UNIVERSITI MALAYA
Centre for Continuing Education

MASTER IN DATA SCIENCE

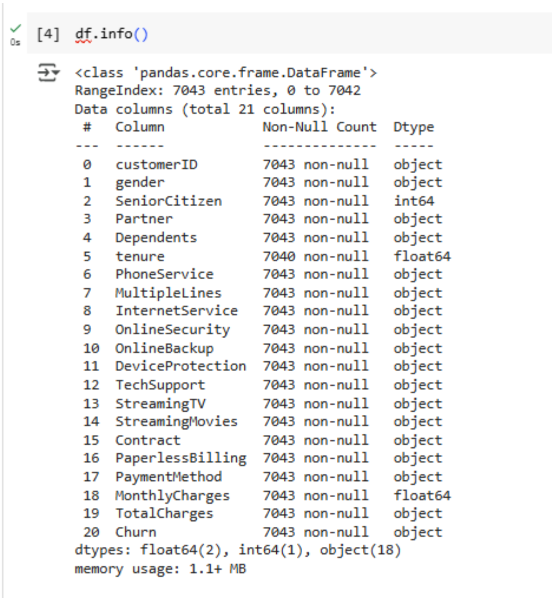
ALTERNATIVE ASSESSMENT 2

COURSE CODE	: WQD7005
COURSE TITLE	: DATA MINING
NAME	: NUR HIDAYAH BINTI AHMAD SHAFII (22120931)
LECTURER	: PROF. DR. NOR LIYANA BT MOHD SHUIB

Table of Contents

1. Data Import, Synthetic Data Generation, and Preprocessing

The data is obtained from Kaggle https://www.kaggle.com/datasets/blastchar/telco-customer-churn/data. It is related to telecommunications customer dataset. It contains well-structured detailed information of customer demographics, service subscriptions, contract details, payment methods, and churn status. There are 21 columns and 7043 rows as shown in Figure 1.1.The Google Colab (https://colab.research.google.com/drive/1ao-HnaA5_t-cxGrSjV8YF5zfttmAmp6c?usp=sharing) was utilized as the primary tool to execute the data preprocessing, feature engineering, and machine learning workflows.

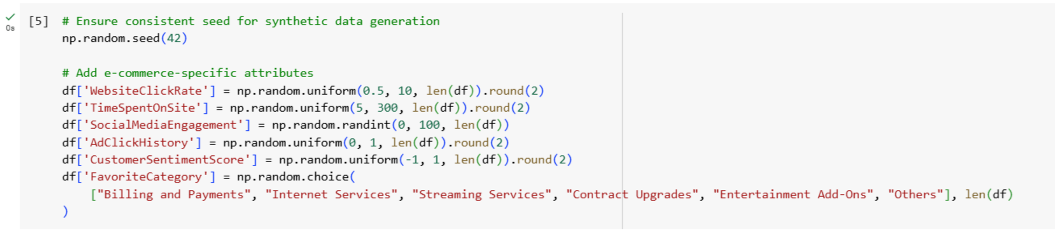


```
[4] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7040 non-null   float64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(2), int64(1), object(18)
memory usage: 1.1+ MB
```

Figure 1.1 Overview of Telecommunications Customer Dataset

1.1 Synthetic Data Generation

Figure 1.2 illustrates the generation of synthetic data by introducing e-commerce-specific attributes to an existing dataset. This process ensures randomness and reproducibility using a fixed seed and generates six new features. The synthetic features, such as WebsiteClickRate, TimeSpentOnSite, and CustomerSentimentScore, were designed to enhance behavioral analysis. Each feature adds valuable information to analyse customer behavior and engagement patterns.



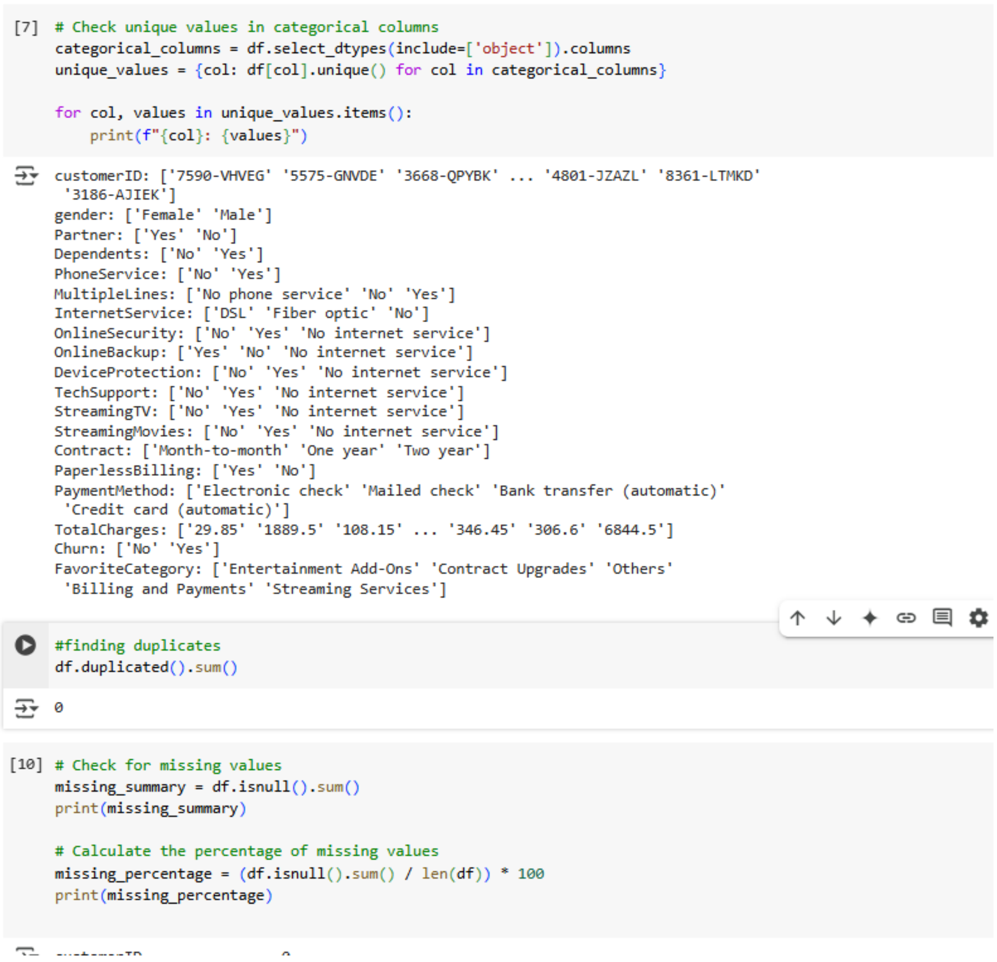
```
[5] # Ensure consistent seed for synthetic data generation
    np.random.seed(42)

    # Add e-commerce-specific attributes
    df['WebsiteClickRate'] = np.random.uniform(0.5, 10, len(df)).round(2)
    df['TimeSpentOnSite'] = np.random.uniform(5, 300, len(df)).round(2)
    df['SocialMediaEngagement'] = np.random.randint(0, 100, len(df))
    df['AdClickHistory'] = np.random.uniform(0, 1, len(df)).round(2)
    df['CustomerSentimentScore'] = np.random.uniform(-1, 1, len(df)).round(2)
    df['FavoriteCategory'] = np.random.choice(
        ["Billing and Payments", "Internet Services", "Streaming Services", "Contract Upgrades", "Entertainment Add-Ons", "Others"], len(df)
    )
```

Figure 1.2 Synthetic Data Generation

1.2 Data Preprocessing

From the result of sanity check as shown in Figure 1.3, several observations were made about the dataset. The unique values in categorical columns such as gender, Partner, Dependents, InternetService, and FavoriteCategory were consistent, with no unusual or inconsistent entries detected. For instance, gender included only "Female" and "Male," while InternetService was limited to "DSL," "Fiber optic," and "No." Additionally, no duplicate rows were found in the dataset.

