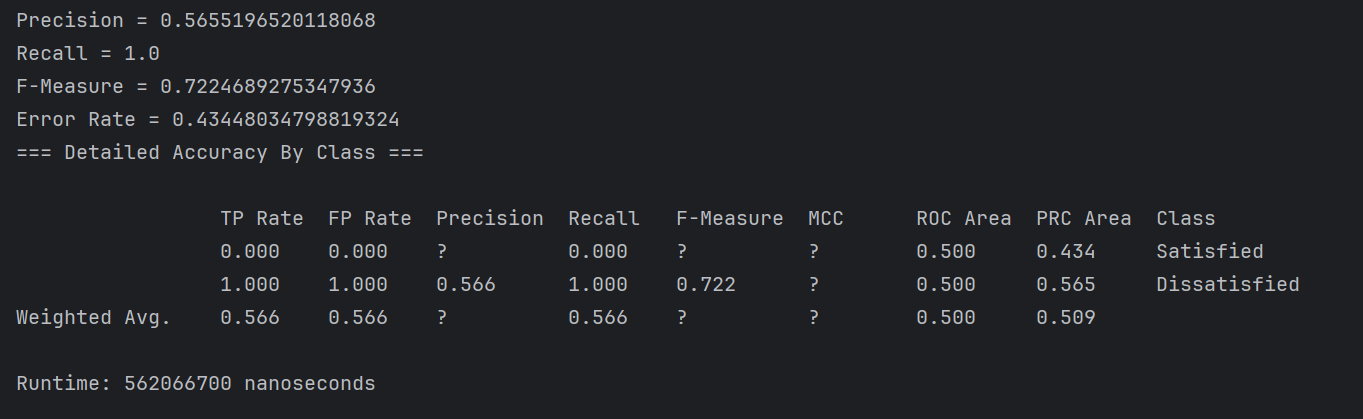
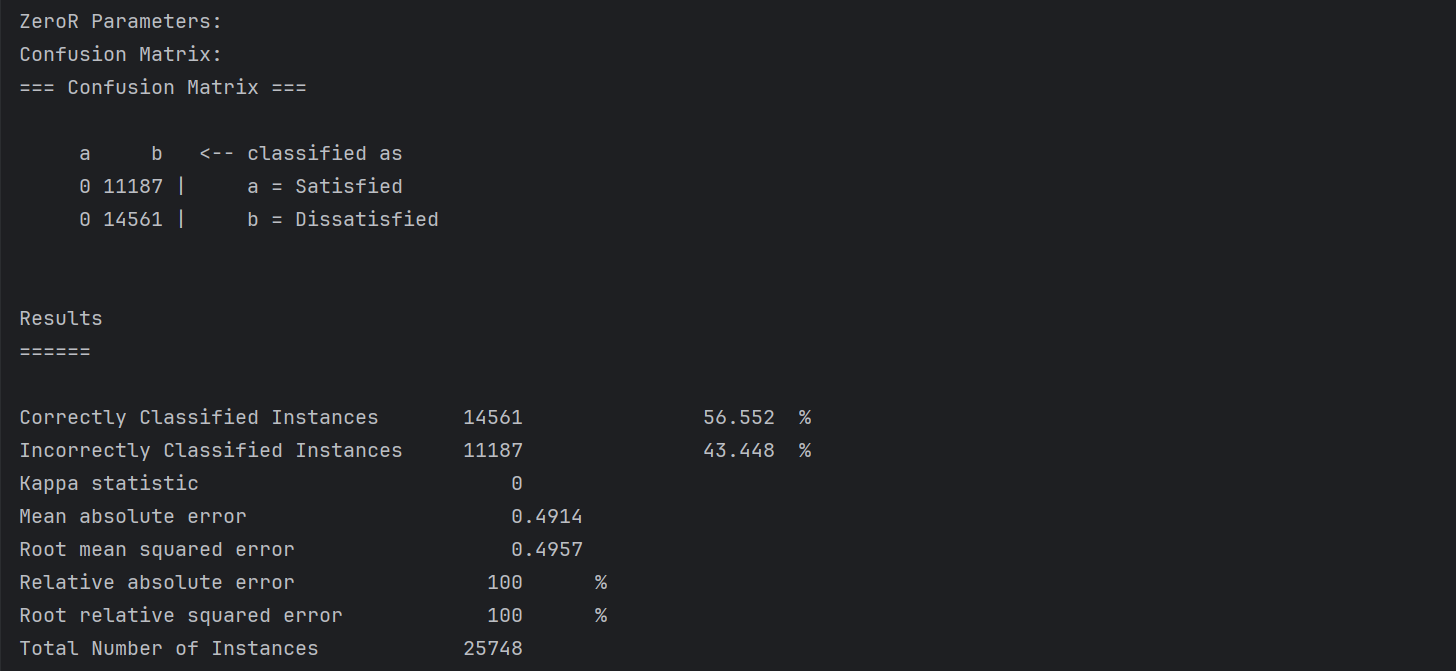


