ALGORITHMIC MACHINE LEARNING

FINAL PROJECT REPORT

Bank Customer Churn Prediction

Group 7

Submitted by:

Namrata Rath

Ronil Sanjay Surve

Shreyash Mehta

Shubham Sharma

CONTENTS

1 INTRODUCTION:

As the banking industry faces increasing competition and higher customer expectations, accurately predicting customer churn has become a critical factor for banks in their efforts to retain customers and enhance overall customer satisfaction. Understanding the underlying factors that contribute to customer churn can provide valuable insights that help banks develop targeted retention strategies, optimize the customer experience, and improve overall customer relationship management practices. By leveraging advanced analytics and data-driven approaches, banks can gain a deeper understanding of customer behaviour and preferences, enabling them to address customer churn more effectively.

2 OBJECTIVE:

The primary objective of this project is to develop a robust churn prediction model that can accurately identify potential churners among bank customers. Through the analysis of various customer-related data such as transaction history, demographics, account activity, and customer interactions, the model aims to predict which customers are most likely to churn in the near future. By utilizing sophisticated machine learning algorithms and predictive analytics techniques, the model can identify patterns and indicators that are indicative of potential churn, allowing banks to take proactive measures to retain these customers and minimize customer attrition. Ultimately, the development of an accurate churn prediction model can significantly contribute to the reduction of customer churn rates and improve overall customer retention for banks.

3 DATA SOURCE:

The dataset used for this project was obtained from Kaggle, specifically the "Bank Customer Churn" dataset, which can be accessed at (https://www.kaggle.com/datasets/radheshyamkollipara/bank-customer-churn). This dataset contains information on bank customers, including transaction history, demographics, account activity, customer interactions, and other relevant features. The dataset comprises 10,000 rows and 18 columns.

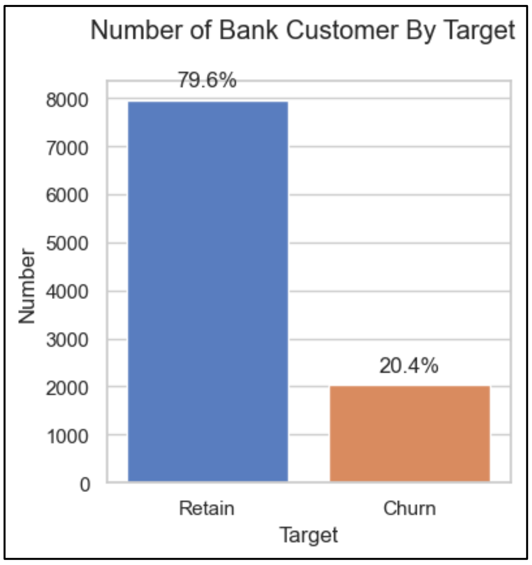
4 DATA PREPARATION:

Data preparation includes the following processes:

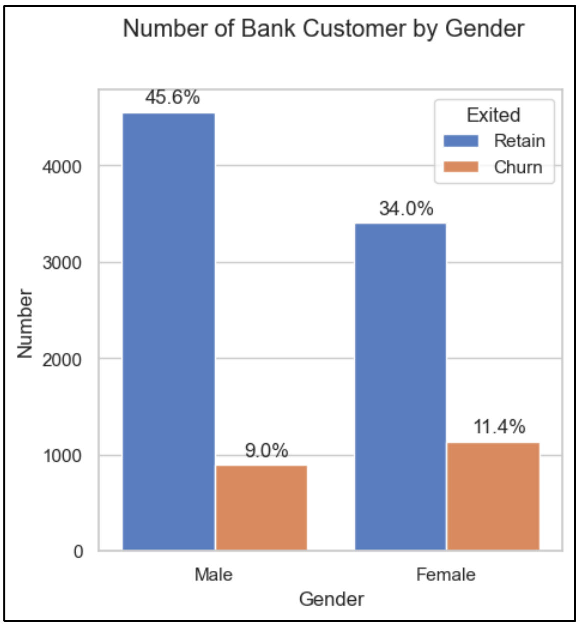
- Checking for data duplication. The result is that there are no duplicate data
- Checking for missing values. There are no missing values in this data
- Feature Engineering, extracting age group features, and transforming the data into the desired form.
- Encoding, converting categorical data into numerical. The encoding method used in this case is ordinal encoding.

