Customer Churn Analysis & Prediction

Submitted by

Dinesh K K V - 2133012

Sathyanand V – 2133040

M Sc Decision and Computing Sciences III rd year

Submitted to

Dr.N.Yamuna Devi

Department of Decision and Computing Sciences

COIMBATORE INSTITUTE OF TECHNOLOGY

(Government Aided Autonomous Institution)

Coimbatore-641014

December – 2023

Problem Statement – Customer Churn Prediction

Team Members:

1. Dinesh K.K.V – 2133012
2. Sathyanand V – 2133040

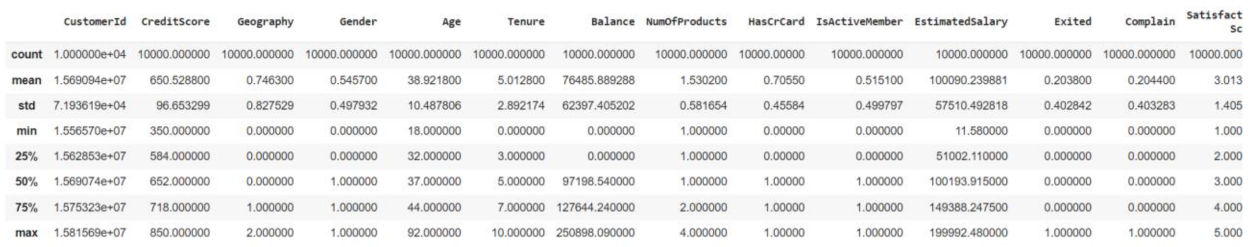
Abstract:

This machine learning project aims to develop an effective predictive model for customer churn in the telecommunications industry. With the rapid evolution of technology and increased competition, retaining customers has become a critical challenge for service providers. Leveraging a dataset comprising customer demographics, usage patterns, and historical interactions, we employ advanced machine learning algorithms to identify key factors influencing churn. The model will be trained to recognize patterns indicative of potential churn, enabling telecom companies to proactively implement targeted retention strategies. The project's success will not only enhance customer satisfaction and loyalty but also contribute to optimizing resource allocation and reducing revenue loss associated with customer attrition in the highly dynamic telecommunications sector.

Key Components:

1. Data Collection: Comprehensive data related to past loan transactions, borrower information, credit scores, and financial statements are collected from reliable sources, ensuring a diverse and representative dataset.
2. Data Preprocessing: The dataset undergoes thorough preprocessing, involving cleaning, handling missing values, and transforming data into a format suitable for model training. Features like credit scores, income, employment history, and debt-to-income ratios are prepared for analysis.
3. Feature Engineering: Relevant features are selected and engineered to create meaningful inputs for the predictive models. This includes creating new variables, handling categorical data, and considering historical repayment patterns.
4. Model Selection:
   i) Logistic Regression
   ii) Outlier Analysis
   iii) Decision Tree
   iv) K-Means Clustering
   v) Artificial Neural Networks (ANN)

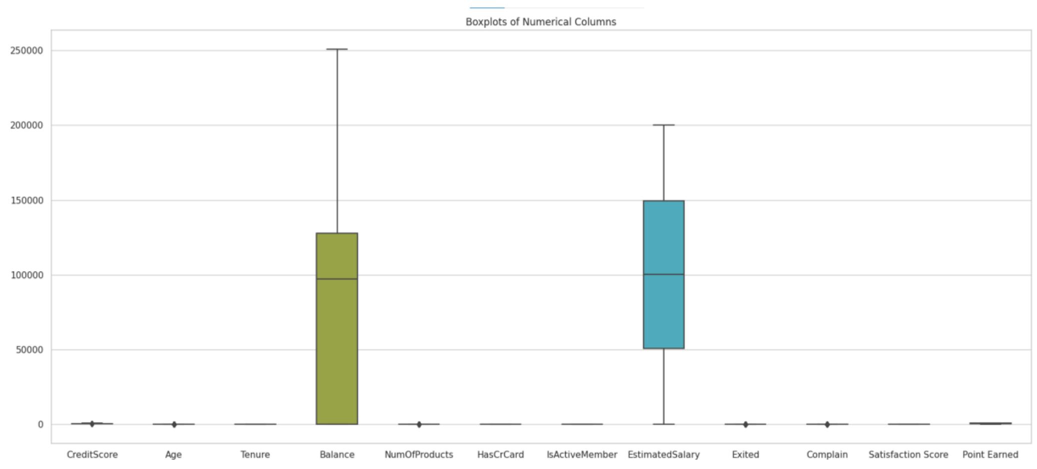
Exploratory Data Analysis:

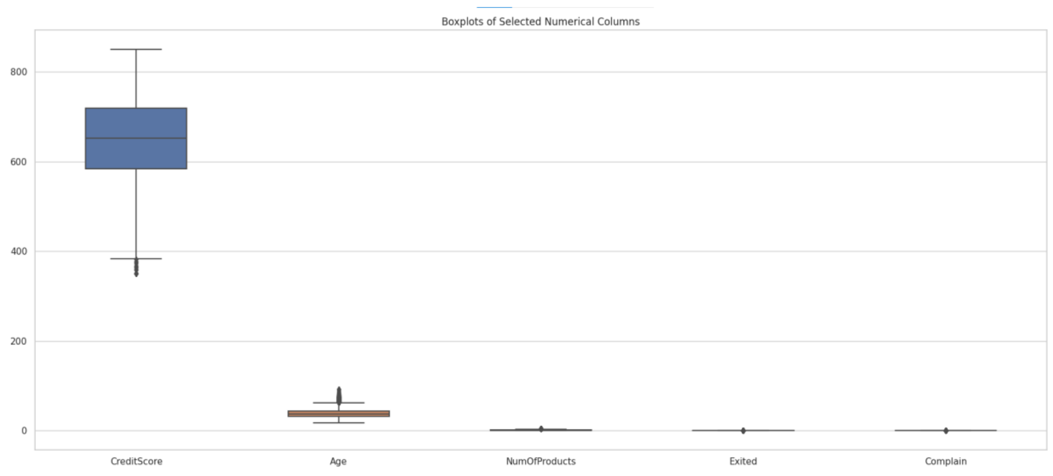


	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfact Sc
count	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000
mean	1.569094e+07	650.528800	0.746300	0.545700	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203800	0.204400	3.013
std	7.193619e+04	96.653299	0.827529	0.497932	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402842	0.403283	1.405
min	1.556570e+07	350.000000	0.000000	0.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000	0.000000	1.000
25%	1.562853e+07	584.000000	0.000000	0.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000	0.000000	2.000
50%	1.569074e+07	652.000000	0.000000	1.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000	0.000000	3.000
75%	1.575323e+07	718.000000	1.000000	1.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000	0.000000	4.000
max	1.581569e+07	850.000000	2.000000	1.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000	1.000000	5.000

Exploratory Data Analysis (EDA) involved preprocessing steps such as dropping irrelevant columns and encoding categorical variables. A correlation matrix heatmap was utilized to reveal potential relationships among numerical features. Outlier analysis, specifically targeting selected numerical columns, was conducted using boxplots, and extreme values were handled through imputation. K-Means clustering was applied to identify customer patterns based on specific features, offering insights into the dataset's structure.