### **AdaBoostM1 Algorithm:**

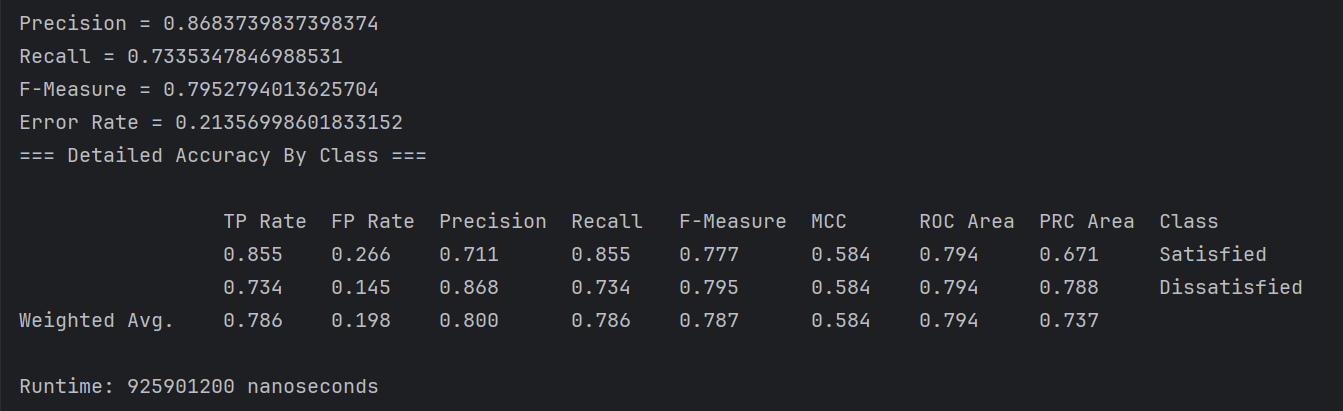
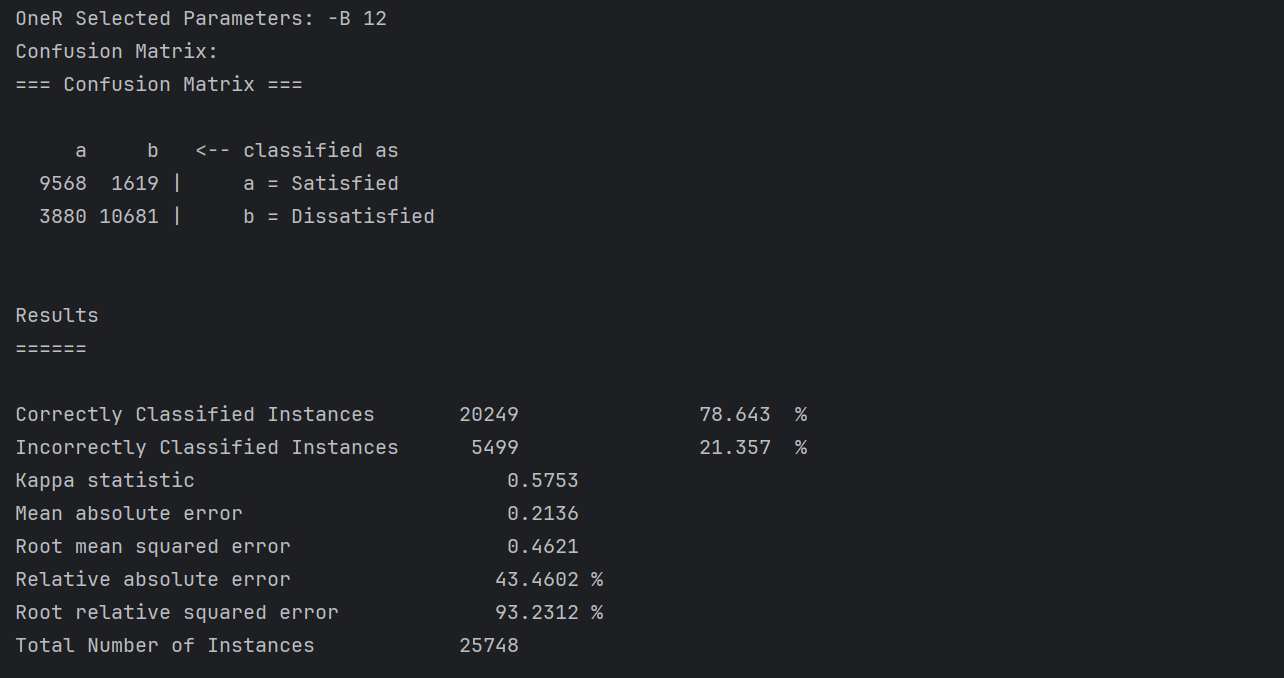