5 EXPLORATORY DATA ANALYSIS (EDA):

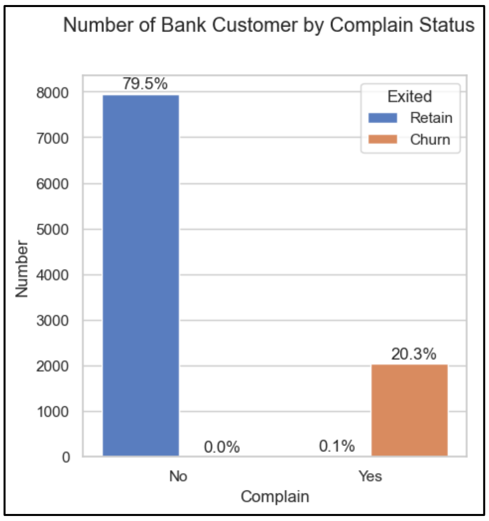
EDA is useful for understanding data deeply and gaining valuable knowledge through data visualization. The EDA in this project consists of univariate, bivariate, and correlation analyses. The following are some graphs from the EDA process:



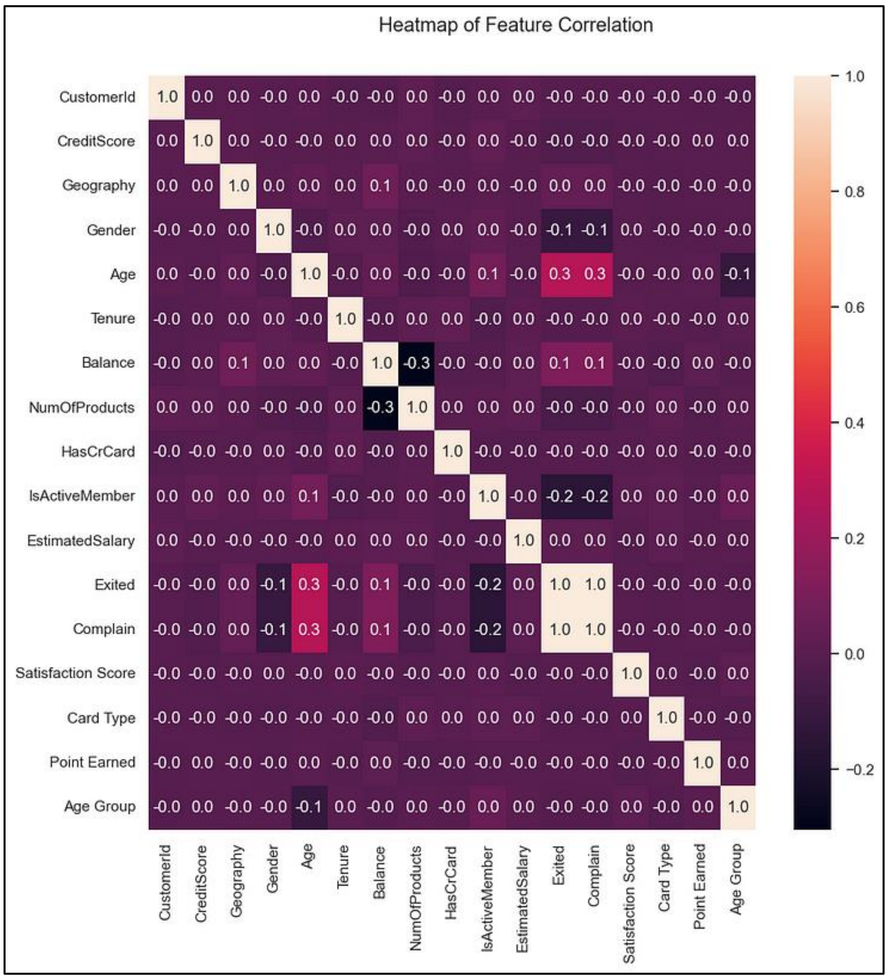
The graph above shows the distribution of the target variable, it is known that the number of Churn bank customers is 20.4%. It is concluded that there is an imbalance in the data on the target variable.



The graph above shows the distribution of the target variable by 'Gender'. Female bank customers churn the most with a percentage of 11.4% compared to males who have a percentage of 9%.



The graph above shows the distribution of ‘Complain’. All bank customers who churn are customers who also make complaints against the bank.