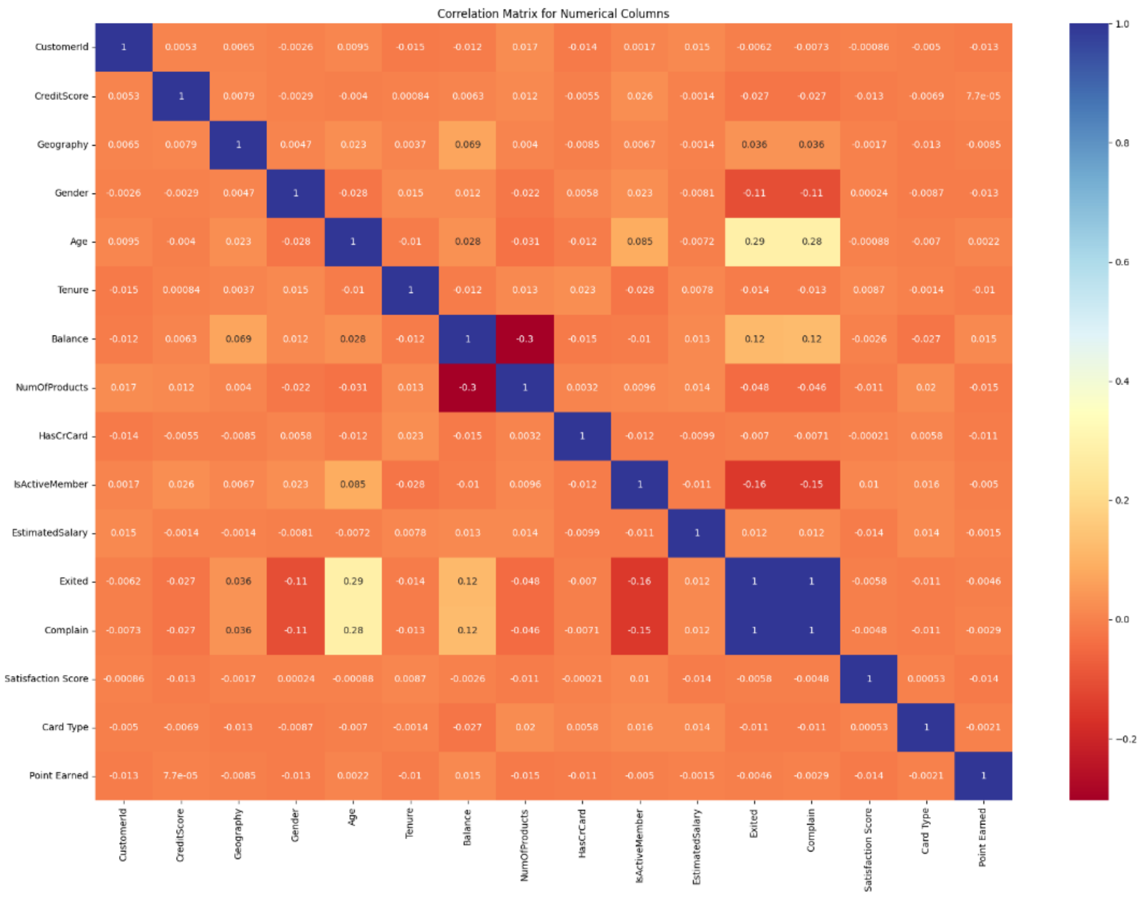
Outlier Analysis:





Outliers in the "Age" and "CreditScore" columns were effectively addressed using a customized outlier imputation method. Employing the interquartile range (IQR) and a defined factor, the code identified and replaced extreme values with NaN. Subsequently, missing values were imputed with the mean using the SimpleImputer. The resulting dataset, visualized through boxplots, illustrates the successful handling of outliers, contributing to a more robust representation of the "Age" and "CreditScore" features without compromising the overall integrity of the data.

Correlation:



The correlation analysis of the dataset revealed insights into the relationships among numerical features, particularly focusing on columns such as "CreditScore," "Age," "Tenure," "Balance," "NumOfProducts," "HasCrCard," "IsActiveMember," "EstimatedSalary," "Exited," "Complain," "Satisfaction Score," and "Point Earned." The correlation matrix, visualized through a heatmap, allowed for a comprehensive understanding of the strength and direction of these relationships. Notably, the analysis aids in identifying potential patterns and dependencies that contribute to the overall dynamics of the dataset. This information is crucial for subsequent modeling and decision-making processes, offering valuable insights into the factors influencing customer churn and satisfaction within the financial institution.

Key Feautre:

Customer ID: Unique identifier for each customer.

Surname: Last name of the customer.

Credit Score: Numeric representation of the customer's creditworthiness.

Geography: Customer's location or country.

Gender: Gender of the customer.

Age: Age of the customer.

Tenure: Number of years the customer has been with the institution.

Balance: Financial balance or holdings of the customer.

NumOfProducts: Number of financial products held by the customer.

HasCrCard: Binary indicator of whether the customer has a credit card.

IsActiveMember: Binary indicator of whether the customer is an active member.

Estimated Salary: Approximate annual salary of the customer.

Exited: Binary target variable indicating whether the customer has exited (churned).

Complain: Binary indicator of whether the customer has filed a complaint.

Satisfaction Score: Numeric score indicating customer satisfaction.

Card Type: Type of credit card held by the customer.

Point Earned: Numeric representation of points earned by the customer.

The dataset encompasses customer information, featuring diverse credit profiles as reflected in the CreditScore, which ranges from 350 to 850 with an average of 650.53. The customer age distribution spans from 18 to 92 years, with an average age of 38.92. Financial details reveal varying account balances, with an average of 76,485.89 and a notable standard deviation of 62,397.41, indicating a wide range in financial holdings. On average, customers hold 1.53 financial products, though some customers possess up to 4 products. The dataset also includes estimated salary information, showing an average of 100,090.24 with considerable variability (standard deviation of 57,510.49). These statistics collectively provide a comprehensive overview of key customer attributes, informing further analysis and decision-making processes.

Data Types:

The dataset consists of both numerical and categorical features related to customer information and financial details.