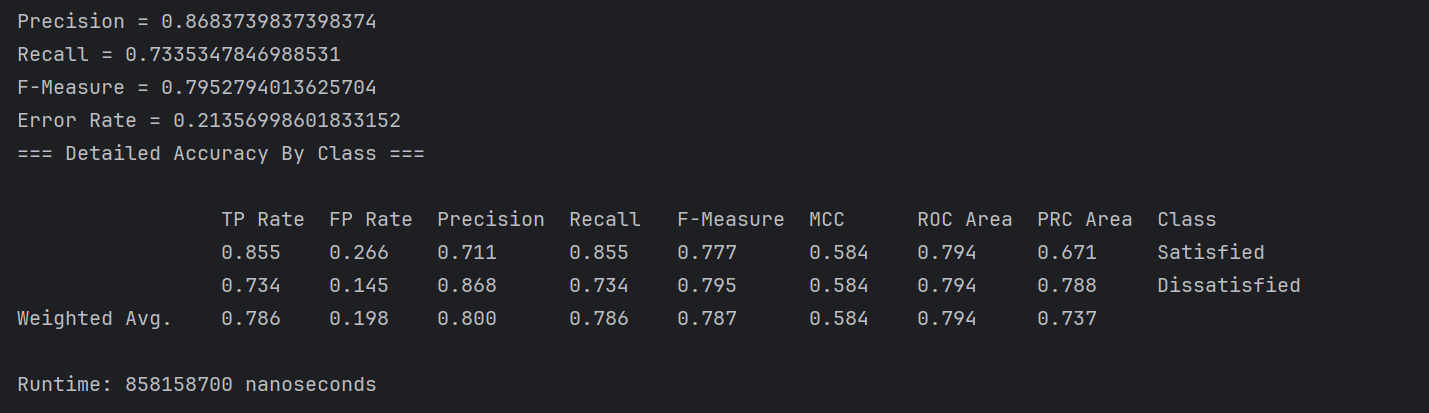
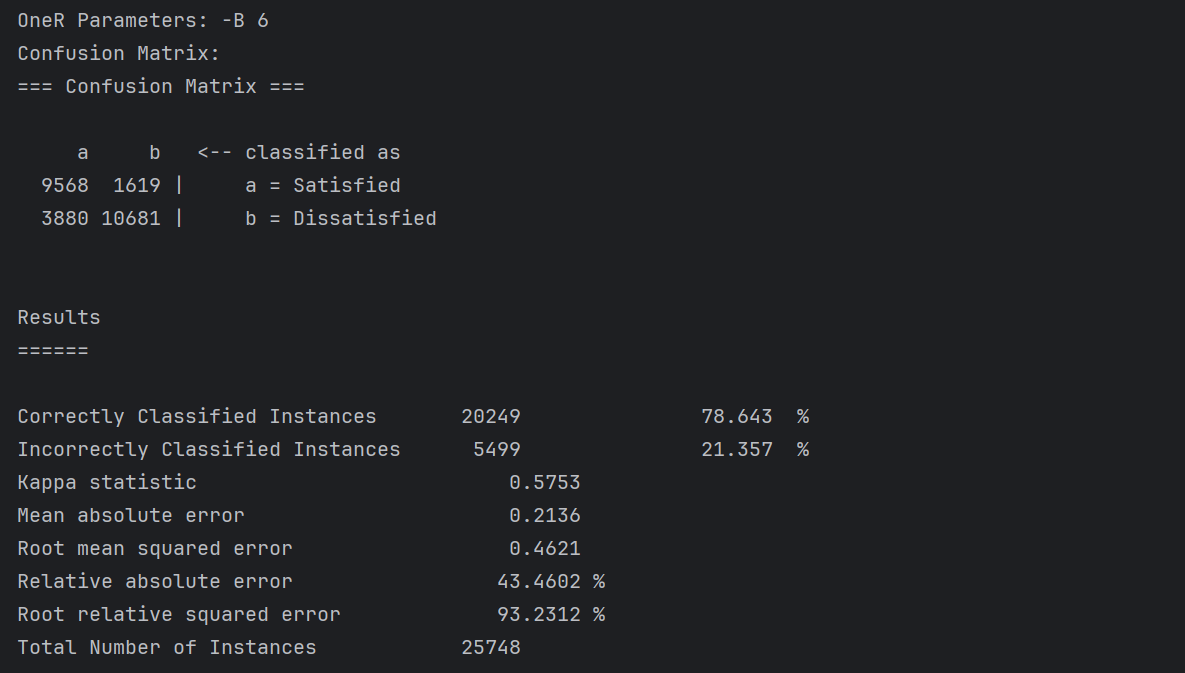