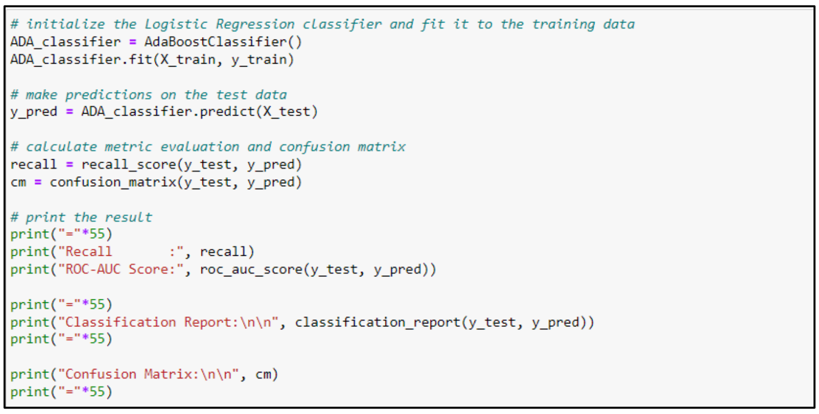


The graph above shows that there are not many features that are strongly correlated linearly with the target. This means that most of the correlations contained in the dataset are non-linear.

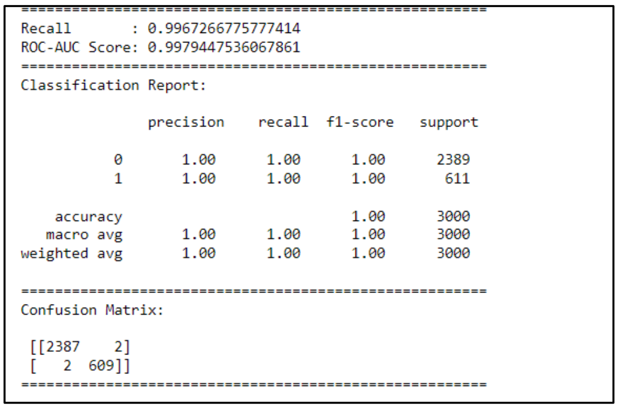
6 MODELLING:

The model experimentation process, in this case, will use the basic algorithms of Naive Bayes, Decision Tree, and AdaBoost. Based on prior knowledge of the correlation between features, the model developed is one that can work well on datasets that have many non-linear correlations between features such as Decision Tree and Gradient Boosting.

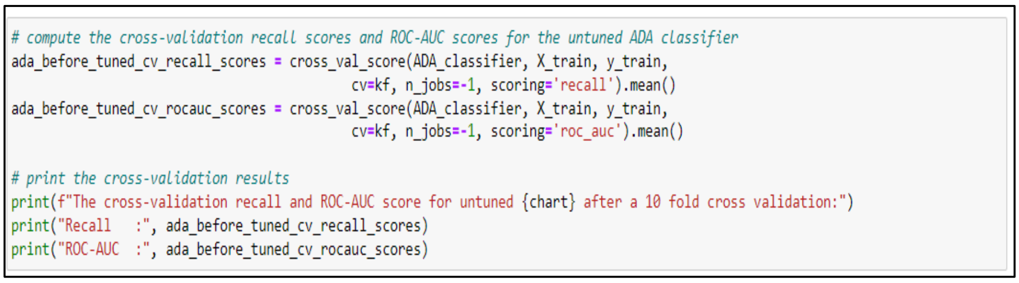
The following is an example of the modelling process for the best model, AdaBoost:



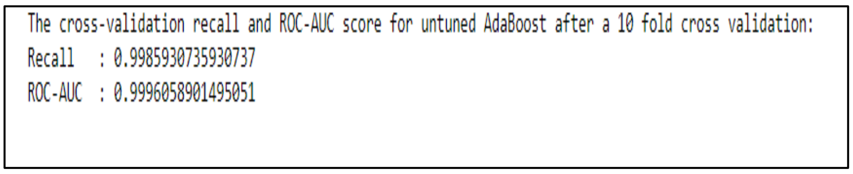
```python
# initialize the Logistic Regression classifier and fit it to the training data
ADA_classifier = AdaBoostClassifier()
ADA_classifier.fit(X_train, y_train)

# make predictions on the test data
y_pred = ADA_classifier.predict(X_test)

# calculate metric evaluation and confusion matrix
recall = recall_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# print the result
print("="*55)
print("Recall       :", recall)
print("ROC-AUC Score:", roc_auc_score(y_test, y_pred))

print("="*55)
print("Classification Report:\n\n", classification_report(y_test, y_pred))
print("="*55)

print("Confusion Matrix:\n\n", cm)
print("="*55)
```