Regarding missing values, only the tenure column had missing entries with a minimal percentage of 0.042595%. Given the low proportion of missing data, it was decided to drop the rows with missing values in the tenure column rather than impute them. In addition, the 11 blanks (non-numeric) entries in the TotalCharges were also removed. This action ensures the integrity of the tenure data without introducing potential biases through imputation. The TotalCharges were converted to numeric after handling the blanks.



```
[7] # Check unique values in categorical columns
    categorical_columns = df.select_dtypes(include=['object']).columns
    unique_values = {col: df[col].unique() for col in categorical_columns}

    for col, values in unique_values.items():
        print(f"{col}: {values}")
```

```
customerID: ['7590-VHVEG' '5575-GNVDE' '3668-QPYBK' ... '4801-JZAZL' '8361-LTMKD'
 '3186-AJIEK']
gender: ['Female' 'Male']
Partner: ['Yes' 'No']
Dependents: ['No' 'Yes']
PhoneService: ['No' 'Yes']
MultipleLines: ['No phone service' 'No' 'Yes']
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: ['No' 'Yes' 'No internet service']
OnlineBackup: ['Yes' 'No' 'No internet service']
DeviceProtection: ['No' 'Yes' 'No internet service']
TechSupport: ['No' 'Yes' 'No internet service']
StreamingTV: ['No' 'Yes' 'No internet service']
StreamingMovies: ['No' 'Yes' 'No internet service']
Contract: ['Month-to-month' 'One year' 'Two year']
PaperlessBilling: ['Yes' 'No']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
TotalCharges: ['29.85' '1889.5' '108.15' ... '346.45' '306.6' '6844.5']
Churn: ['No' 'Yes']
FavoriteCategory: ['Entertainment Add-Ons' 'Contract Upgrades' 'Others'
 'Billing and Payments' 'Streaming Services']
```

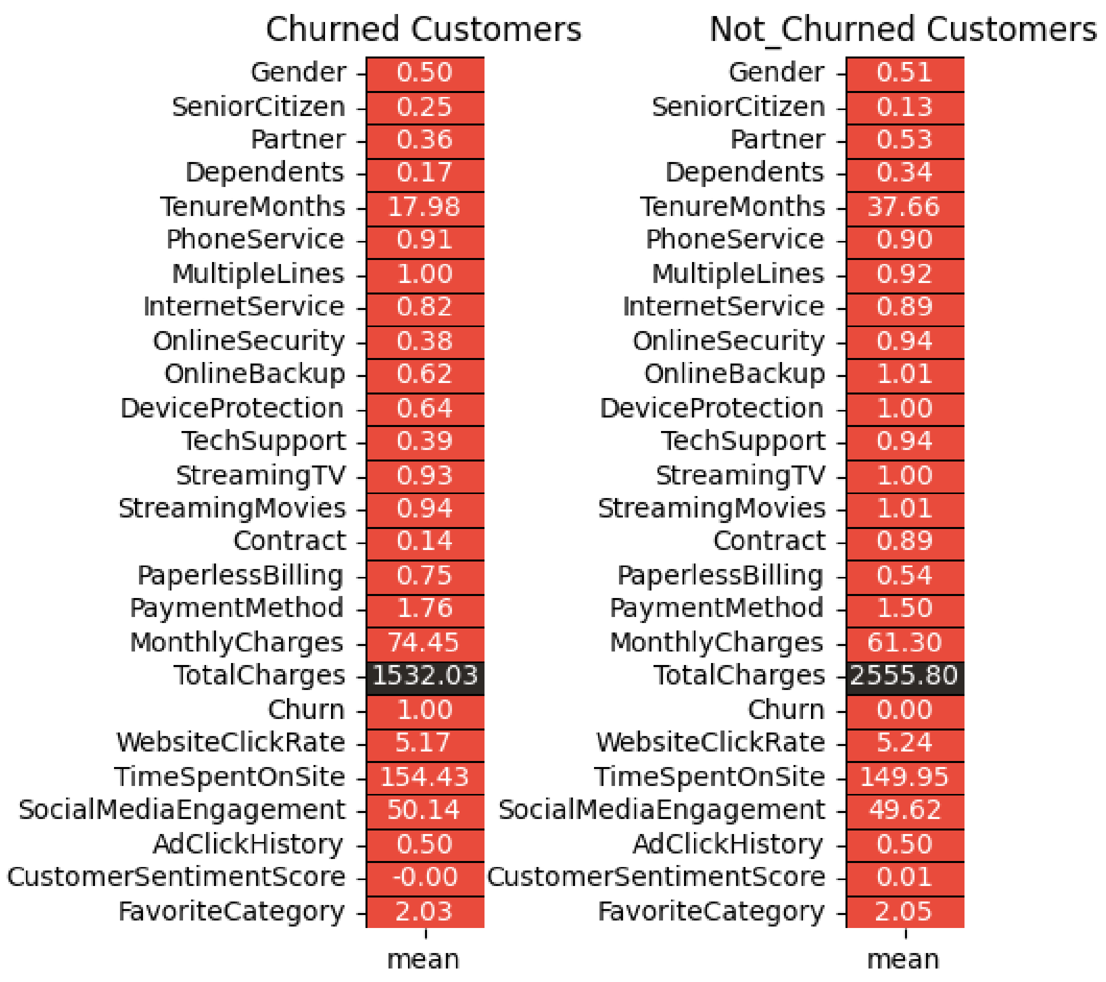
```
#finding duplicates
df.duplicated().sum()
```

```
0
```

```
[10] # Check for missing values
     missing_summary = df.isnull().sum()
     print(missing_summary)

     # Calculate the percentage of missing values
     missing_percentage = (df.isnull().sum() / len(df)) * 100
     print(missing_percentage)
```

Figure 1.3 Sanity Check

Next, the tenure and gender were renamed to TenureMonths, and Gender respectively. These changes intended to enhance readability and clarity in the dataset. The customerID column were dropped as it was irrelevant. Finally, the categorical variables with object data types were transformed into numeric labels using Label Encoding. This transformation ensures that categorical data can be processed effectively by models that require numeric inputs.



	Churned Customers mean	Not_Churned Customers mean
Gender	0.50	0.51
SeniorCitizen	0.25	0.13
Partner	0.36	0.53
Dependents	0.17	0.34
TenureMonths	17.98	37.66
PhoneService	0.91	0.90
MultipleLines	1.00	0.92
InternetService	0.82	0.89
OnlineSecurity	0.38	0.94
OnlineBackup	0.62	1.01
DeviceProtection	0.64	1.00
TechSupport	0.39	0.94
StreamingTV	0.93	1.00
StreamingMovies	0.94	1.01
Contract	0.14	0.89
PaperlessBilling	0.75	0.54
PaymentMethod	1.76	1.50
MonthlyCharges	74.45	61.30
TotalCharges	1532.03	2555.80
Churn	1.00	0.00
WebsiteClickRate	5.17	5.24
TimeSpentOnSite	154.43	149.95
SocialMediaEngagement	50.14	49.62
AdClickHistory	0.50	0.50
CustomerSentimentScore	-0.00	0.01
FavoriteCategory	2.03	2.05

Figure 1.4 **Mean** Values Features for Churned and Not-Churned Customers

Based on Figure 1.4, average tenure of churned customers is significantly lower at 17.98 months compared to 37.57 months for non-churned customers. This suggests that longer tenure is associated with lower churn rates. Next, non-churned customers also show higher engagement with services and stronger online engagement across website interactions, social media, and advertisements.This indicates that these services could positively impact customer retention. In addition, mean value for the Contract feature is much higher for non-churned customers. It indicates they are more likely to be on longer-term contracts compared to churned customers, who likely opt for month-to-month plans. Interestingly, churned customers face higher average monthly charges at 74.44 compared to 61.27 for non-churned customers. It might contribute to dissatisfaction. However, total charges are significantly higher for non-churned customers (2557.31) due to their longer tenure.

1.3 Specify Variables Role

The variables were categorized by role as shown in Table 1.1. The dataset includes CustomerID as the unique identifier and Churn as the target variable. Demographic variables (e.g., Gender, SeniorCitizen), service details (e.g., PhoneService, InternetService), contract information (e.g., Contract, PaymentMethod), financial data (e.g., MonthlyCharges, TotalCharges), and tenure (TenureMonths) are categorized to streamline the analysis process and ensure proper application of statistical and machine learning techniques.