### **Rules Algorithm:**

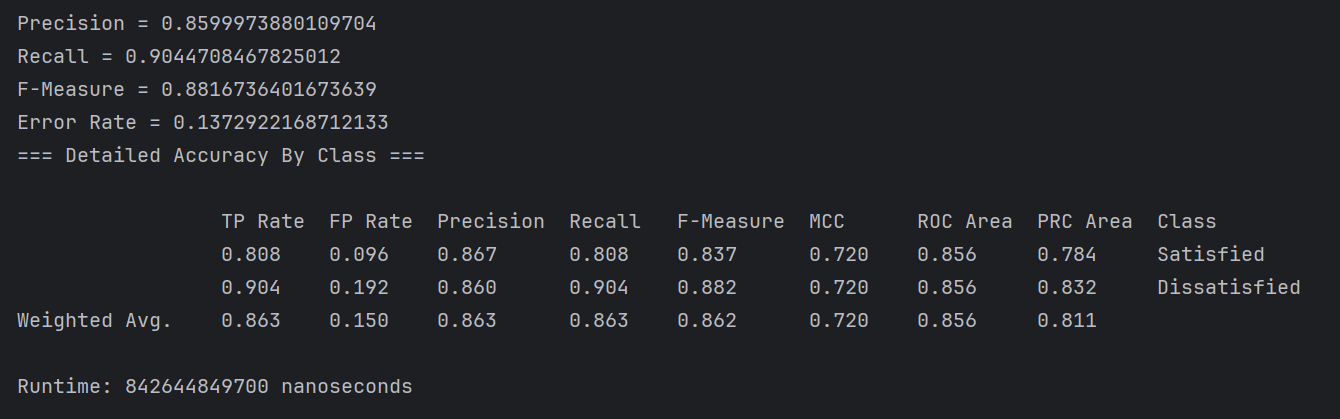
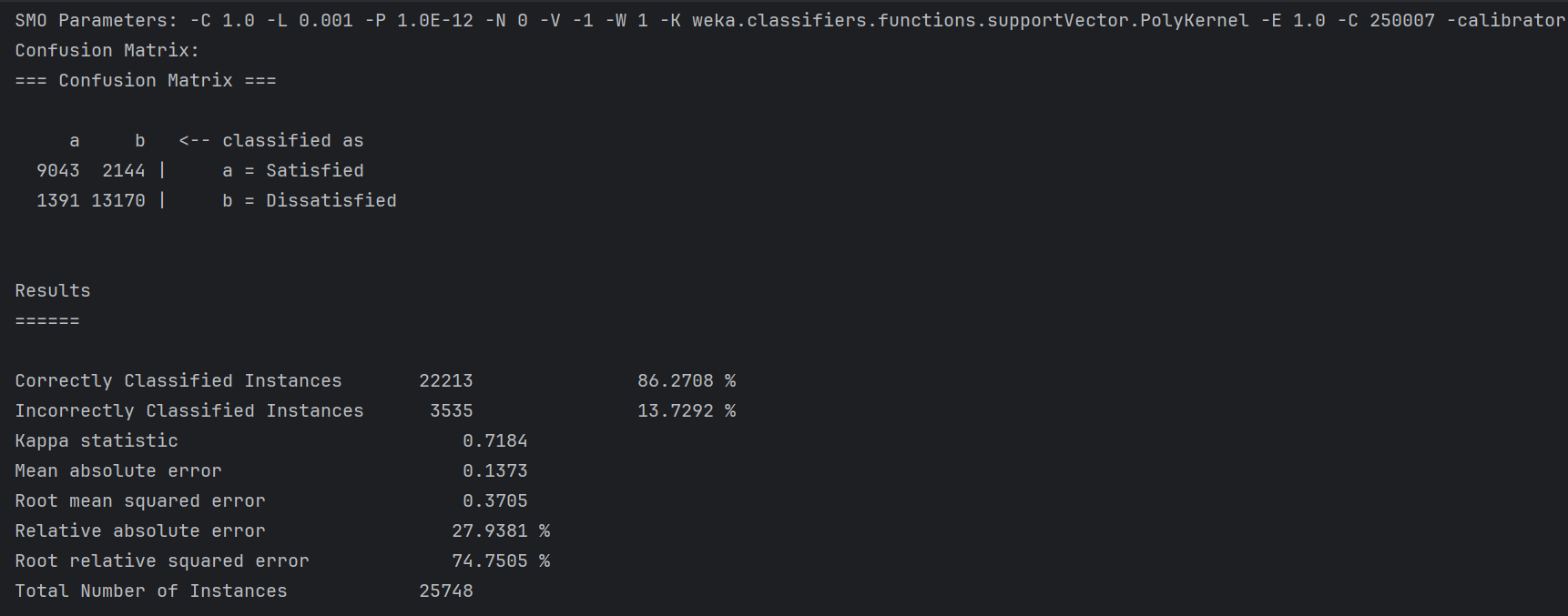
#### ZeroR Algorithm:



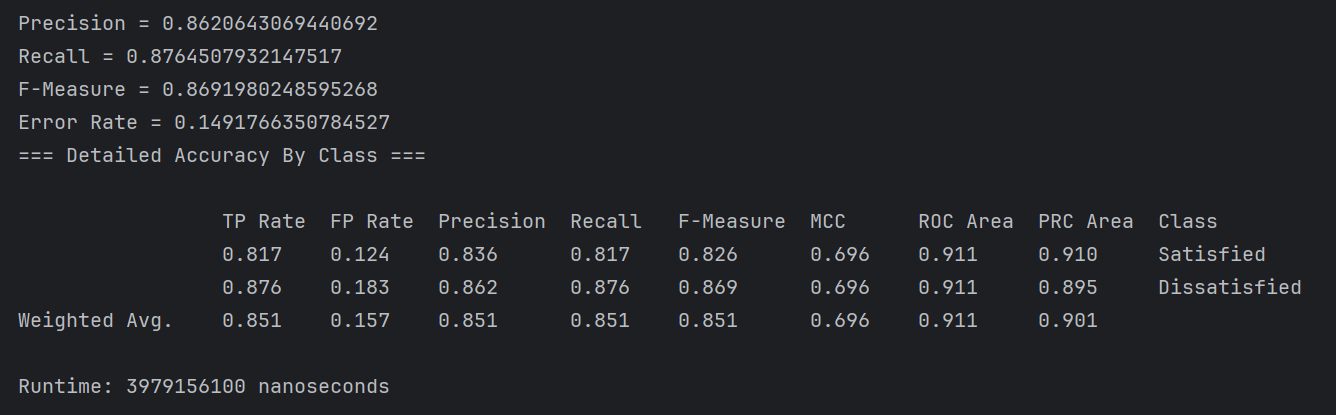
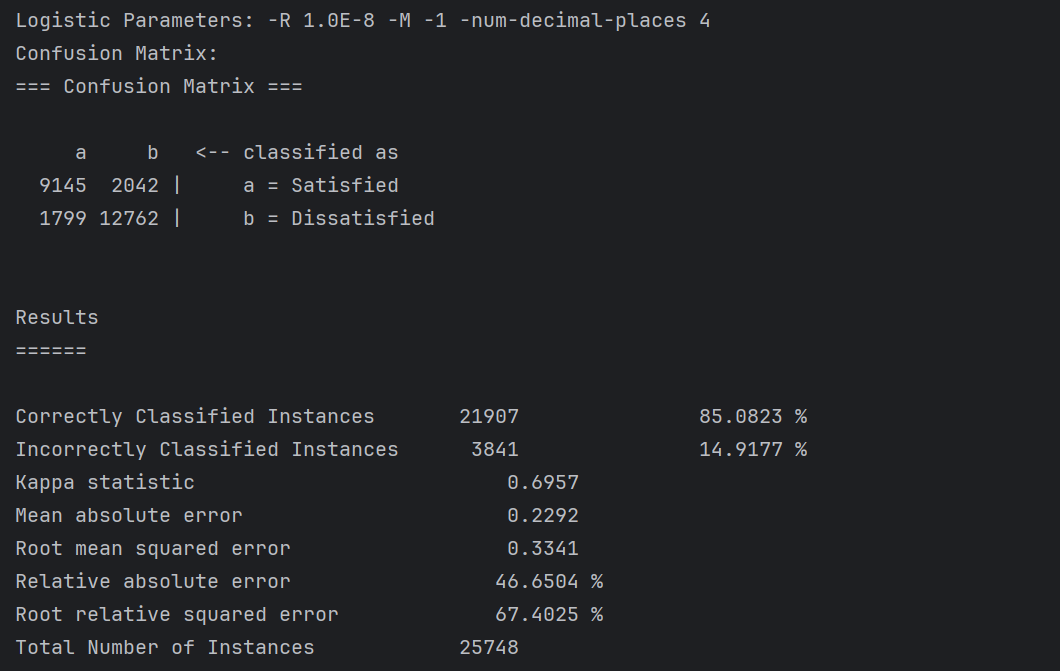
#### OneR Algorithm:

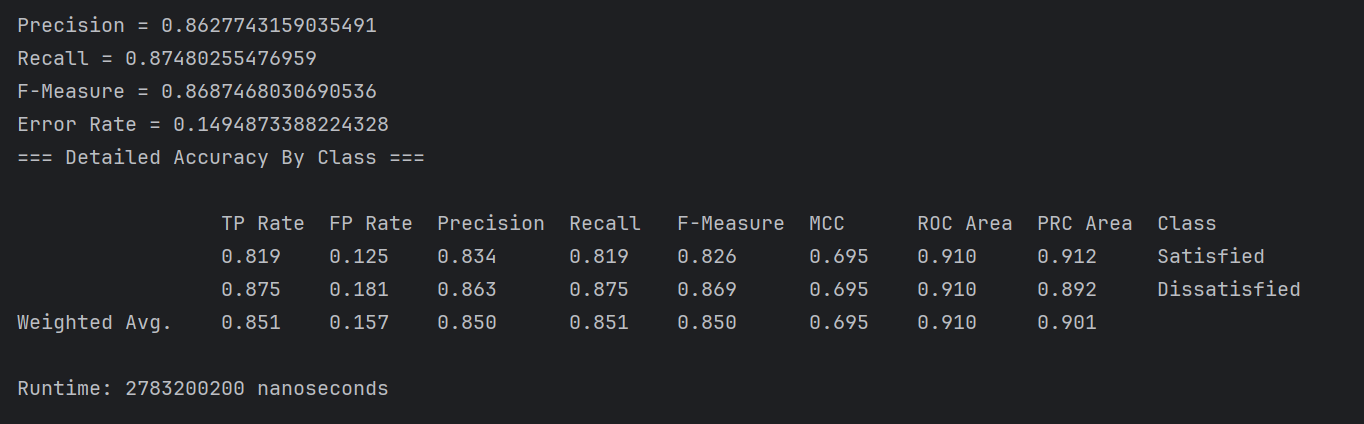
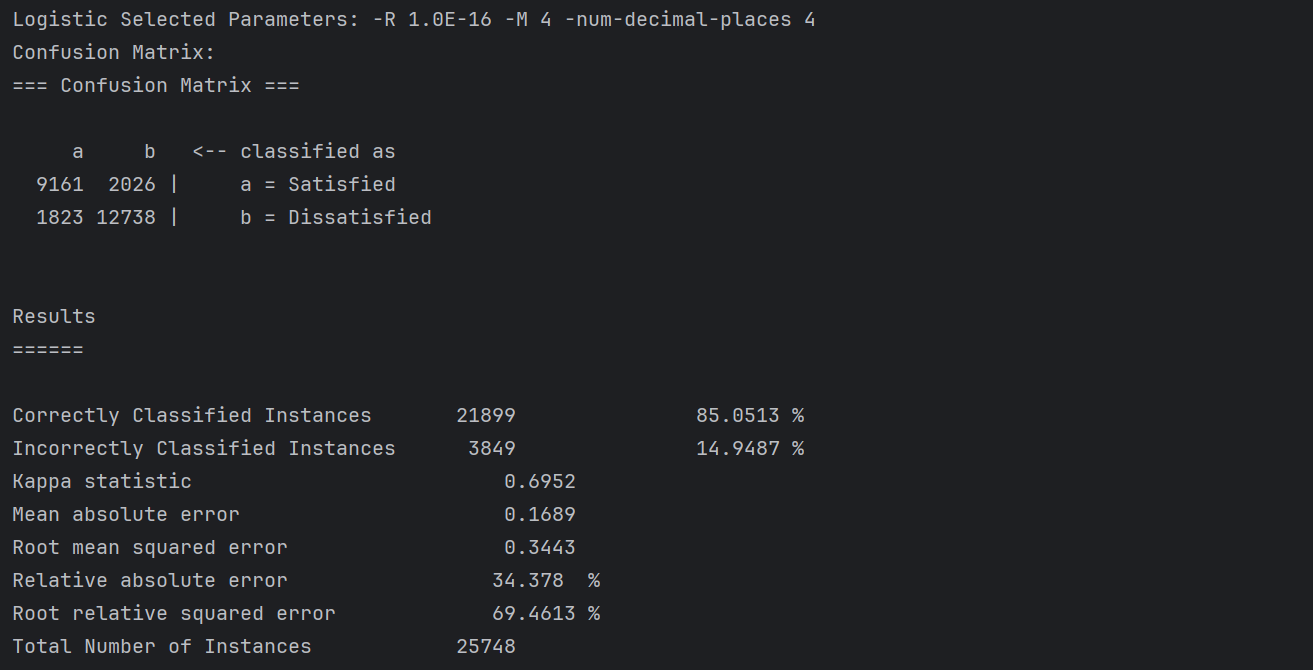


### **SMO (Sequential Minimal Optimization) Algorithm:**



### **Logistic Algorithm:**





## 

## **Analysis of Results**

**NaiveBayes Algorithm:**

|  |  |
| --- | --- |
| **Metric** | **Interpretation** |
| Correctly Classified Instances | 22182 instances were classified (86.1504 %).  ⇒ The classification accuracy indicates that Naive Bayes performs well, achieving a solid level of correct predictions, making it a reliable choice for categorical data. |
| Kappa Statistic | 0.7175.  ⇒ The Kappa statistic reflects moderate to strong agreement between predictions and actual class labels, showing that the model performs significantly better than random classification |
| Mean Absolute Error (MAE) | 0.1802.  ⇒ The MAE indicates that the average magnitude of errors is relatively low, suggesting that the model makes fairly accurate predictions. |
| Root Mean Squared Error (RMSE) | 0.3241.  ⇒ The RMSE reflects the average squared deviation of predictions from actual values, showing that larger errors are not overly frequent in the model's predictions. |
| Relative Absolute Error (RAE) | 36.6692 %.  ⇒ The RAE shows that the model's error is around 36.67% of the error of a naive predictor (e.g., predicting the mean), highlighting the model's effectiveness. |
| Root Relative Squared Error (RRSE) | 65.3864 %.  ⇒ The RRSE indicates that the model's squared error is approximately 65.39% of the baseline error, which, while not perfect, demonstrates decent predictive performance. |

### **SMO (Sequential Minimal Optimization) Algorithm:**

|  |  |
| --- | --- |
| Correctly Classified Instances | Pre-Tuning: 22,213 instances correctly classified (86.2708%).  Post-Tuning: 22,212 instances correctly classified (86.2669%).  ⇒ The performance remains almost the same, with no meaningful change after tuning. |
| Kappa Statistic | Pre-Tuning: 0.7184.  Post-Tuning: 0.7183.  ⇒ No noticeable improvement in agreement between predicted and actual classifications. |
| Mean Absolute Error (MAE) | Pre-Tuning: 0.1373.  Post-Tuning: 0.1373.  ⇒ Unchanged. |
| Root Mean Squared Error (RMSE) | Pre-Tuning: 0.3705.  Post-Tuning: 0.3706.  ⇒ Minimal change. |
| Relative Absolute Error (RAE) | Pre-Tuning: 27.9381%.  Post-Tuning: 27.946%.  ⇒ Insignificant increase. |
| Root Relative Squared Error (RRSE) | Pre-Tuning: 74.7505%.  Post-Tuning: 74.7611%.  ⇒ Insignificant increase. |