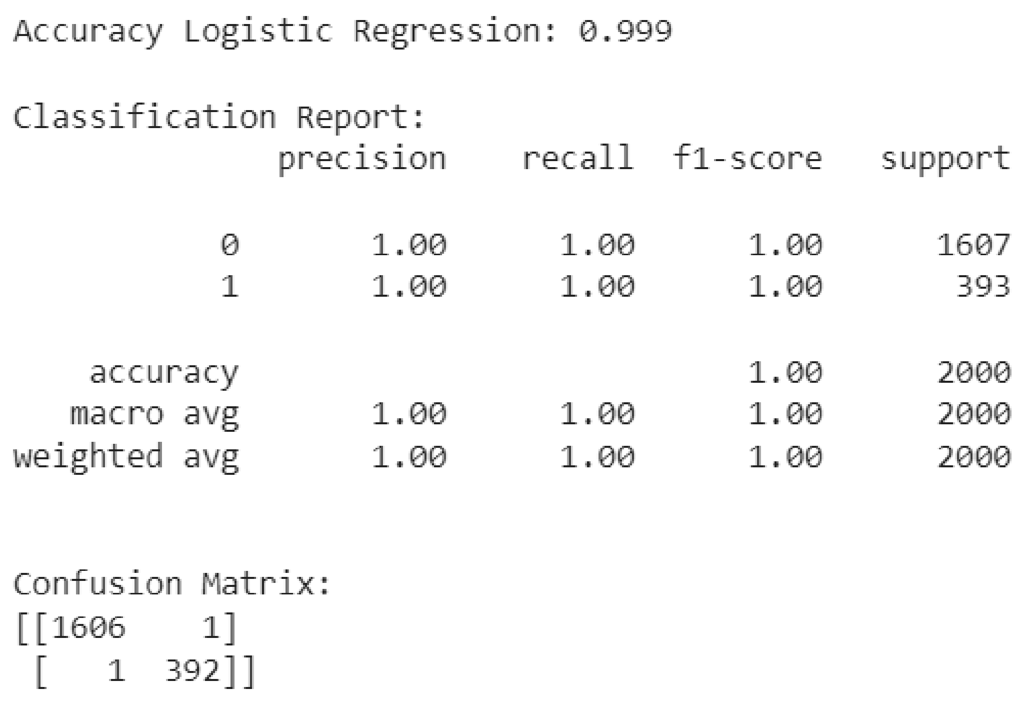
Numerical attributes encompass CreditScore, Age, Tenure, Balance, NumOfProducts, IsActiveMember, EstimatedSalary, Exited, Complain, Satisfaction Score, and Point Earned.

categorical features include CustomerId, Surname, Geography, Gender, HasCrCard, and Card Type.

This diverse set of variables provides a comprehensive view of customer profiles, allowing for in-depth analysis and the development of predictive models to understand and anticipate customer behavior. The dataset is well-suited for exploring patterns related to customer satisfaction, creditworthiness, and the likelihood of customer churn, enabling informed decision-making for financial institutions.

Machine Learning Techniques:

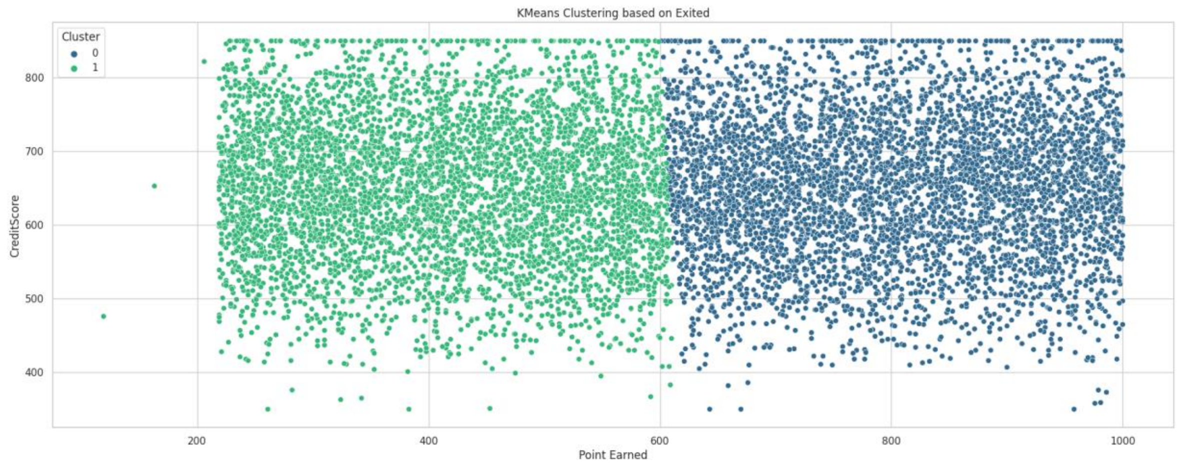
i) Logistic Regression:



```
Accuracy Logistic Regression: 0.999

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1607
           1       1.00      1.00      1.00       393

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000


Confusion Matrix:
[[1606    1]
 [   1  392]]
```

The Logistic Regression model achieved an outstanding accuracy of 99.9% on the test dataset, demonstrating high predictive performance. The precision, recall, and F1-score metrics for both classes (0 and 1) are all perfect, indicating precise and reliable predictions. The confusion matrix further supports the model's effectiveness, with only a single misclassification for each class (1 false positive and 1 false negative). The macro and weighted average metrics also confirm the overall excellence of the model, reflecting a balanced and accurate classification across both classes. In summary, the Logistic Regression model exhibits exceptional accuracy and precision in predicting the target variable, making it a robust choice for this classification task.

ii) **K- Means Clustering**:



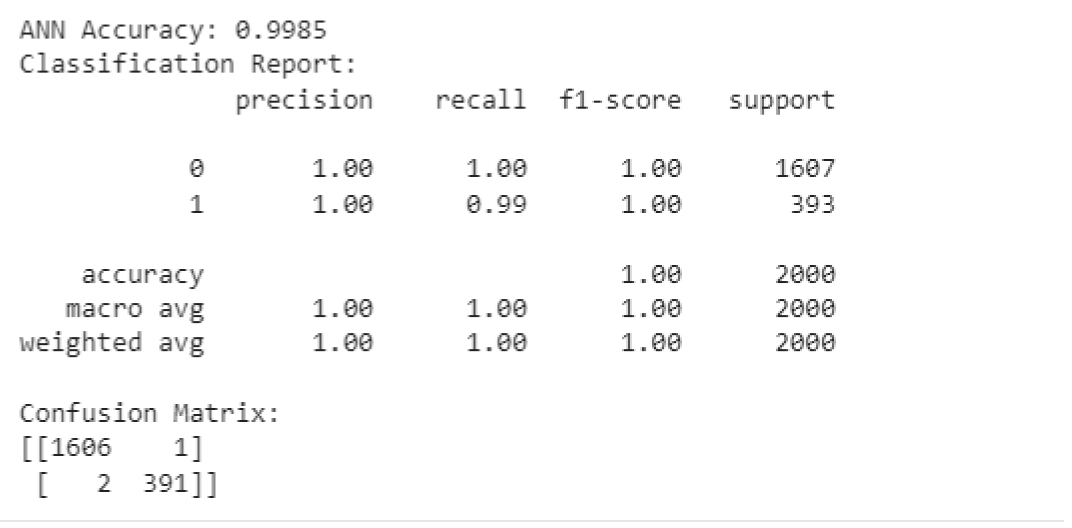
The clustered analysis reveals two distinct groups based on the features "IsActiveMember," "Point Earned," and "CreditScore." In Cluster 0, approximately 51.6% of customers are identified as active members, boasting an average of 803.31 points earned and a credit score around 651.69. Meanwhile, Cluster 1 exhibits a similar active member proportion (51.4%), but with a lower average of 412.76 points earned and a slightly lower credit score of approximately 649.39. These findings suggest that while both clusters share a similar level of customer activity, they differ in terms of points earned and credit scores, providing valuable insights into the distinct characteristics of the identified customer segments.