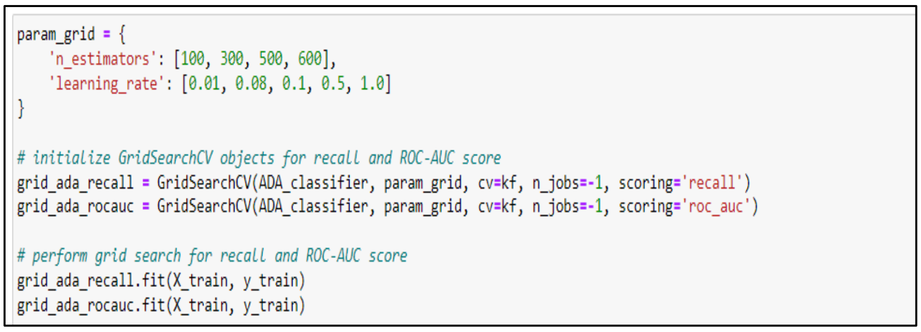
```
=======================================================
Recall       : 0.9967266775777414
ROC-AUC Score: 0.9979447536067861
=======================================================
Classification Report:

               precision    recall  f1-score   support

           0       1.00      1.00      1.00      2389
           1       1.00      1.00      1.00       611

    accuracy                           1.00      3000
   macro avg       1.00      1.00      1.00      3000
weighted avg       1.00      1.00      1.00      3000

=======================================================
Confusion Matrix:

 [[2387    2]
 [   2  609]]
=======================================================
```



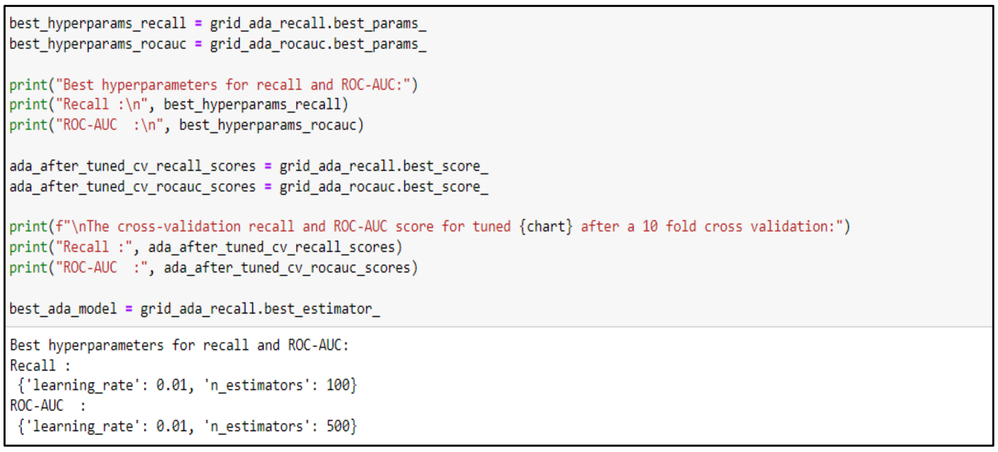
```python
# compute the cross-validation recall scores and ROC-AUC scores for the untuned ADA classifier
ada_before_tuned_cv_recall_scores = cross_val_score(ADA_classifier, X_train, y_train,
                                                    cv=kf, n_jobs=-1, scoring='recall').mean()
ada_before_tuned_cv_rocauc_scores = cross_val_score(ADA_classifier, X_train, y_train,
                                                    cv=kf, n_jobs=-1, scoring='roc_auc').mean()

# print the cross-validation results
print(f"The cross-validation recall and ROC-AUC score for untuned {chart} after a 10 fold cross validation:")
print("Recall   :", ada_before_tuned_cv_recall_scores)
print("ROC-AUC  :", ada_before_tuned_cv_rocauc_scores)
```



```
The cross-validation recall and ROC-AUC score for untuned AdaBoost after a 10 fold cross validation:
Recall   : 0.9985930735930737
ROC-AUC  : 0.9996058901495051
```

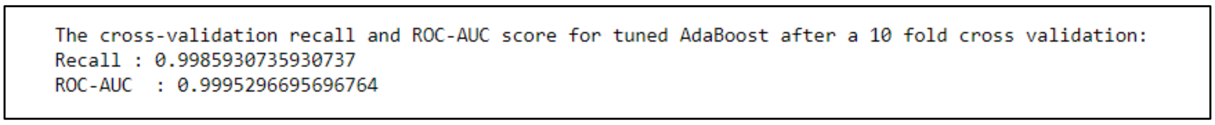


```python
param_grid = {
    'n_estimators': [100, 300, 500, 600],
    'learning_rate': [0.01, 0.08, 0.1, 0.5, 1.0]
}

# initialize GridSearchCV objects for recall and ROC-AUC score
grid_ada_recall = GridSearchCV(ADA_classifier, param_grid, cv=kf, n_jobs=-1, scoring='recall')
grid_ada_rocauc = GridSearchCV(ADA_classifier, param_grid, cv=kf, n_jobs=-1, scoring='roc_auc')

# perform grid search for recall and ROC-AUC score
grid_ada_recall.fit(X_train, y_train)
grid_ada_rocauc.fit(X_train, y_train)
```