**IBk Algorithm:**

|  |  |
| --- | --- |
| Correctly Classified Instances | Pre-Tuning: 23,623 instances correctly classified (91.7469%).  Post-Tuning: 23,875 instances correctly classified (92.7256%).  ⇒ A noticeable improvement in classification accuracy after tuning. |
| Kappa Statistic | Pre-Tuning: 0.8319.  Post-Tuning: 0.8515.  ⇒ The agreement between predictions and actual values improved, with the post-tuning kappa statistic indicating a stronger relationship. |
| Mean Absolute Error (MAE) | Pre-Tuning: 0.0831.  Post-Tuning: 0.0867.  ⇒ Minor increase. |
| Root Mean Squared Error (RMSE) | Pre-Tuning: 0.2709.  Post-Tuning: 0.2355.  ⇒ Improved. |
| Relative Absolute Error (RAE) | Pre-Tuning: 16.9065%.  Post-Tuning: 47.501%.  ⇒ Notable improvement. |

**ZeroR Algorithm:**

|  |  |
| --- | --- |
| Correctly Classified Instances | 14561 Correctly Classified Instances (56.552 %).  ⇒ The accuracy represents the majority class baseline, as ZeroR predicts the most frequent class without considering any features. It serves as a benchmark for evaluating the performance of more complex models. |
| Kappa Statistic | 0  ⇒ A Kappa value of 0 indicates no agreement beyond random chance, which is expected since ZeroR does not utilize any information from the features. |
| Mean Absolute Error (MAE) | 0.4914  ⇒ The MAE reflects a high average error, demonstrating that ZeroR's predictions are far from the true values due to its simplistic majority-class prediction. |
| Root Mean Squared Error (RMSE) | 0.4957.  ⇒ The RMSE confirms that ZeroR has relatively high errors, particularly for misclassified instances, as it does not account for feature variability. |
| Relative Absolute Error (RAE) | 100 %.  ⇒ The RAE shows that ZeroR performs no better than a naive prediction using the mean or majority class, making it purely a baseline for comparison. |

**OneR Algorithm:**

|  |  |
| --- | --- |
| Correctly Classified Instances | Pre-Tuning: 20249 Correctly Classified Instances (78.643 %)  Post-Tuning: 20249 Correctly Classified Instances (78.643 %)  ⇒ The classification rate had no change after tuning, indicating that hyperparameter tuning might not have significantly improved the accuracy. |
| Kappa Statistic | Pre-Tuning: 0.5753.  Post-Tuning: 0.5753.  ⇒ The Kappa statistic remains the same, showing no significant change in the agreement between predictions and actual values. |
| Mean Absolute Error (MAE) | Pre-Tuning: 0.2136.  Post-Tuning: 0.2136.  ⇒ The MAE stays the same after tuning, meaning a stable MAE. |
| Root Mean Squared Error (RMSE) | Pre-Tuning: 0.4621.  Post-Tuning: 0.4621.  ⇒ The RMSE stays the same after tuning. |
| Relative Absolute Error (RAE) | Pre-Tuning: 43.4602 %.  Post-Tuning: 43.4602 %.  ⇒ The Relative Absolute Error (RAE) pre-tuning and post-tuning values are identical at 43.4602%. This suggests that the tuning process did not lead to any improvement in the model's error rate, implying that the model was already well-optimized. |

**Logistic Algorithm:**

|  |  |
| --- | --- |
| Correctly Classified Instances | Pre-Tuning: 21907 Correctly Classified Instances (85.0823 %)  Post-Tuning: 21899 Correctly Classified Instances (85.0513 %)  ⇒ The classification rate slightly decreased after tuning, indicating that hyperparameter tuning might not have significantly improved the accuracy. |
| Kappa Statistic | Pre-Tuning: 0.6957.  Post-Tuning: 0.6952.  ⇒ The Kappa statistic remains almost identical, showing no significant change in the agreement between predictions and actual values. |
| Mean Absolute Error (MAE) | Pre-Tuning: 0.2292.  Post-Tuning: 0.1689.  ⇒ The MAE improved significantly after tuning, meaning the model’s average prediction error decreased. |
| Root Mean Squared Error (RMSE) | Pre-Tuning: 0.3341.  Post-Tuning: 0.3443.  ⇒ The RMSE slightly increased after tuning, suggesting that the tuning may have led to slightly higher sensitivity to larger errors. |
| Relative Absolute Error (RAE) | Pre-Tuning: 46.6504%.  Post-Tuning: 34.378%.  ⇒ A major reduction in RAE post-tuning highlights an improvement in the model’s relative performance compared to the baseline. |