Table 1.1 Categorization of Variables by Roles

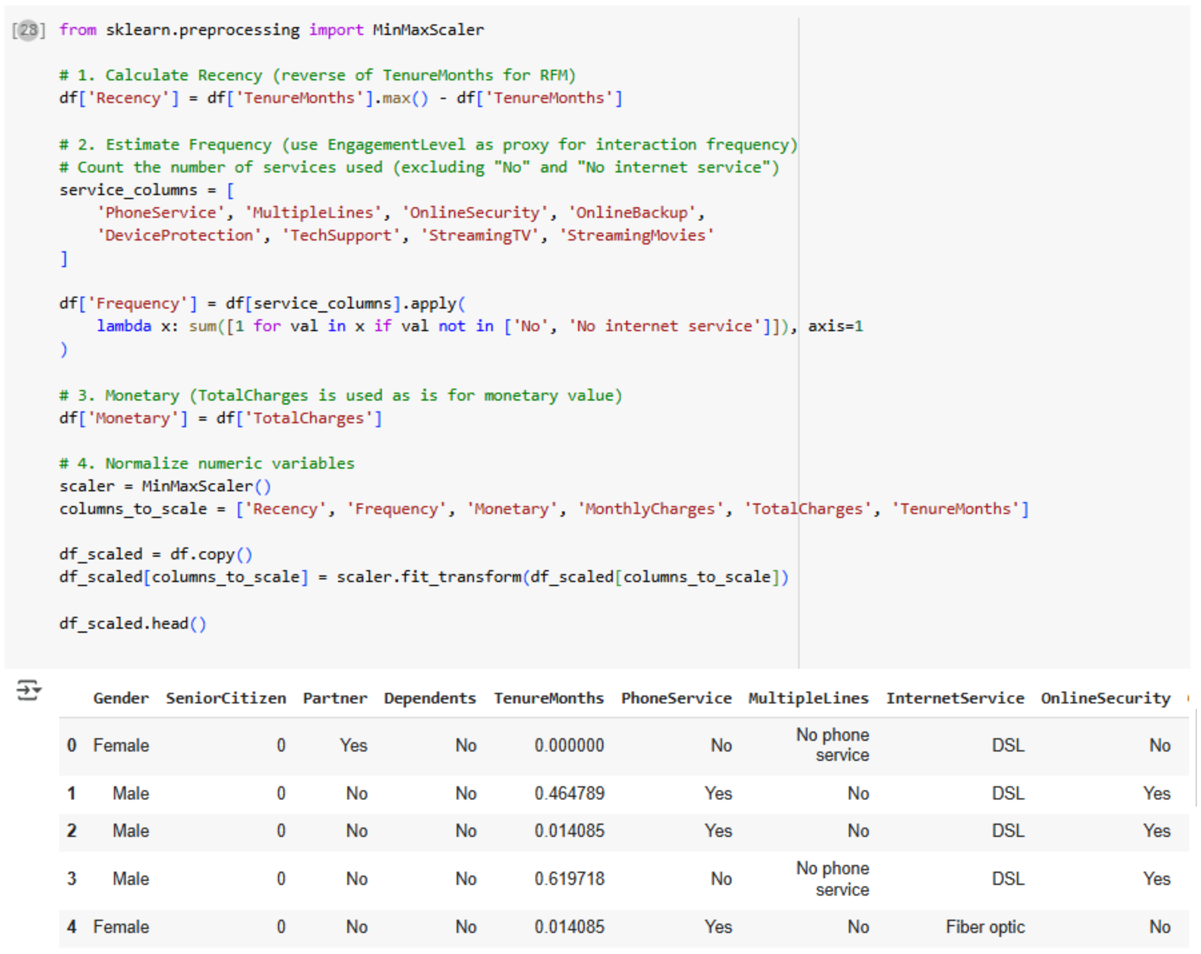
Variable Role	Variables
ID Variable	CustomerID
Target Variable	Churn
Demographic Variables	Gender, SeniorCitizen, Partner, Dependents
Service Variables	PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies
Contract Information	Contract, PaperlessBilling, PaymentMethod
Financial Variables	MonthlyCharges, TotalCharges
Tenure Information	TenureMonths

1.4 Feature Engineering

We introduced RFM (Recency, Frequency, and Monetary) metrics to better understand customer behavior as shown in Figure 1.4. Recency was calculated as the reverse of the TenureMonths. It represent the time elapsed since the beginning of a customer’s tenure. Smaller values indicate more recent engagement. Next, frequency was estimated using engagement levels by counting the number of subscribed services (e.g., PhoneService, MultipleLines, OnlineSecurity) for each customer. Excluded from this count were services marked as "No" or "No internet service," to ensure that only active engagements contributed to this metrics. On the other hand, monetary value was directly derived from the TotalCharges column which reflect the total revenue generated from each customer.

To standardize the dataset and enable fair comparisons, numerical features including Recency, Frequency, Monetary, MonthlyCharges, TenureMonths and TotalCharges were normalized using the MinMaxScaler. MinMaxScaler is ideal for datasets with uniform distributions and no significant outliers. This transformation scaled all values to a uniform range between 0 and 1 to remove the effects of varying magnitudes across features. This structured preprocessing ensures the dataset is optimized for meaningful insights and actionable outcomes.

Selecting an appropriate scaling method that preserves feature relationships without distorting the data was an important consideration. The MinMaxScaler was chosen for its simplicity and effectiveness in normalizing features to a consistent range of 0–1. This approach ensures fair comparisons between features with changing magnitudes and prevents features with large values from dominating the analysis. By scaling the data uniformly, MinMaxScaler helps maintain the integrity of feature relationships while optimizing the dataset for machine learning and statistical analysis.



```python
from sklearn.preprocessing import MinMaxScaler

# 1. Calculate Recency (reverse of TenureMonths for RFM)
df['Recency'] = df['TenureMonths'].max() - df['TenureMonths']

# 2. Estimate Frequency (use EngagementLevel as proxy for interaction frequency)
# Count the number of services used (excluding "No" and "No internet service")
service_columns = [
    'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup',
    'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies'
]

df['Frequency'] = df[service_columns].apply(
    lambda x: sum([1 for val in x if val not in ['No', 'No internet service']]), axis=1
)

# 3. Monetary (TotalCharges is used as is for monetary value)
df['Monetary'] = df['TotalCharges']

# 4. Normalize numeric variables
scaler = MinMaxScaler()
columns_to_scale = ['Recency', 'Frequency', 'Monetary', 'MonthlyCharges', 'TotalCharges', 'TenureMonths']

df_scaled = df.copy()
df_scaled[columns_to_scale] = scaler.fit_transform(df_scaled[columns_to_scale])

df_scaled.head()
```

	Gender	SeniorCitizen	Partner	Dependents	TenureMonths	PhoneService	MultipleLines	InternetService	OnlineSecurity
0	Female	0	Yes	No	0.000000	No	No phone service	DSL	No
1	Male	0	No	No	0.464789	Yes	No	DSL	Yes
2	Male	0	No	No	0.014085	Yes	No	DSL	Yes
3	Male	0	No	No	0.619718	No	No phone service	DSL	Yes
4	Female	0	No	No	0.014085	Yes	No	Fiber optic	No

Figure 1.5 Feature Engineering

2 Decision Tree Analysis

For the decision tree analysis, we focused on the correlation of features with respect to the target variable as shown in Figure 2.1. The features with correlation coefficient between (-0.1,0.1) were identified due to a weak relationship with Churn and were subsequently dropped.

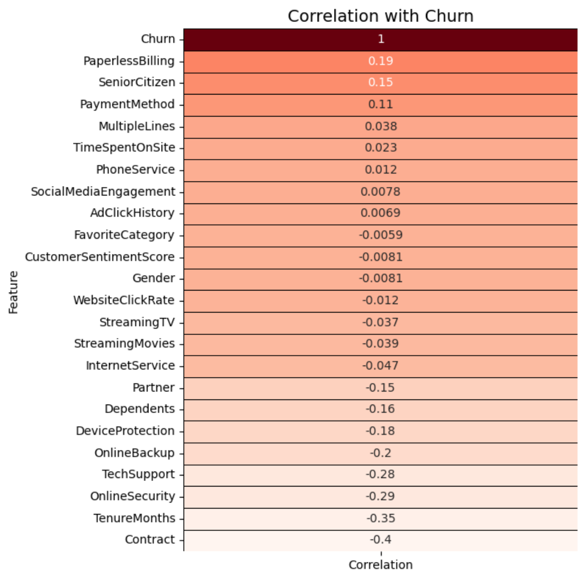


Figure 2.1 Correlation Analysis

Based on the chi-squared analysis in Figure 2.2, several features demonstrate minimal relevance to the Churn due to low chi-squared scores. Features such as 'InternetService', 'StreamingMovies', 'StreamingTV' contribute little to the predictive power of the model and can be considered irrelevant for the analysis.

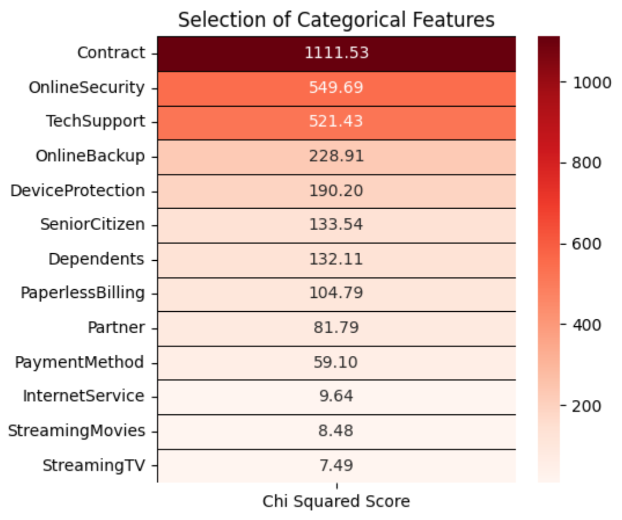


Figure 2.2 Selection of Categorical Features

Since all numerical features in Figure 2.3 show meaningful relationships with the target variable (Churn) based on their ANOVA scores, all three numerical features should be included in the modeling process. Each feature contributes valuable information and removing any could potentially harm the model's predictive performance.