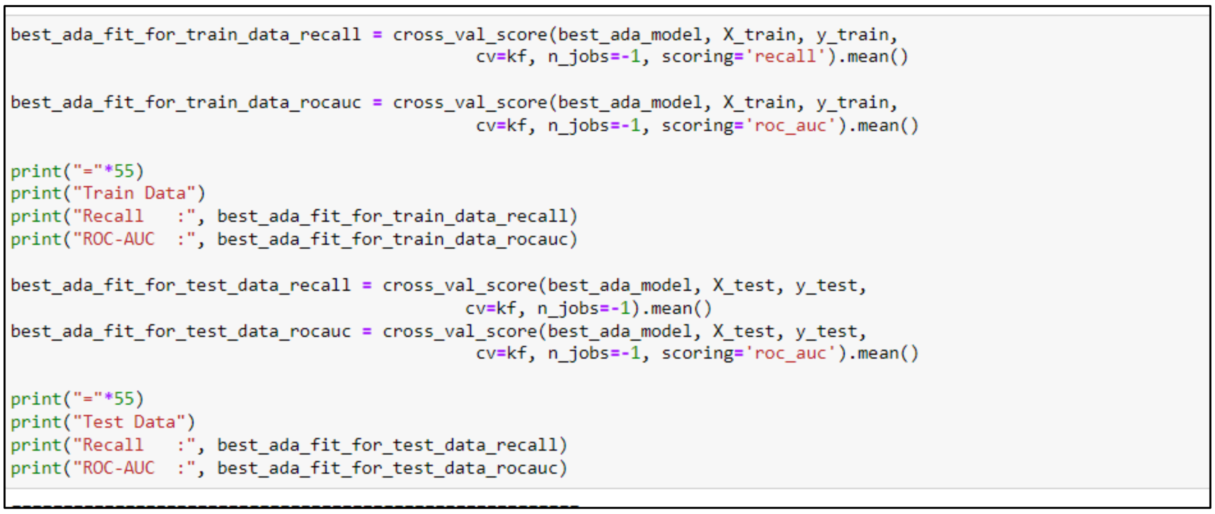


```python
best_hyperparams_recall = grid_ada_recall.best_params_
best_hyperparams_rocauc = grid_ada_rocauc.best_params_

print("Best hyperparameters for recall and ROC-AUC:")
print("Recall :\n", best_hyperparams_recall)
print("ROC-AUC  :\n", best_hyperparams_rocauc)

ada_after_tuned_cv_recall_scores = grid_ada_recall.best_score_
ada_after_tuned_cv_rocauc_scores = grid_ada_rocauc.best_score_

print(f"\nThe cross-validation recall and ROC-AUC score for tuned {chart} after a 10 fold cross validation:")
print("Recall :", ada_after_tuned_cv_recall_scores)
print("ROC-AUC  :", ada_after_tuned_cv_rocauc_scores)

best_ada_model = grid_ada_recall.best_estimator_
```

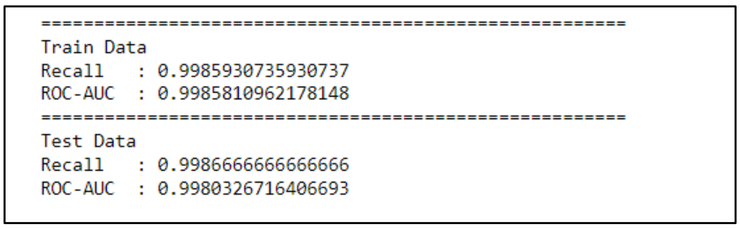
```
Best hyperparameters for recall and ROC-AUC:
Recall :
 {'learning_rate': 0.01, 'n_estimators': 100}
ROC-AUC  :
 {'learning_rate': 0.01, 'n_estimators': 500}
```



```
The cross-validation recall and ROC-AUC score for tuned AdaBoost after a 10 fold cross validation:
Recall : 0.9985930735930737
ROC-AUC  : 0.9995296695696764
```