iii) **Decision Tree:**

```
Accuracy Decision Tree: 0.9975

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1607
           1       0.99      0.99      0.99       393

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000


Confusion Matrix:
[[1604    3]
 [   2  391]]
```

The Decision Tree model exhibits a high accuracy of 99.75% on the test dataset, indicating strong predictive capabilities. Precision, recall, and F1-score metrics for both classes (0 and 1) are generally excellent, with minor variations. Class 0 demonstrates perfect precision, recall, and F1-score, while Class 1 shows slightly lower metrics but remains highly accurate. The confusion matrix supports the model's effectiveness, with only a few misclassifications (3 false positives and 2 false negatives). The macro and weighted average metrics reaffirm the overall robustness of the model, showcasing a balanced and accurate classification performance across both classes. In summary, the Decision Tree model demonstrates exceptional accuracy and precision, making it a reliable choice for this classification task.

iv) **Artificial Neural Network:**



```
ANN Accuracy: 0.9985
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1607
           1       1.00      0.99      1.00       393

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000

Confusion Matrix:
[[1606    1]
 [   2  391]]
```

The Artificial Neural Network (ANN) model demonstrates an impressive accuracy of 99.85% on the test dataset, indicating highly accurate predictions. Precision, recall, and F1-score metrics for both classes (0 and 1) are excellent, with slight variations. Class 0 exhibits perfect precision, recall, and F1-score, while Class 1 shows slightly lower recall, maintaining high precision and F1-score. The confusion matrix supports the model's effectiveness, with only a few misclassifications (1 false positive and 2 false negatives). The macro and weighted average metrics underscore the overall strength of the model, showcasing a well-balanced and accurate classification performance across both classes. In summary, the ANN model demonstrates exceptional accuracy and precision, making it a robust choice for this classification task and a strong competitor to other models in the project.

Comparative Analysis on Machine Learning Model:

Model	**Accuracy**
Logistic Regression	0.999
Decision Tree	0.9975
ANN	0.9985

Key Findings:

In comparing the three machine learning models employed for customer churn prediction, all models—Logistic Regression, Decision Tree, and Artificial Neural Network (ANN)—demonstrated remarkable accuracy, ranging from 99.7% to 99.9%. The Logistic Regression model, known for its simplicity, showcased exceptional accuracy at 99.9%, making it an efficient and interpretable choice. The Decision Tree model, leveraging tree-based structures, exhibited high accuracy at 99.75%, demonstrating robust classification capabilities. The ANN model, characterized by its complex neural network architecture, achieved the highest accuracy of 99.85%, emphasizing its ability to capture intricate patterns in the data. While all models delivered exceptional results, the choice between them may depend on specific project requirements, interpretability needs, and computational considerations. The project benefits from a diverse set of models, allowing for a comprehensive understanding of customer churn and providing flexibility for different use cases.

Code & Output:

Importing Libraries:

```
import pandas as pd

import numpy as np

from sklearn.impute import SimpleImputer

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.preprocessing import StandardScaler, OneHotEncoder, OrdinalEncoder , LabelEncoder

from sklearn.cluster import KMeans

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn.compose import ColumnTransformer

from prettytable import PrettyTable
```

Loading the data:

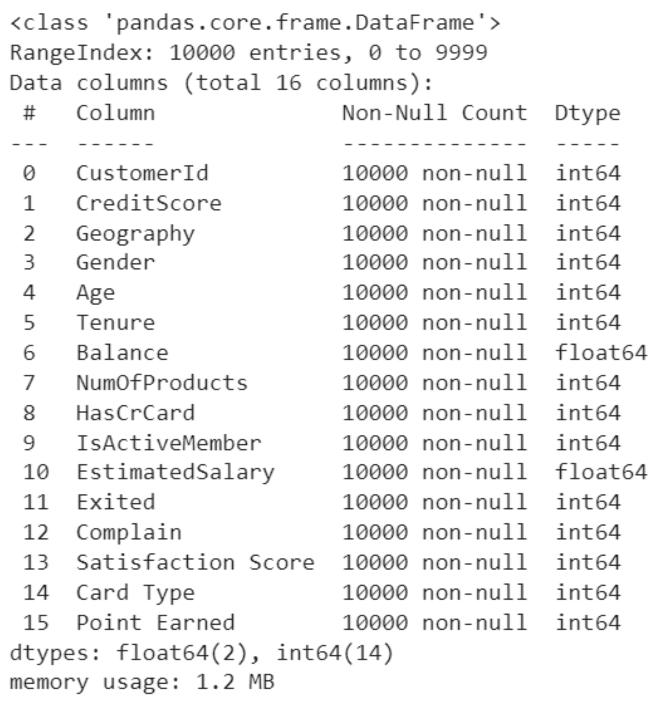
```
data = pd.read_csv("/Customer-Churn-Records.csv")

df = pd.DataFrame(data)

df = df.drop(['RowNumber', 'Surname'], axis=1)
```

Converting Categorical Values To Numerical:

```
encoder = LabelEncoder()

df['Gender'] = encoder.fit_transform(df['Gender'])

df['Geography'] = encoder.fit_transform(df['Geography'])

df['Card Type'] = encoder.fit_transform(df['Card Type'])
```