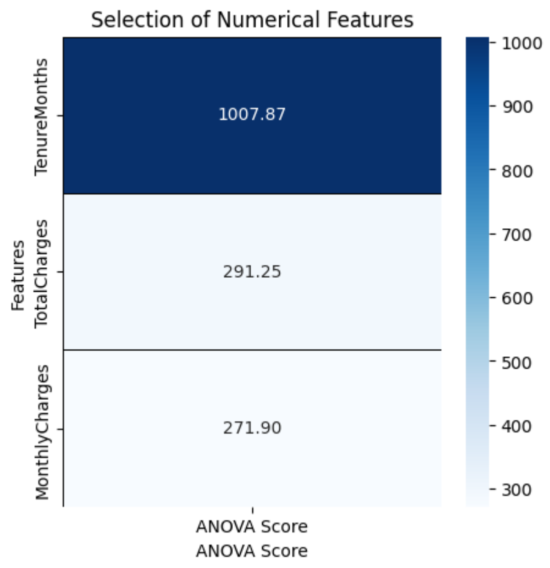
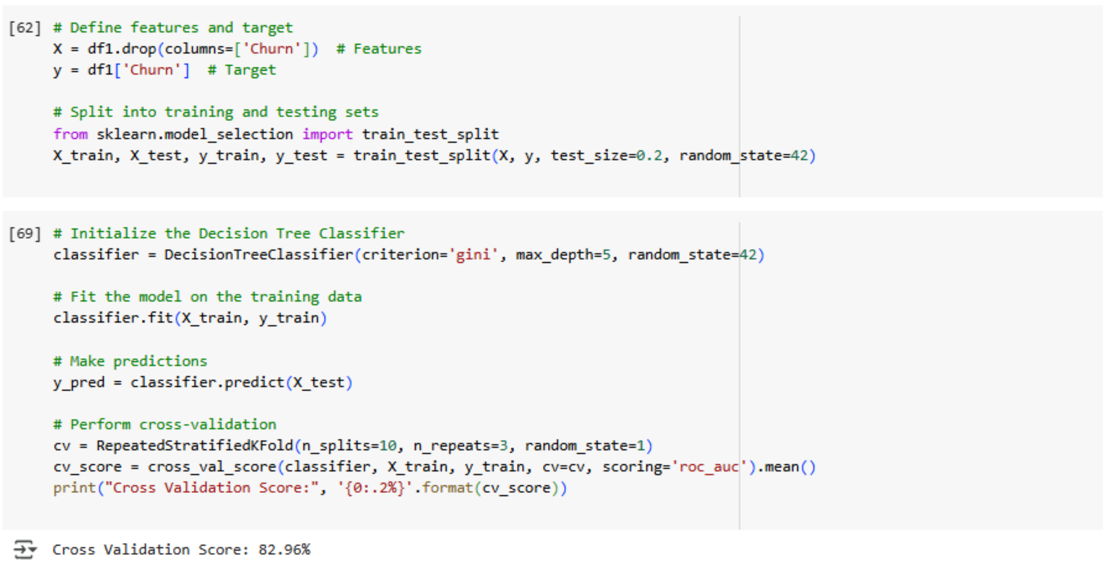


Figure 2.3 Selection of Numerical Features

The dataset was split into training (80%) and testing (20%) sets to enable reliable evaluation of model performance. A Decision Tree Classifier was trained with a maximum depth of 5 to balance model complexity and avoid overfitting as showin in Figure 2.4. The cross validation score of 82.96% means that the model performs consistently and generalizes well across different subsets of the data. This score measures the model's ability to distinguish between the two classes: Churn and Not Churn.



```
[62] # Define features and target
     X = df1.drop(columns=['Churn'])  # Features
     y = df1['Churn']  # Target

     # Split into training and testing sets
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[69] # Initialize the Decision Tree Classifier
     classifier = DecisionTreeClassifier(criterion='gini', max_depth=5, random_state=42)

     # Fit the model on the training data
     classifier.fit(X_train, y_train)

     # Make predictions
     y_pred = classifier.predict(X_test)

     # Perform cross-validation
     cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
     cv_score = cross_val_score(classifier, X_train, y_train, cv=cv, scoring='roc_auc').mean()
     print("Cross Validation Score:", '{0:.2%}'.format(cv_score))
```

```
Cross Validation Score: 82.96%
```

Figure 2.4 Decision Tree

Figure 2.4 indicates that the decision tree classifier achieved a ROC-AUC score of 69.26%. The result shows a moderate ability to distinguish between Churn and Not Churn. The curve shows better-than-random performance with the classifier capturing some patterns in the data. However, the model's predictive capability is not optimal as a perfect classifier would have a ROC-AUC score closer to 100%. The model could be improved through feature selection, hyperparameter tuning, or incorporating additional relevant features.

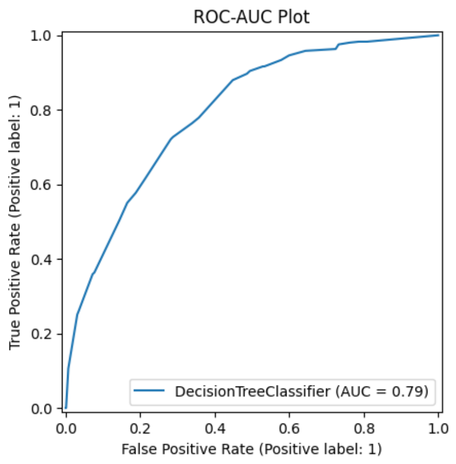


Figure 2.5 Decision Tree ROC-AUC Graph

Figure 2.6 shows the performance of the decision tree classifier to predict Churn. The model correctly identified 834 true negatives (59.32%), where customers who did not churn were classified correctly. Similarly, it identified 224 true positives (15.93%), where customers who churned were accurately predicted. However, the model also misclassified 165 false positives (11.74%), where non-churning customers were incorrectly classified as churned, and 183 false negatives (13.02%), where churning customers were missed. These results suggest that the model performs moderately well but struggles to balance sensitivity.