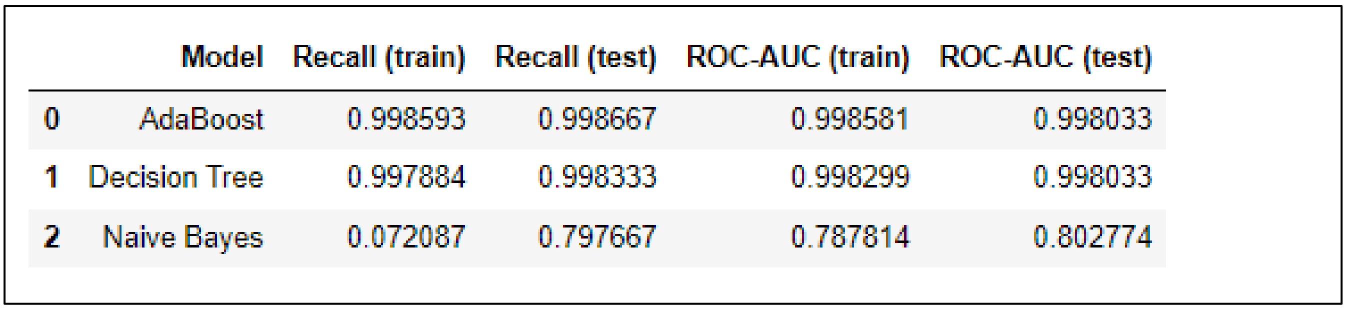
```python
best_ada_fit_for_train_data_recall = cross_val_score(best_ada_model, X_train, y_train,
                                          cv=kf, n_jobs=-1, scoring='recall').mean()

best_ada_fit_for_train_data_rocauc = cross_val_score(best_ada_model, X_train, y_train,
                                          cv=kf, n_jobs=-1, scoring='roc_auc').mean()

print("="*55)
print("Train Data")
print("Recall    :", best_ada_fit_for_train_data_recall)
print("ROC-AUC   :", best_ada_fit_for_train_data_rocauc)

best_ada_fit_for_test_data_recall = cross_val_score(best_ada_model, X_test, y_test,
                                         cv=kf, n_jobs=-1).mean()
best_ada_fit_for_test_data_rocauc = cross_val_score(best_ada_model, X_test, y_test,
                                          cv=kf, n_jobs=-1, scoring='roc_auc').mean()

print("="*55)
print("Test Data")
print("Recall    :", best_ada_fit_for_test_data_recall)
print("ROC-AUC   :", best_ada_fit_for_test_data_rocauc)
```



```
=======================================================
Train Data
Recall    : 0.9985930735930737
ROC-AUC   : 0.9985810962178148
=======================================================
Test Data
Recall    : 0.9986666666666666
ROC-AUC   : 0.9980326716406693
```

7 MODEL EVALUATION:

7.1 MODEL SELECTION



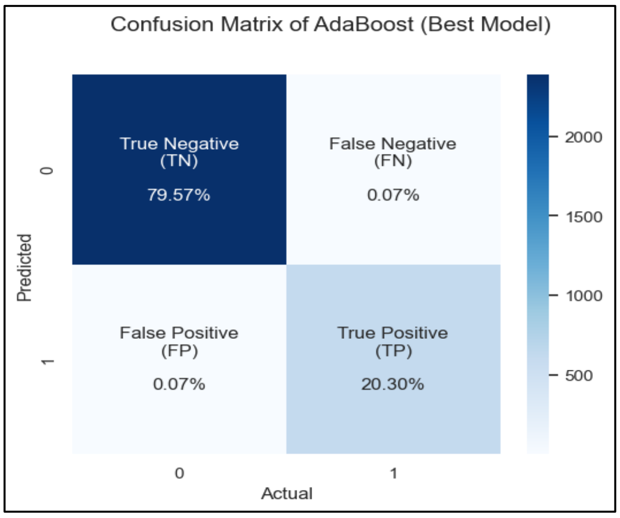
	Model	Recall (train)	Recall (test)	ROC-AUC (train)	ROC-AUC (test)
0	AdaBoost	0.998593	0.998667	0.998581	0.998033
1	Decision Tree	0.997884	0.998333	0.998299	0.998033
2	Naive Bayes	0.072087	0.797667	0.787814	0.802774

In churn analysis, the main goal is to identify customers who are likely to churn so that appropriate actions can be taken to retain them. The best model is the one that can generate the most correct predictions. However, in this business context, the percentage of False Negative (FN) is important. A False Negative predicts a customer will not churn when they actually do. False negatives are critical because they represent a missed opportunity to intervene and retain customers who are at risk of churning. Based on that, the percentage of False Negative should be minimized. In this regard, the most suitable evaluation metric is recall because it takes into account False Negative.