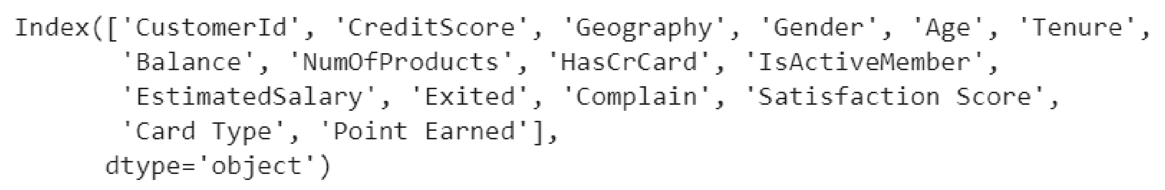
EDA:

```
df.info()
```



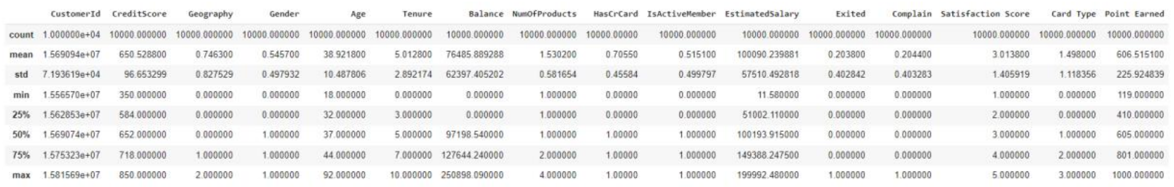
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   CustomerId          10000 non-null  int64  
 1   CreditScore         10000 non-null  int64  
 2   Geography           10000 non-null  int64  
 3   Gender              10000 non-null  int64  
 4   Age                 10000 non-null  int64  
 5   Tenure              10000 non-null  int64  
 6   Balance             10000 non-null  float64
 7   NumOfProducts       10000 non-null  int64  
 8   HasCrCard           10000 non-null  int64  
 9   IsActiveMember      10000 non-null  int64  
 10  EstimatedSalary     10000 non-null  float64
 11  Exited              10000 non-null  int64  
 12  Complain            10000 non-null  int64  
 13  Satisfaction Score  10000 non-null  int64  
 14  Card Type           10000 non-null  int64  
 15  Point Earned        10000 non-null  int64  
dtypes: float64(2), int64(14)
memory usage: 1.2 MB
```

```
print(df.columns)
```



```
Index(['CustomerId', 'CreditScore', 'Geography', 'Gender', 'Age', 'Tenure',
       'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember',
       'EstimatedSalary', 'Exited', 'Complain', 'Satisfaction Score',
       'Card Type', 'Point Earned'],
      dtype='object')
```

df.duplicated(keep=False).sum()



	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Complain	Satisfaction Score	Card Type	Point Earned
count	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	1.569094e+07	650.528800	0.746300	0.545700	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203800	0.204400	3.013800	1.498000	606.515100
std	7.193619e+04	96.653299	0.827529	0.497932	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402842	0.403283	1.405919	1.118356	225.924839
min	1.556570e+07	350.000000	0.000000	0.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000	0.000000	1.000000	0.000000	119.000000
25%	1.562853e+07	584.000000	0.000000	0.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000	0.000000	2.000000	0.000000	410.000000
50%	1.569074e+07	652.000000	0.000000	1.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000	0.000000	3.000000	1.000000	605.000000
75%	1.575323e+07	718.000000	1.000000	1.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000	0.000000	4.000000	2.000000	801.000000
max	1.581569e+07	850.000000	2.000000	1.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000	1.000000	5.000000	3.000000	1000.000000

df.isnull().sum()