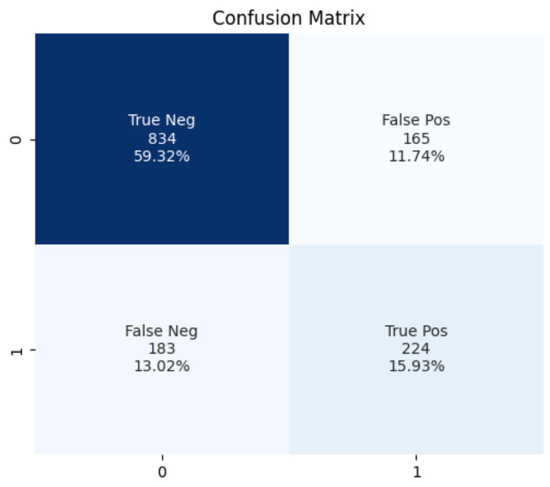


Figure 2.6 Decision Tree Confusion Matrix

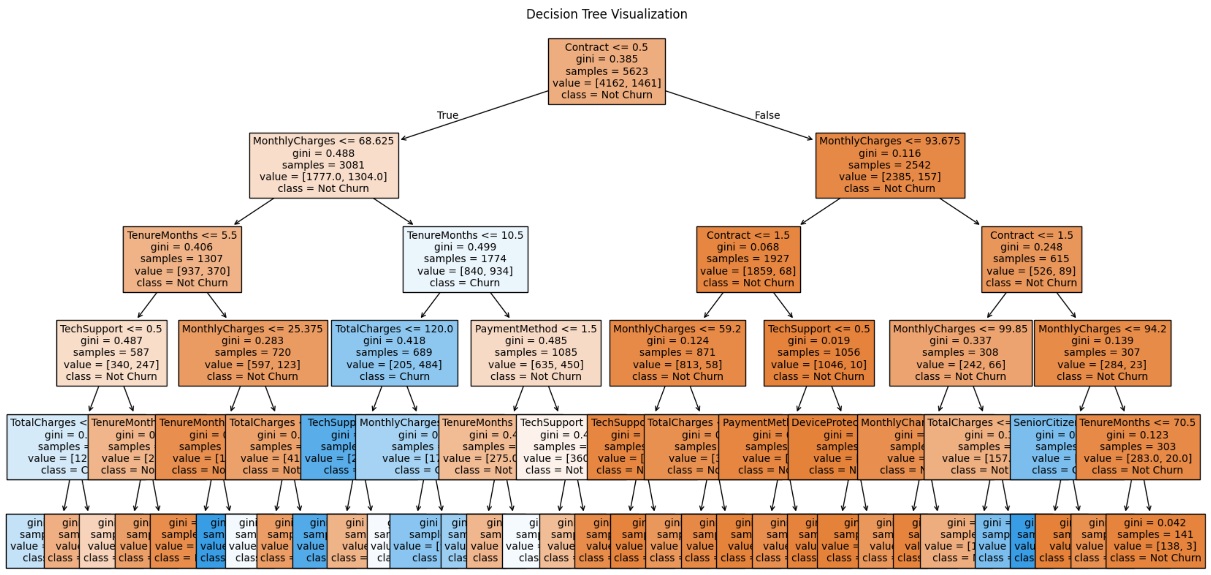


Figure 2.7 Decision Tree Visualization

Figure 2.7 illustrates the key factors of classifier splits the data based on features like MonthlyCharges, TenureMonths, and Contract to predict Churn. The tree starts with the most significant feature (Contract) and continues to create branches.This visualization highlights the structure and logic of the decision-making process.

Table 2.1 indicates the performance of the decision tree classifier to predict Churn. For the majority class (Not Churn), the precision, recall, and F1-score are 0.82, 0.83, and 0.83, respectively. It highlights strong performance. However, for the minority class (Churn), the precision, recall, and F1-score drop to 0.58, 0.55, and 0.56. This means that the model has difficulty to identify customers who churn. The overall accuracy of the model is 75%, with a macro average F1-score of 0.70. It showed moderate effectiveness in handling class imbalance. Further optimization could improve recall and F1-score for the minority class.

Table 2.1 Classification Report of Decision Tree

	Precision	Recall	F1-Score	Support
0	0.82	0.83	0.83	999
1	0.58	0.55	0.56	407
Accuracy			0.75	1406
Macro avg	0.70	0.69	0.70	1406
Weighted avg	0.75	0.75	0.75	1406

3 Advanced Techniques in Customer Segmentation

For the customer segmentation analysis, K-means clustering was applied to group customers based on their behavioral data. The process began with data preprocessing, where the Churn column was removed as it was not relevant for clustering and the remaining features were scaled using StandardScaler to ensure all features were on the same scale. Next, Principal Component Analysis (PCA) was applied to reduce the dataset's dimensionality to two principal components for visualization. The PCA output shows that the first two principal components explain approximately 28.39% and 15.38% of the variance, respectively. The total variance of ~43.77% of the total variance provides a reasonable representation of the data for clustering and visualization purposes.

Using the reduced data, the K-means algorithm was applied with the number of clusters set to 4. This value was chosen to balance interpretability and meaningful segmentation. The clustering results were evaluated using the silhouette score which was 0.39. It demonstrated moderate separation between clusters. A scatter plot in Figure 3.1 was generated to visualize the clusters in the reduced PCA space. Each cluster corresponds to a group of customers with similar behavioral traits.

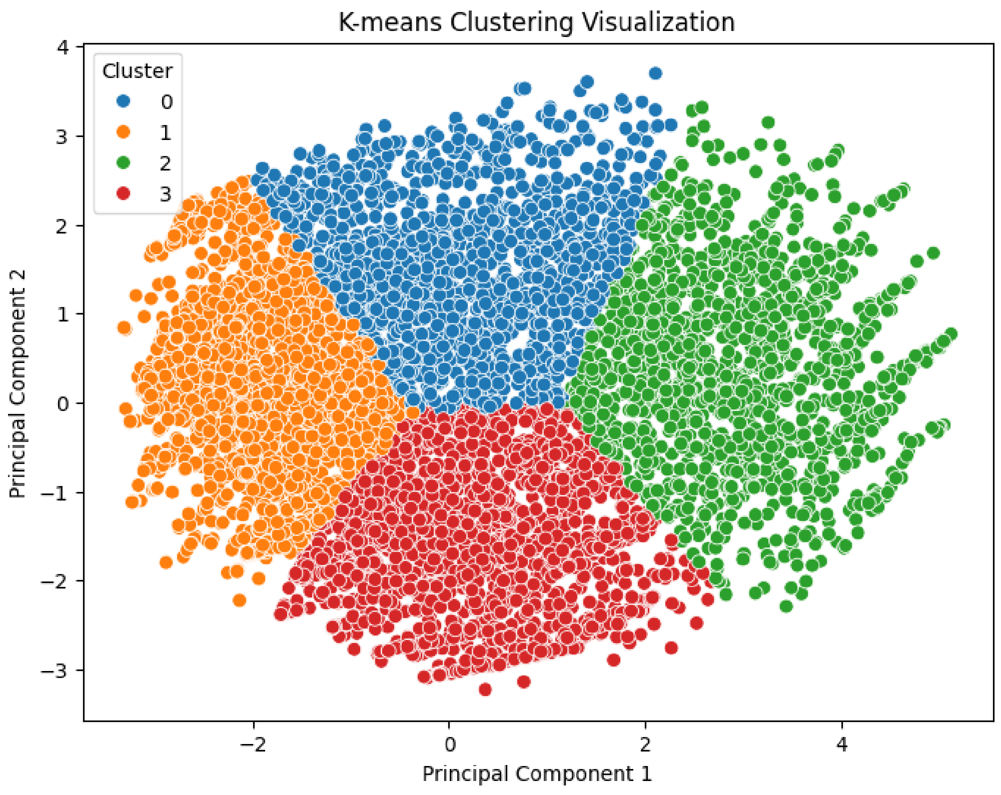
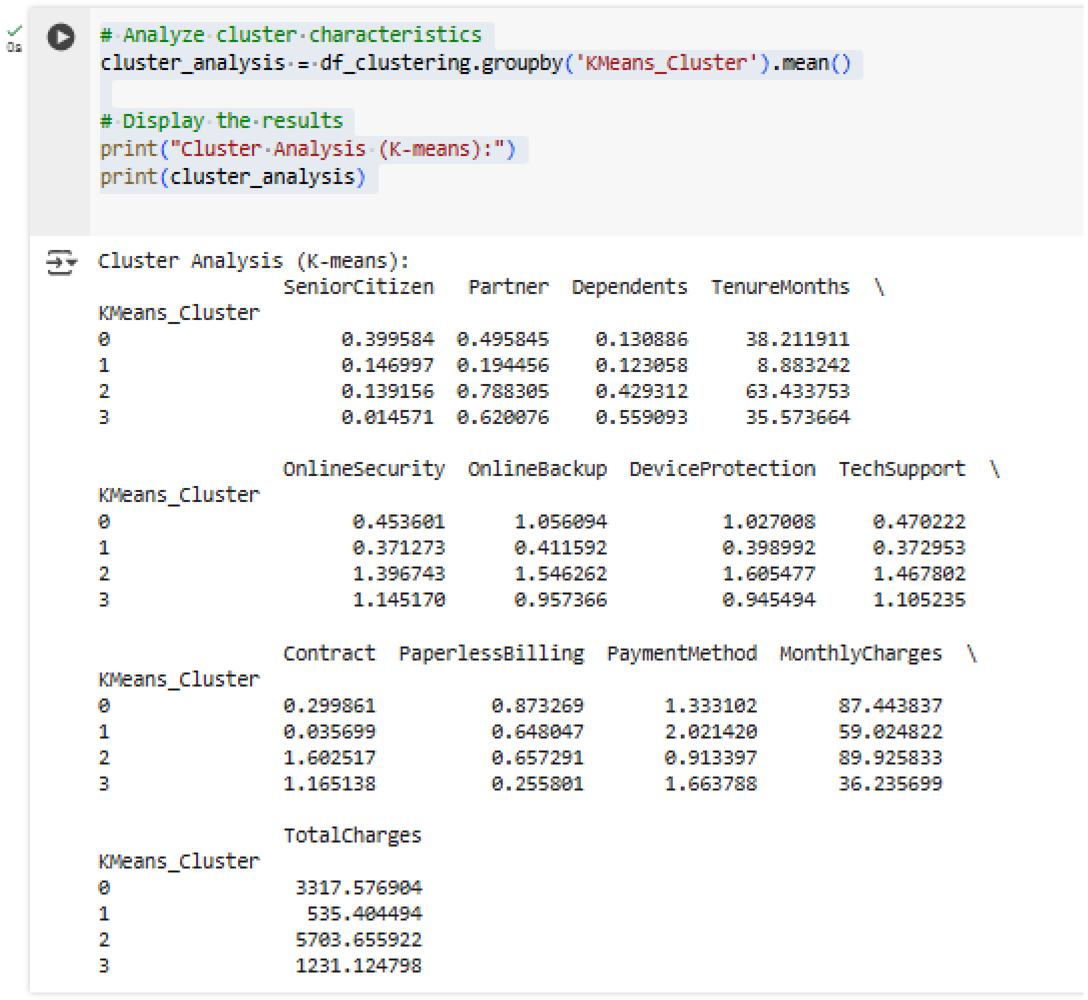