Prioritizing recall ensures that the model can avoid mistakenly predicting no churn that is actually churn, thereby increasing the chances of taking proactive action to retain those customers and having a good business impact. In addition, the ROC-AUC score provides a comprehensive measure of the model's performance across various classification thresholds and demonstrates the model's ability to correctly rank classes. This approach allows for a more balanced evaluation and selection of models based on their ability to accurately predict the target variable.

The best model is selected from the model that produces the highest average recall value for the test data and does not indicate overfitting and underfitting. Based on the table above, the selected model is **AdaBoost** with an average recall for the testing data of **0.998667**.

7.2 Confusion Matrix:

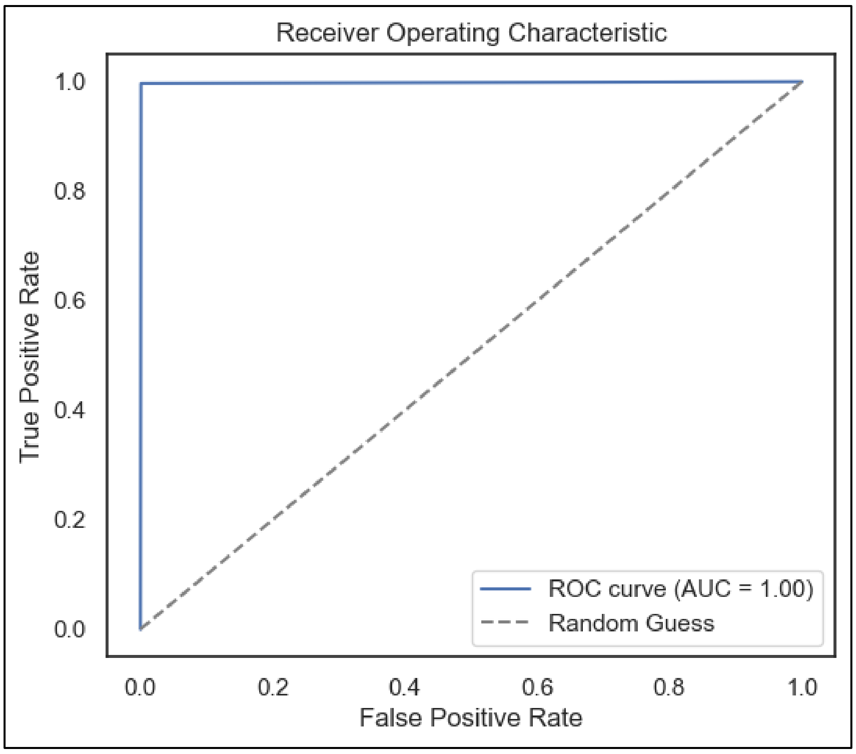


Class 0: Retain
Class 1: Churn

Confusion matrix of the best model after the tuning process. Resulting in True Negative (TN) 79.57%, True Positive (TP) 20.30%, False Negative (FN) 0.07%, False Positive (FP) 0.07% which means the model is good enough in predicting churn.

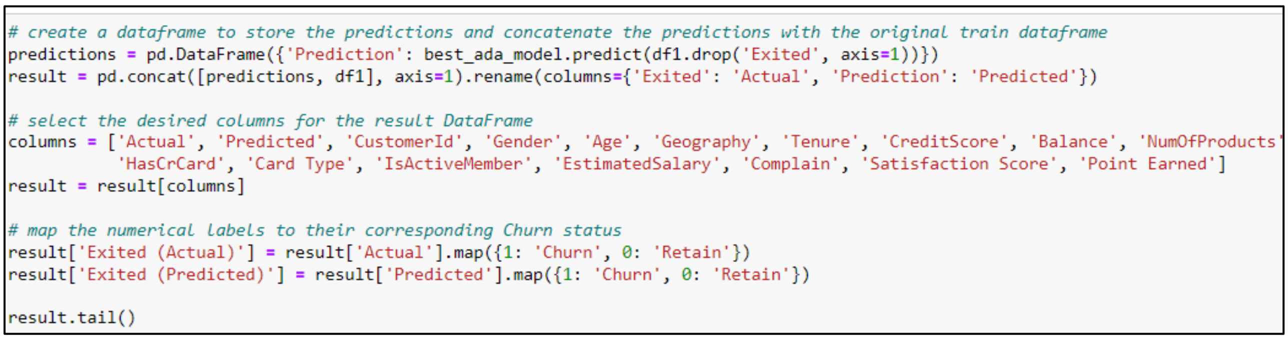
7.3 ROC AUC Curve:

```
from sklearn.metrics import roc_curve, auc

y_prob = best_ada_model.predict_proba(X_test)[:, 1]

fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label='ROC curve (AUC = {:.2f})'.format(roc_auc))
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.show()
```