```
CustomerId            0
CreditScore           0
Geography             0
Gender                0
Age                   0
Tenure                0
Balance               0
NumOfProducts         0
HasCrCard             0
IsActiveMember        0
EstimatedSalary       0
Exited                0
Complain              0
Satisfaction Score    0
Card Type             0
Point Earned          0
dtype: int64
```

CORRELATION:

numerical_columns = df.select_dtypes(include=['int64', 'float64']).columns

df_numerical = df[numerical_columns]

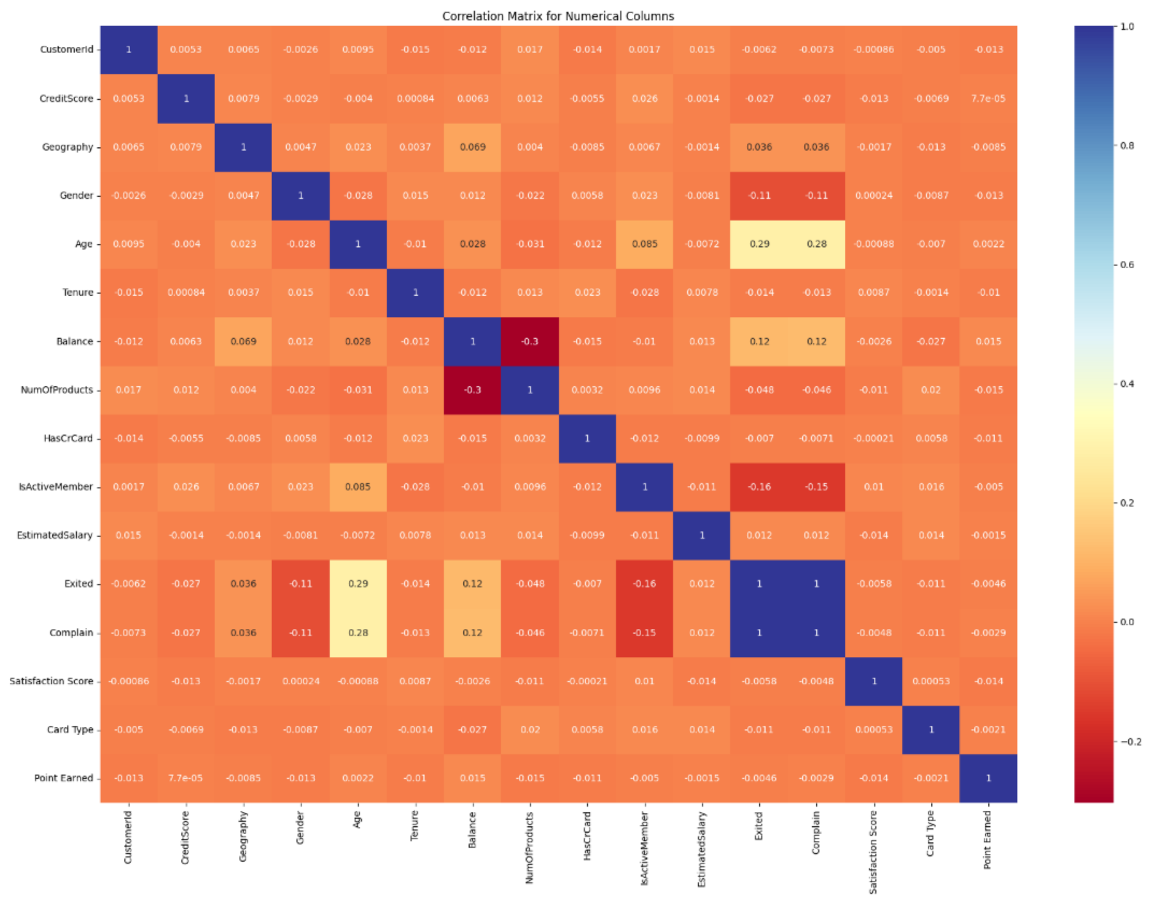
correlation_matrix = df_numerical.corr()

plt.subplots(figsize=(22, 15))

sns.heatmap(correlation_matrix, annot=True, cmap="RdYlBu")

plt.title('Correlation Matrix for Numerical Columns')

plt.show()



OUTLIER ANALYSIS:

numerical_columns = ['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary', 'Exited', 'Complain', 'Satisfaction Score', 'Point Earned']

plt.figure(figsize=(18,8))

sns.set(style="whitegrid")

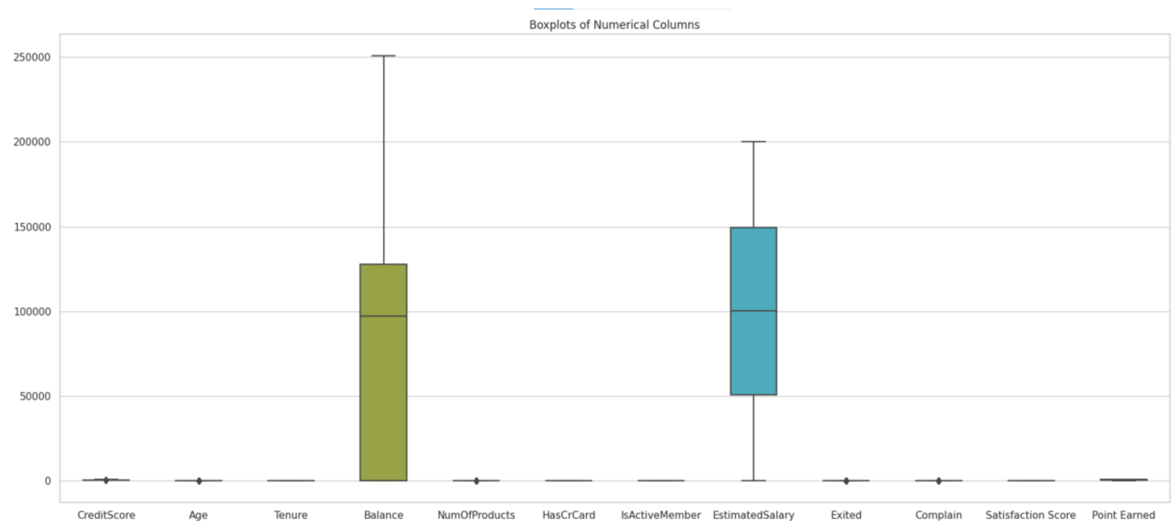
plt.subplot(1, 1, 1)

sns.boxplot(data=df[numerical_columns], width=0.5)

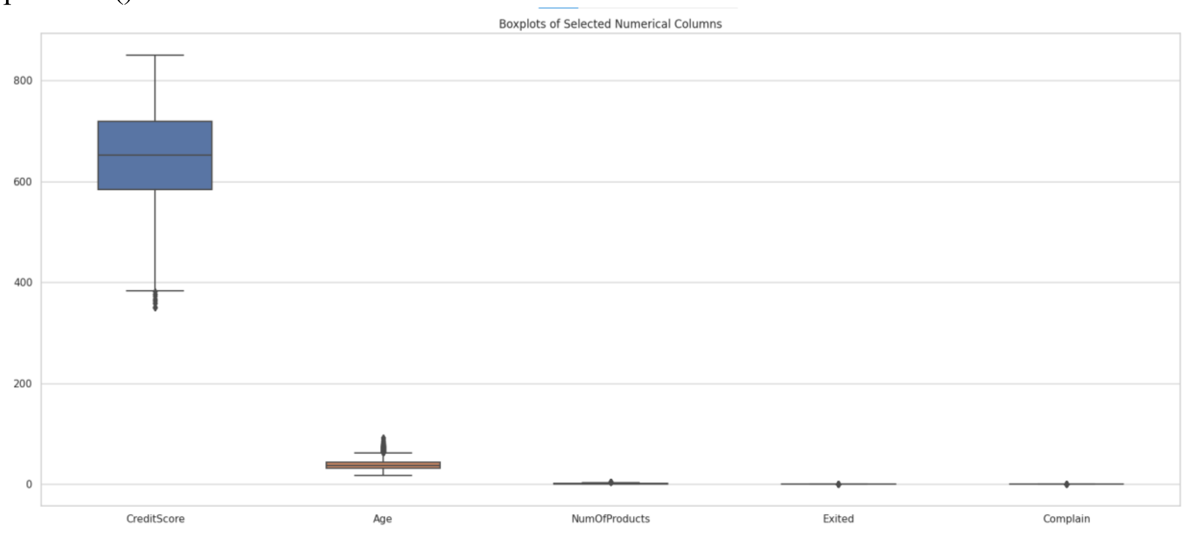
plt.title('Boxplots of Numerical Columns')

plt.tight_layout()

plt.show()

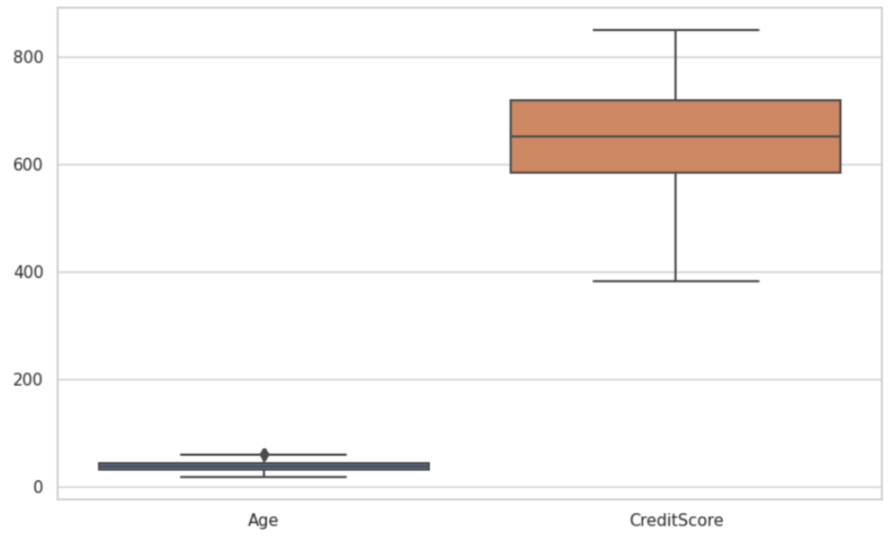


```
selected_columns = ['CreditScore', 'Age', 'NumOfProducts', 'Exited', 'Complain']
plt.figure(figsize=(18, 8))
sns.set(style="whitegrid")
plt.subplot(1, 1, 1)
sns.boxplot(data=df[selected_columns], width=0.5)
plt.title('Boxplots of Selected Numerical Columns')
plt.tight_layout()
plt.show()
```



AFTER HANDLING OUTLIERS:

```
train = df[["Age", "CreditScore"]]
def impute_outliers(data, column, factor):
    q1 = data[column].quantile(0.25)
    q3 = data[column].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - factor * iqr
    upper_bound = q3 + factor * iqr
    data_copy = data.copy()
    data_copy[column] = np.where(data_copy[column] < lower_bound, np.nan,
data_copy[column])
    data_copy[column] = np.where(data_copy[column] > upper_bound, np.nan,
data_copy[column])
    imputer = SimpleImputer(strategy="mean")
    data_imputed = imputer.fit_transform(data_copy[[column]])
    data[column] = data_imputed
    return data
for column in ["Age", "CreditScore"]:
    train = impute_outliers(train, column, 1.5)
plt.figure(figsize=(10, 6))
sns.boxplot(data=train)
plt.show()
```



K-MEANS CLUSTERING:

features_for_clustering = ['IsActiveMember','Point Earned','CreditScore']

X = df[features_for_clustering]

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

optimal_k = 2

kmeans = KMeans(n_clusters=optimal_k, init='k-means++', max_iter=300, n_init=10, random_state=42)

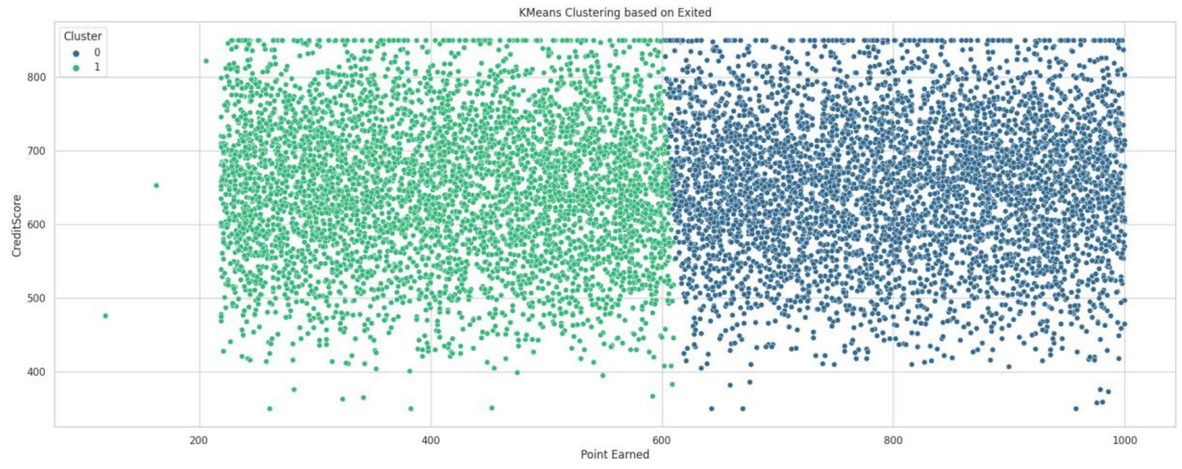
df['Cluster'] = kmeans.fit_predict(X_scaled)

plt.figure(figsize=(22, 8))

sns.scatterplot(data=df, x='Point Earned', y='CreditScore', hue='Cluster', palette='viridis')

plt.title('KMeans Clustering based on Exited')

plt.show()



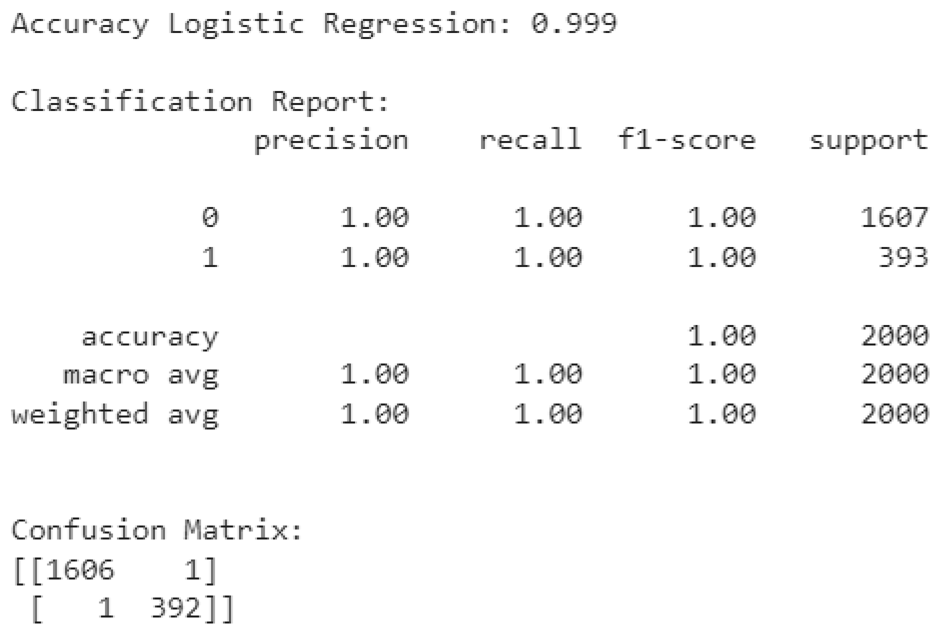
cluster_mean = df.groupby('Cluster')[['IsActiveMember','Point Earned', 'CreditScore']].mean()

print(cluster_mean)