Figure 3.1 K-Means Scatter Plot

Finally, a detailed cluster analysis was conducted by examining the mean values of each feature for the four clusters. This step provided valuable insights into the behavioral characteristics of each customer segment, such as tenure, monthly charges, and adoption of additional services. The analysis shown in Figure 3.2 enabled actionable recommendations tailored to the specific needs and behaviors of each cluster.



```
# Analyze cluster characteristics
cluster_analysis = df_clustering.groupby('KMeans_Cluster').mean()

# Display the results
print("Cluster Analysis (K-means):")
print(cluster_analysis)
```

```
Cluster Analysis (K-means):
                SeniorCitizen   Partner  Dependents  TenureMonths  \
KMeans_Cluster
0                    0.399584  0.495845    0.130886     38.211911
1                    0.146997  0.194456    0.123058      8.883242
2                    0.139156  0.788305    0.429312     63.433753
3                    0.014571  0.620076    0.559093     35.573664

                OnlineSecurity  OnlineBackup  DeviceProtection  TechSupport  \
KMeans_Cluster
0                     0.453601      1.056094          1.027008     0.470222
1                     0.371273      0.411592          0.398992     0.372953
2                     1.396743      1.546262          1.605477     1.467802
3                     1.145170      0.957366          0.945494     1.105235

                Contract  PaperlessBilling  PaymentMethod  MonthlyCharges  \
KMeans_Cluster
0               0.299861          0.873269       1.333102       87.443837
1               0.035699          0.648047       2.021420       59.024822
2               1.602517          0.657291       0.913397       89.925833
3               1.165138          0.255801       1.663788       36.235699

                TotalCharges
KMeans_Cluster
0                3317.576904
1                 535.404494
2                5703.655922
3                1231.124798
```

Figure 3.2 K-Means Cluster Analysis

According to Figure 3.2, there is four distinct groups whereby each characterized by unique behavioral patterns and engagement levels. Firstly, Cluster 0 consist customers with a moderate average tenure of approximately 38 months and the highest monthly charges of 87.44. These customers have a high adoption rate of add-on services such as OnlineBackup (1.05), DeviceProtection (1.03), and OnlineSecurity (0.45). With most of them enrolled in paperless billing (87%), their total charges are also the highest among all clusters, at 3317.57. This group represents loyal and high-value customers. Therefore, we should focus on retention through personalized rewards and exclusive offers to recognize their value and maintain their satisfaction.

In contrast, Cluster 1 includes customers with the shortest average tenure of 8.8 months and the lowest total charges, at 535.46. These customers exhibit minimal engagement with services such as OnlineBackup (0.41) and DeviceProtection (0.39), while their monthly charges are moderate, at 59.42. Approximately 64% use paperless billing. This segment likely consists of new or at-risk customers who require attention through onboarding programs, introductory promotions, and improved customer engagement to reduce churn risk. Therefore, we should implement proactive customer support initiatives, such as regular follow-ups or onboarding assistance, to enhance their experience and engagement.

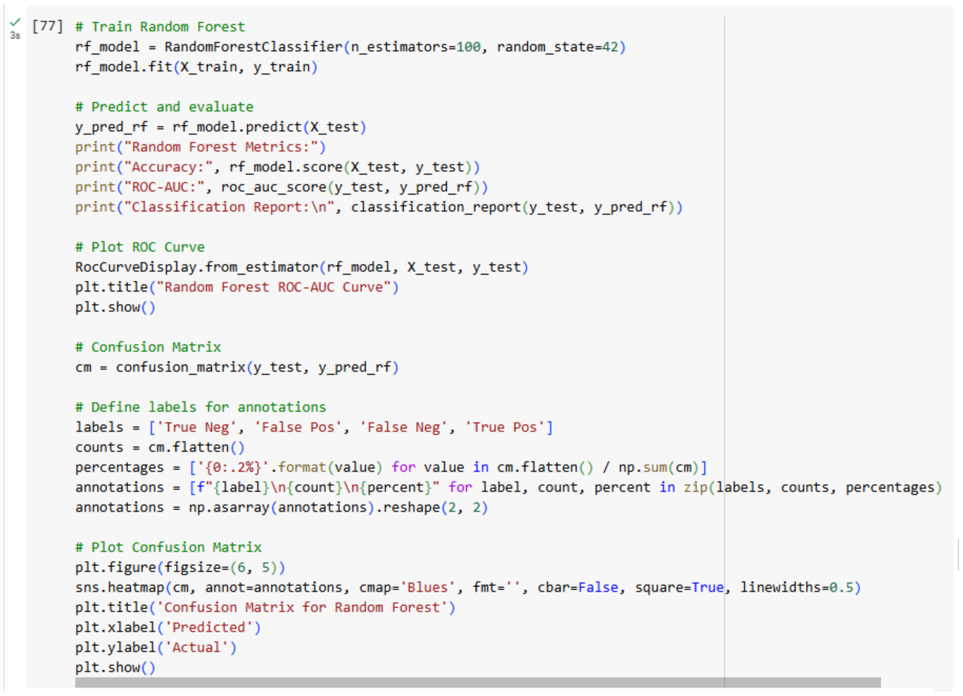
In addition, Cluster 2 represents long-term, highly engaged customers with the longest tenure of 63.43 months and high total charges of 5703.45. These customers demonstrate the highest adoption rates of additional services such as OnlineBackup (1.55), DeviceProtection (1.61), and OnlineSecurity (1.40). Most customers in this group are on long-term contracts (1.26), but fewer use paperless billing (25%). These are highly profitable and loyal customers. Efforts should focus on offering premium services and tailored incentives to enhance their experience and ensure long-term retention. For example, long-term contract renewal incentives such as discounted rates or added benefits, to ensure continued retention.

Lastly, Cluster 3 consists of customers with a medium tenure of 35.57 months and the lowest monthly charges, at 36.23. This group has moderate adoption of services like OnlineBackup (0.96) and DeviceProtection (0.95) and a lower reliance on paperless billing (66%). With total charges at 1231.12, these customers appear to be cost-sensitive. Upsell bundled services at discounted rates and implement retention strategies designed for cost-conscious customers to increase their engagement and value. Additionally, provide personalized recommendations based on their usage patterns to enhance their experience.

In conclusion, high-value customers in Cluster 0 and long-term customers in Cluster 2 should be prioritized for retention through loyalty programs. In contrast, cost-sensitive customers in Cluster 3 and at-risk customers in Cluster 1 require focused efforts to increase engagement and reduce churn.

4 Ensemble Methods

4.1 Apply Bagging (Random Forests)



```
[77] # Train Random Forest
     rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
     rf_model.fit(X_train, y_train)

     # Predict and evaluate
     y_pred_rf = rf_model.predict(X_test)
     print("Random Forest Metrics:")
     print("Accuracy:", rf_model.score(X_test, y_test))
     print("ROC-AUC:", roc_auc_score(y_test, y_pred_rf))
     print("Classification Report:\n", classification_report(y_test, y_pred_rf))

     # Plot ROC Curve
     RocCurveDisplay.from_estimator(rf_model, X_test, y_test)
     plt.title("Random Forest ROC-AUC Curve")
     plt.show()

     # Confusion Matrix
     cm = confusion_matrix(y_test, y_pred_rf)

     # Define labels for annotations
     labels = ['True Neg', 'False Pos', 'False Neg', 'True Pos']
     counts = cm.flatten()
     percentages = ['{0:.2%}'.format(value) for value in cm.flatten() / np.sum(cm)]
     annotations = [f"{label}\n{count}\n{percent}" for label, count, percent in zip(labels, counts, percentages)]
     annotations = np.asarray(annotations).reshape(2, 2)

     # Plot Confusion Matrix
     plt.figure(figsize=(6, 5))
     sns.heatmap(cm, annot=annotations, cmap='Blues', fmt='', cbar=False, square=True, linewidths=0.5)
     plt.title('Confusion Matrix for Random Forest')
     plt.xlabel('Predicted')
     plt.ylabel('Actual')
     plt.show()
```

Figure 4.1 Random Forest Code Snippet

The Random Forest model as demonstrated in Figure 4.1 show a moderate performance to predict customer churn. It has accuracy of 76.53%, cross validation score of 79.48% and ROC-AUC score of 80%. Figure 4.2 indicates that the model has a good ability to distinguish between churned and non-churned customers.