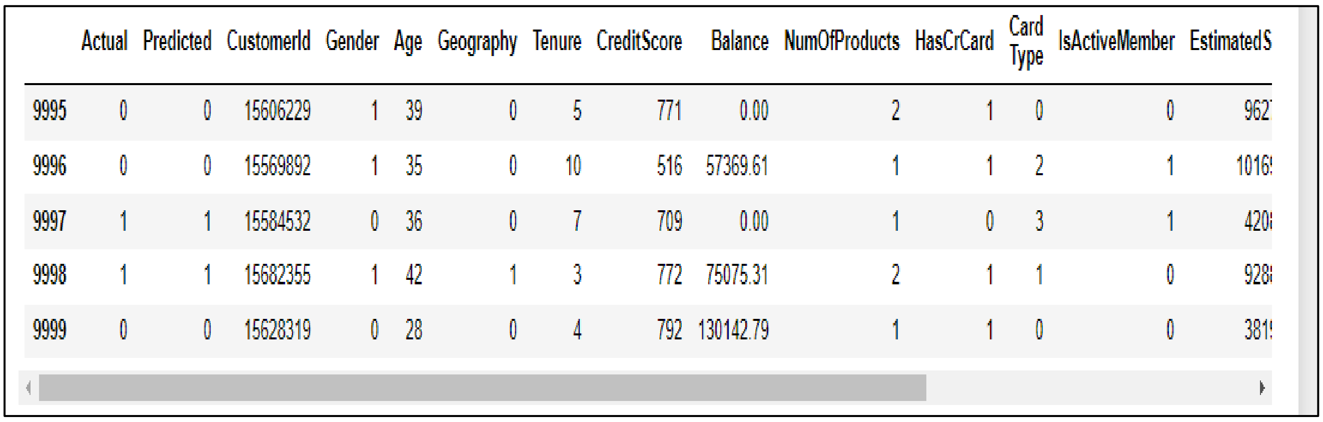


The graph shows the ROC AUC curve with an almost perfect score of 0.999998. This means that the model works well in predicting each class.

8 PREDICTION:



```
# create a dataframe to store the predictions and concatenate the predictions with the original train dataframe
predictions = pd.DataFrame({'Prediction': best_ada_model.predict(df1.drop('Exited', axis=1))})
result = pd.concat([predictions, df1], axis=1).rename(columns={'Exited': 'Actual', 'Prediction': 'Predicted'})

# select the desired columns for the result DataFrame
columns = ['Actual', 'Predicted', 'CustomerId', 'Gender', 'Age', 'Geography', 'Tenure', 'CreditScore', 'Balance', 'NumOfProducts'
           'HasCrCard', 'Card Type', 'IsActiveMember', 'EstimatedSalary', 'Complain', 'Satisfaction Score', 'Point Earned']
result = result[columns]

# map the numerical labels to their corresponding Churn status
result['Exited (Actual)'] = result['Actual'].map({1: 'Churn', 0: 'Retain'})
result['Exited (Predicted)'] = result['Predicted'].map({1: 'Churn', 0: 'Retain'})

result.tail()
```



	Actual	Predicted	CustomerId	Gender	Age	Geography	Tenure	CreditScore	Balance	NumOfProducts	HasCrCard	Card Type	IsActiveMember	EstimatedS
9995	0	0	15606229	1	39	0	5	771	0.00	2	1	0	0	962
9996	0	0	15569892	1	35	0	10	516	57369.61	1	1	2	1	1016
9997	1	1	15584532	0	36	0	7	709	0.00	1	0	3	1	420
9998	1	1	15682355	1	42	1	3	772	75075.31	2	1	1	0	928
9999	0	0	15628319	0	28	0	4	792	130142.79	1	1	0	0	381

9 RESULT:

Based on the whole process from data preparation, analysis of data exploration results, and modeling, the following are the results:

The most suitable model evaluation metrics, in this case, are recall and ROC-AUC, using recall because it needs to take into account the percentage of False Negative (FN) that must be minimized. While ROC-AUC is to measure how well (balanced) the model predicts each class.

The best model obtained is AdaBoost which produces recall and ROC-AUC scores of 0.998667 and 0.998033 in test data, respectively.