**J48 Algorithm:**

|  |  |
| --- | --- |
| Correctly Classified Instances | Pre-Tuning: 24179 Correctly Classified Instances(93.9063 %)  Post-Tuning: 24197 Correctly Classified Instances (93.9762 %)  ⇒ The classification accuracy shows a slight improvement after tuning, indicating marginally better model performance in correctly predicting the class labels. |
| Kappa Statistic | Pre-Tuning:0.8756  Post-Tuning:0.877  ⇒ The Kappa statistic remains almost identical after tuning, showing no significant improvement in the agreement between predicted and actual values. This suggests the model's consistency in classification did not change substantially. |
| Mean Absolute Error (MAE) | Pre-Tuning: 0.0913  Post-Tuning: 0.09  ⇒ A small decrease in MAE indicates that the average magnitude of prediction errors slightly improved, leading to more accurate predictions on average. |
| Root Mean Squared Error (RMSE) | Pre-Tuning: 0.2241  Post-Tuning: 0.2239  ⇒ The minimal reduction in RMSE suggests only a marginal improvement in the model's ability to minimize the influence of larger errors. |
| Relative Absolute Error (RAE) | Pre-Tuning: 18.5765 %  Post-Tuning: 18.3233 %  ⇒ A slight reduction in RAE indicates improved error consistency relative to the mean, suggesting a small but positive impact of tuning on error reduction. |

**RandomForest Algorithm:**

|  |  |
| --- | --- |
| Correctly Classified Instances | Pre-Tuning: 24135 Correctly Classified Instances (93.7354 %)  Post-Tuning: 24200 Correctly Classified Instances (93.9879 %)  ⇒ The tuning process slightly improved classification accuracy, indicating that parameter adjustments enhanced the model's ability to correctly classify instances. |
| Kappa Statistic | Pre-Tuning: 0.8722  Post-Tuning: 0.8773  ⇒ The agreement between predictions and actual values slightly improved, with the post-tuning kappa statistic indicating a bit stronger relationship. |
| Mean Absolute Error (MAE) | Pre-Tuning: 0.0843  Post-Tuning: 0.0885  ⇒ A minor increase in MAE indicates that, while accuracy improved, the average prediction error for the model slightly increased, which could suggest overfitting in some cases. |
| Root Mean Squared Error (RMSE) | Pre-Tuning: 0.217  Post-Tuning: 0.2145  ⇒ A small reduction in RMSE implies that the model's predictions are marginally closer to the actual values after tuning, reducing the influence of larger errors. |
| Relative Absolute Error (RAE) | Pre-Tuning: 17.1645 %  Post-Tuning: 17.9998 %  ⇒ A small reduction in RMSE implies that the model's predictions are marginally closer to the actual values after tuning, reducing the influence of larger errors. |

**AdaBoostM1 Algorithm:**

|  |  |
| --- | --- |
| Correctly Classified Instances | Pre-Tuning: 22179 Correctly Classified Instances(86.1387 %)  Post-Tuning: 22826 Correctly Classified Instances (88.6515 %)  ⇒ The tuning process significantly enhanced the classification accuracy, showing that parameter adjustments improved the model's ability to correctly classify more instances |
| Kappa Statistic | Pre-Tuning: 0.714  Post-Tuning: 0.7691  ⇒ The Kappa statistic improved noticeably after tuning, indicating better agreement between predicted and actual class labels. This suggests a stronger model performance with reduced classification errors. |
| Mean Absolute Error (MAE) | Pre-Tuning: 0.2017  Post-Tuning: 0.1624  ⇒ The reduction in MAE post-tuning demonstrates that the average magnitude of errors decreased, indicating more accurate predictions on average. |
| Root Mean Squared Error (RMSE) | Pre-Tuning: 0.3088  Post-Tuning: 0.3031  ⇒ A slight reduction in RMSE post-tuning reflects improved prediction accuracy by minimizing the impact of larger errors. |
| Relative Absolute Error (RAE) | Pre-Tuning: 41.0451 %  Post-Tuning: 33.045 %  ⇒ A substantial decrease in RAE shows that the overall error, relative to the mean prediction, reduced significantly, highlighting the effectiveness of the tuning process in improving model reliability. |

# **Conclusions**

This project developed a robust data mining framework, integrating models like Decision Trees, Naive Bayes, IBk, and ensemble methods. Random Forest achieved the best classification accuracy of 93.98% with a Kappa statistic of 0.8773 after tuning, demonstrating its effectiveness. AdaBoostM1 also performed well, reaching 88.65% accuracy. Baseline models like ZeroR and OneR provided valuable benchmarks.

The results highlight the importance of preprocessing and systematic evaluation in achieving reliable predictions. Future work could explore advanced techniques like deep learning and feature engineering to further enhance performance. This framework lays a strong foundation for predictive analytics in real-world applications.

# **References**

List all references, including:

* Dataset sources:
* Weka documentation and tutorials:

[weka.classifiers.functions](https://weka.sourceforge.io/doc.stable/index.html?weka/classifiers/functions/package-summary.html)

[WEKA API Tutorial: How to use WEKA in JAVA - YouTube](https://www.youtube.com/playlist?list=PLea0WJq13cnBVfsPVNyRAus2NK-KhCuzJ)

* Tools used:
* Python language using Google Colaboratory for preprocessing Data
* Java language using intelliJ IDEA, VsCode with Weka library for constructing and evaluating models