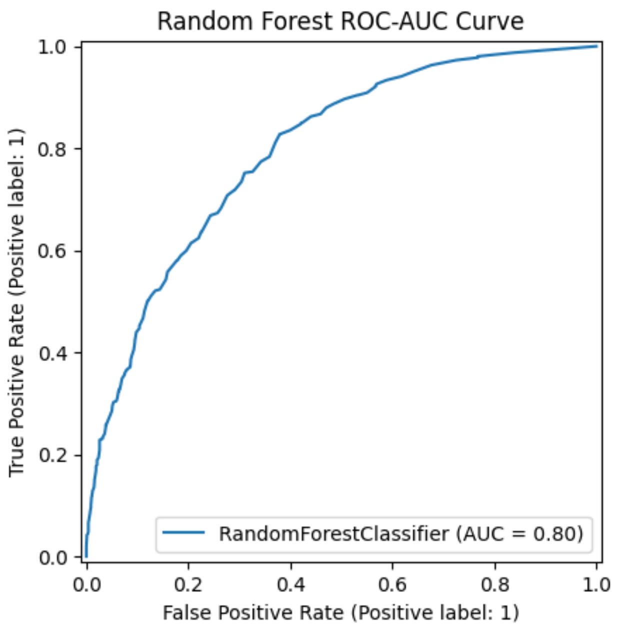


Figure 4.2 Random Forest ROC-AUC Graph

Figure 4.3 showed that the model correctly classified 904 true negatives (64.30%), where non-churning customers were accurately identified and 172 true positives (12.23%) where churned customers correctly predicted by the model. However, there are 95 false positives (6.76%), where non-churning customers were incorrectly classified as churning, and 235 false negatives (16.71%), where churned customers were misclassified as non-churning.

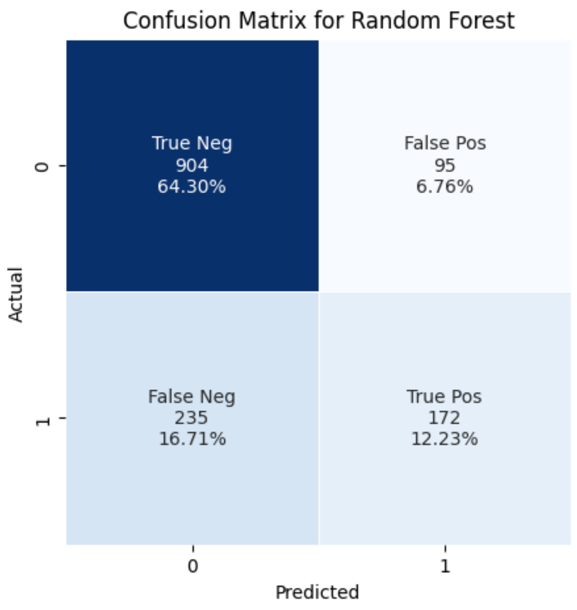


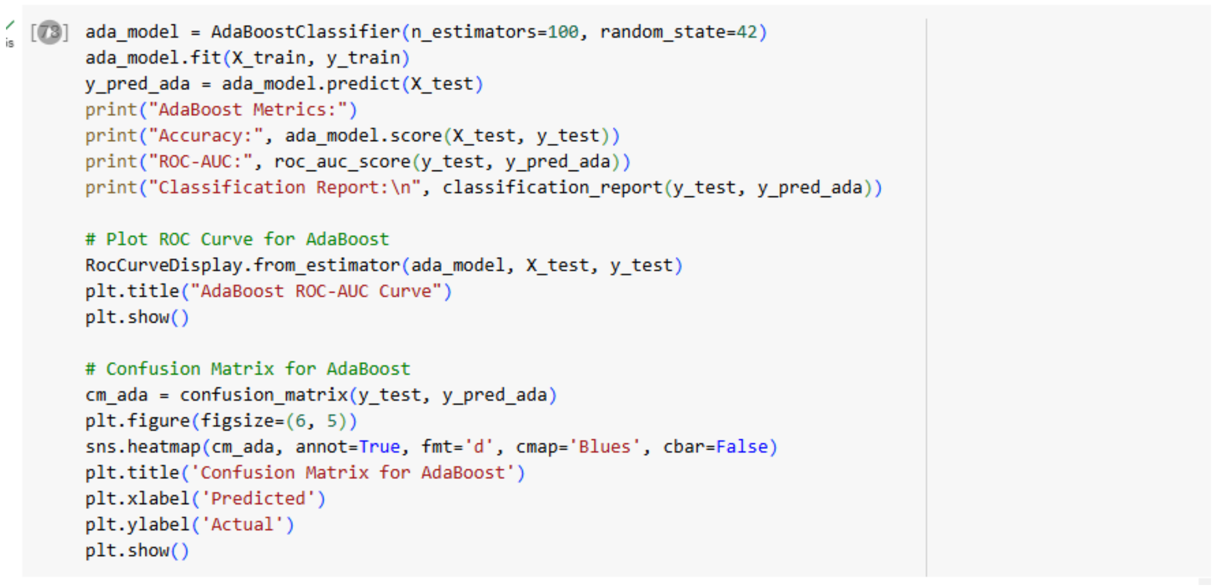
Figure 4.3 Confusion Matrix of Random Forest

Table 4.1 indicates the performance of the Random Forest to predict Churn. For non-churned customers, the model achieved a precision of 0.79, recall of 0.90, and an F1-score of 0.85.It showed strong performance in correctly identifying customers who will not churn. However, the model showed a lower precision of 0.64, recall of 0.42, and an F1-score of 0.56 for churned customers. It reflected the difficulty of model to predict customers who will churn. The overall accuracy of the model is 77%, with a weighted average F1-score of 0.75. This means the model performs moderately well across both classes. The disparity in recall between the two classes indicates that the model is more effective at identifying non-churned customers, which could be addressed through techniques such as dataset balancing, hyperparameter tuning, or using ensemble methods with class weights to improve the model's sensitivity toward churned customers.

Table 4.1 Classification Report of Random Forest

	Precision	Recall	F1-Score	Support
0	0.79	0.90	0.85	999
1	0.64	0.42	0.56	407
Accuracy			0.77	1406
Macro avg	0.72	0.66	0.78	1406
Weighted avg	0.75	0.77	0.75	1406