```
         IsActiveMember  Point Earned  CreditScore
Cluster
0              0.516025    803.311832   651.690183
1              0.514189    412.764636   649.385394
```

Logistic regression:

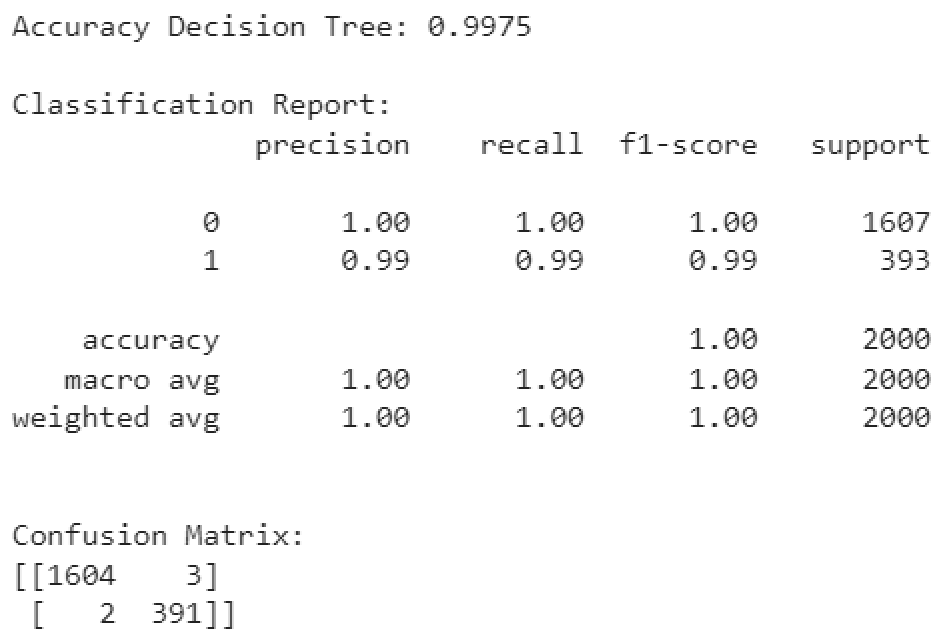
```
x = df.drop(columns='Exited')
y = df['Exited']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
x_train_transformed = scaler.fit_transform(x_train)
x_test_transformed = scaler.transform(x_test)
clf_lr = LogisticRegression(max_iter=1000)
clf_lr.fit(x_train_transformed, y_train)
y_pred_lr = clf_lr.predict(x_test_transformed)
accuracy_lr = accuracy_score(y_test, y_pred_lr)
print("Accuracy Logistic Regression:", accuracy_lr)
print("\nClassification Report:")
print(classification_report(y_test, y_pred_lr))
conf_matrix = confusion_matrix(y_test, y_pred_lr)
print("\nConfusion Matrix:")
print(conf_matrix)
```



```
Accuracy Logistic Regression: 0.999

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1607
           1       1.00      1.00      1.00       393

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000


Confusion Matrix:
[[1606    1]
 [   1  392]]
```

Decision Tree:

```
clf_dt = DecisionTreeClassifier()
clf_dt.fit(x_train_transformed, y_train)
y_pred_dt = clf_dt.predict(x_test_transformed)
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print("Accuracy Decision Tree:", accuracy_dt)
print("\nClassification Report:")
print(classification_report(y_test, y_pred_dt))
conf_matrix_dt = confusion_matrix(y_test, y_pred_dt)
print("\nConfusion Matrix:")
print(conf_matrix_dt)
```

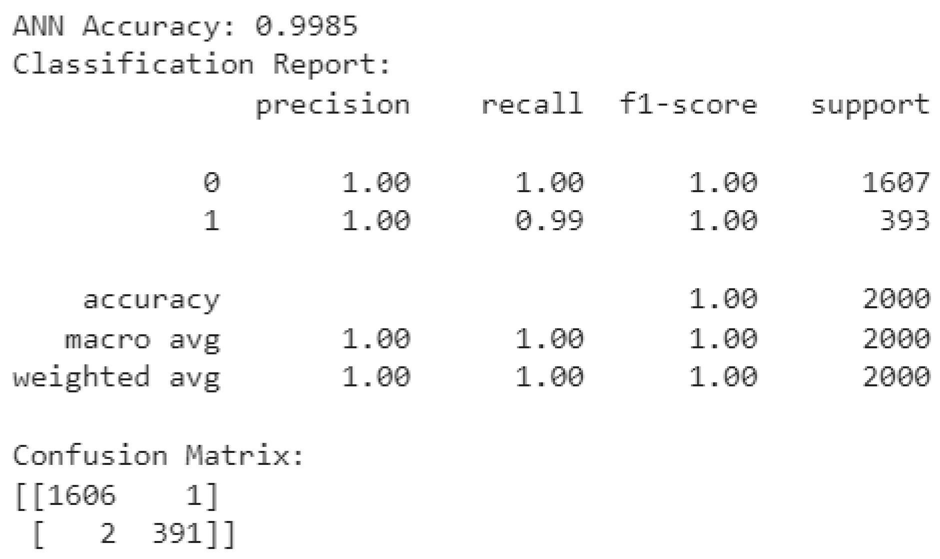


```
Accuracy Decision Tree: 0.9975

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1607
           1       0.99      0.99      0.99       393

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000


Confusion Matrix:
[[1604    3]
 [   2  391]]
```

ANN:

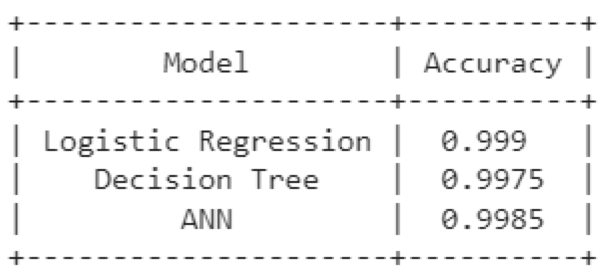
```
model = Sequential()
model.add(Dense(units=128, activation='relu', input_dim=x_train_transformed.shape[1]))
model.add(Dense(units=64, activation='relu'))
model.add(Dense(units=1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x_train_transformed, y_train, epochs=20, batch_size=32, validation_split=0.2)
```

```
y_pred_ann = (model.predict(x_test_transformed) > 0.5).astype("int32")
accuracy_ann = accuracy_score(y_test, y_pred_ann)
print(f'ANN Accuracy: {accuracy_ann}')
print('Classification Report:')
print(classification_report(y_test, y_pred_ann))
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred_ann))
```



```
ANN Accuracy: 0.9985
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1607
           1       1.00      0.99      1.00       393

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000

Confusion Matrix:
[[1606    1]
 [   2  391]]
```

```
table = PrettyTable()
table.field_names = ["Model", "Accuracy"]
table.add_row(["Logistic Regression", accuracy_lr])
table.add_row(["Decision Tree", accuracy_dt])
table.add_row(["ANN", accuracy_ann])
print(table)
```



```
+---------------------+----------+
|        Model        | Accuracy |
+---------------------+----------+
| Logistic Regression |  0.999   |
|    Decision Tree    |  0.9975  |
|         ANN         |  0.9985  |
+---------------------+----------+
```

Conclusion:

In conclusion, the provided code encompasses a comprehensive approach to customer churn prediction, involving data preprocessing, exploratory data analysis (EDA), outlier handling, clustering, and the implementation of three machine learning models: Logistic Regression, Decision Tree, and an Artificial Neural Network (ANN). The EDA phase provided valuable insights into the dataset's structure, and the outlier analysis ensured data robustness. K-Means clustering identified distinct customer groups based on selected features. The machine learning models exhibited exceptional accuracy, with Logistic Regression achieving 99.9%, Decision Tree at 99.75%, and ANN leading with 99.85%. Each model demonstrated strengths based on simplicity, interpretability, or capacity to capture complex patterns. The choice of the best model depends on specific project requirements, and the diverse set of models allows for flexibility in addressing different use cases. Overall, the project offers insights into customer behavior and presents viable models for predicting customer churn in a financial institution.