4.2 AdaBoost



```
ada_model = AdaBoostClassifier(n_estimators=100, random_state=42)
ada_model.fit(X_train, y_train)
y_pred_ada = ada_model.predict(X_test)
print("AdaBoost Metrics:")
print("Accuracy:", ada_model.score(X_test, y_test))
print("ROC-AUC:", roc_auc_score(y_test, y_pred_ada))
print("Classification Report:\n", classification_report(y_test, y_pred_ada))

# Plot ROC Curve for AdaBoost
RocCurveDisplay.from_estimator(ada_model, X_test, y_test)
plt.title("AdaBoost ROC-AUC Curve")
plt.show()

# Confusion Matrix for AdaBoost
cm_ada = confusion_matrix(y_test, y_pred_ada)
plt.figure(figsize=(6, 5))
sns.heatmap(cm_ada, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title('Confusion Matrix for AdaBoost')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Figure 4.4 AdaBoost Code Snippet

The AdaBoost model of Figure 4.4 demonstrated an accuracy of 78.31%. It showed that the model correctly classified 78.31% of the test data. The ROC-AUC score of 70.32% suggests that the model has a moderate ability to distinguish between churned and non-churned customers. The AUC score of 83% in Figure 4.5 indicated that the model effectively identifies positive (Churn) and negative (Non-Churn) cases. The curve rises steeply demonstrated a high true positive rate (sensitivity) even at a relatively low false positive rate.

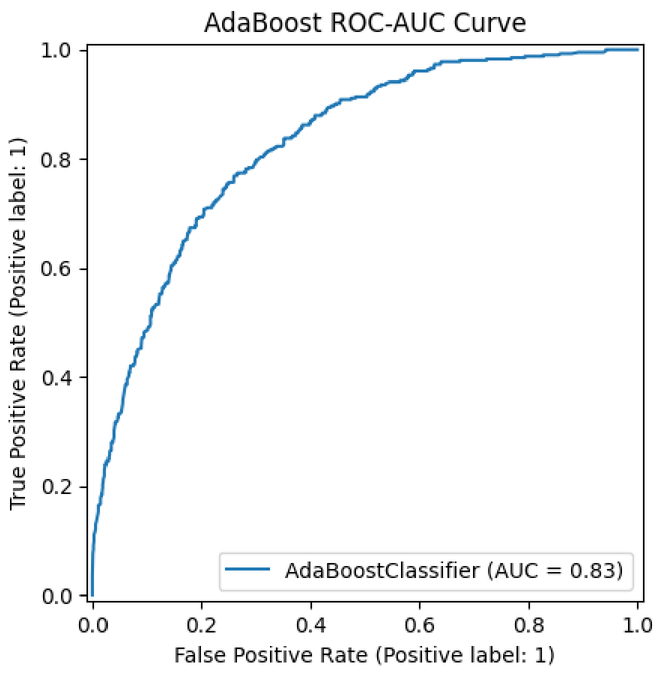


Figure 4.5 Ada Boost ROC-AUC Graph

Figure 4.6 illustrated AdaBoost model performance to predict customer churn. The model correctly identified 892 non-churned customers (63.44%) and 209 churned customers (14.86%). It demonstrated moderate effectiveness in predicting both classes. However, the model misclassified 107 non-churned customers as churned (7.61%) and 198 churned customers as non-churned (14.08%). This means that the model has difficulty to detect churned customers. The relatively high number of false negatives suggests that further refinement is needed to improve its sensitivity toward churned customers and reduce missed churn predictions.

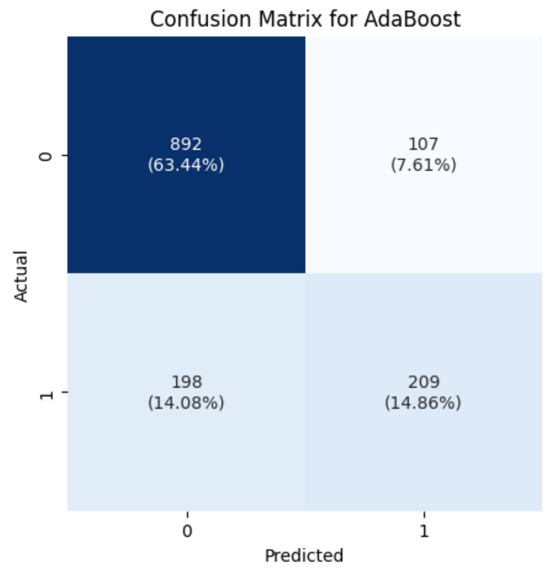


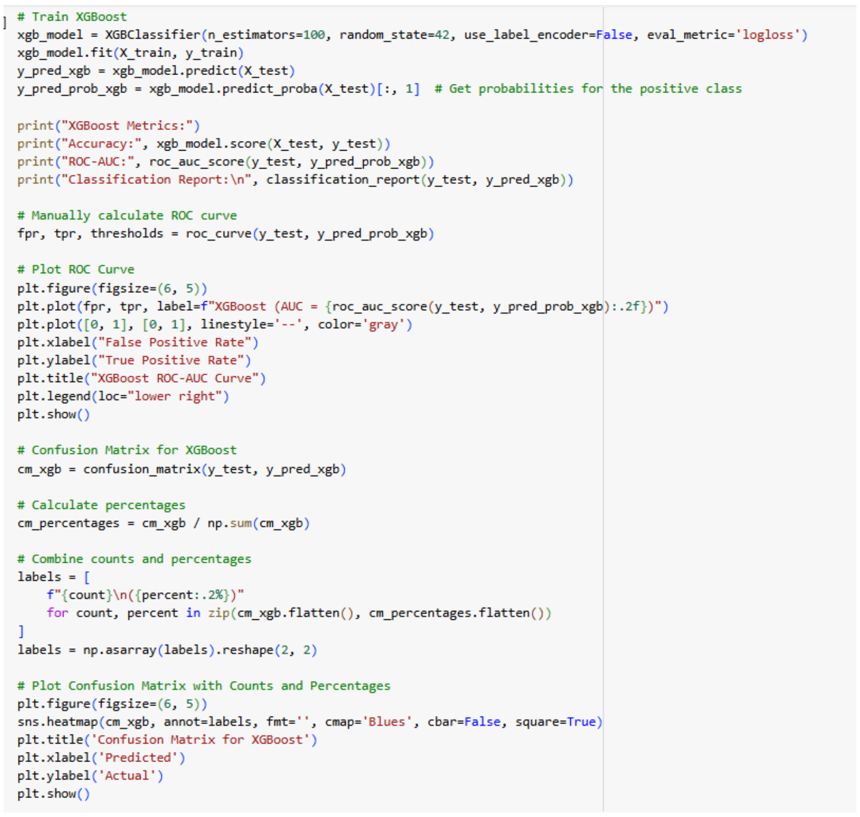
Figure 4.6 AdaBoost Confusion Matrix

Table 4.2 demonstrated AdaBoost model performance in predicting customer churn. For the non-churned class, the model achieved a precision of 82%, a recall of 89%, and an F1-score of 85%. This showed strong performance in identifying non-churned customers. For the churned class, the model recorded a lower precision of 66%, a recall of 51%, and an F1-score of 58%. It reflect challenges in detecting churned customers. Overall, the model achieved an accuracy of 78%, with a macro average F1-score of 72% and a weighted average F1-score of 77%.

Table 4.2 Classification Report of Ada Boost

	Precision	Recall	F1-Score	Support
0	0.82	0.89	0.85	999
1	0.66	0.51	0.58	407
Accuracy			0.78	1406
Macro avg	0.74	0.70	0.72	1406
Weighted avg	0.77	0.78	0.77	1406

4.3 XGBoost



```python
# Train XGBoost
xgb_model = XGBClassifier(n_estimators=100, random_state=42, use_label_encoder=False, eval_metric='logloss')
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)
y_pred_prob_xgb = xgb_model.predict_proba(X_test)[:, 1]  # Get probabilities for the positive class

print("XGBoost Metrics:")
print("Accuracy:", xgb_model.score(X_test, y_test))
print("ROC-AUC:", roc_auc_score(y_test, y_pred_prob_xgb))
print("Classification Report:\n", classification_report(y_test, y_pred_xgb))

# Manually calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_xgb)

# Plot ROC Curve
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f"XGBoost (AUC = {roc_auc_score(y_test, y_pred_prob_xgb):.2f})")
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("XGBoost ROC-AUC Curve")
plt.legend(loc="lower right")
plt.show()

# Confusion Matrix for XGBoost
cm_xgb = confusion_matrix(y_test, y_pred_xgb)

# Calculate percentages
cm_percentages = cm_xgb / np.sum(cm_xgb)

# Combine counts and percentages
labels = [
    f"{count}\n({percent:.2%})"
    for count, percent in zip(cm_xgb.flatten(), cm_percentages.flatten())
]
labels = np.asarray(labels).reshape(2, 2)

# Plot Confusion Matrix with Counts and Percentages
plt.figure(figsize=(6, 5))
sns.heatmap(cm_xgb, annot=labels, fmt='', cmap='Blues', cbar=False, square=True)
plt.title('Confusion Matrix for XGBoost')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Figure 4.7 XGBoost Code Snippet

The XGBoost model of Figure 4.7 demonstrated an accuracy of 75.89%. It indicate that it correctly classified 75.89% of the test data. Additionally, the model's ROC-AUC score of 80.09% demonstrated its effectiveness in distinguishing between churned and non-churned customers. Figure 4.8 graph showed an AUC score of 80%. This indicates that the model effectively balances the true positive rate (sensitivity) and the false positive rate

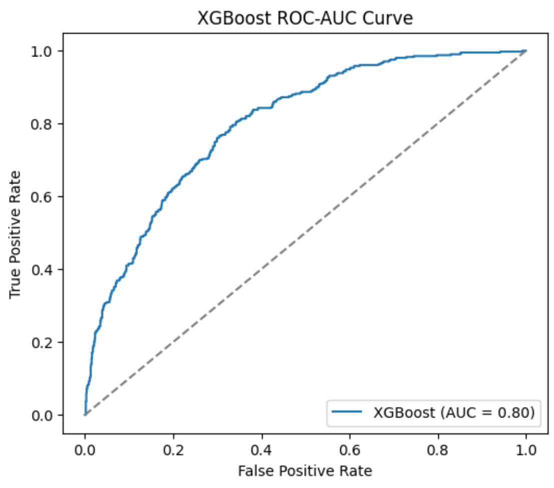


Figure 4.8 XGBoost ROC-AUC Graph

The confusion matrix for the XGBoost model in Figure 4.9 illustrated its performance in predicting customer churn. The model correctly classified 881 non-churned customers (62.66%) and 186 churned customers (13.23%). This showed its ability to identify a substantial portion of both classes. However, 118 non-churned customers (8.39%) were misclassified as churned while 221 churned customers (15.72%) were misclassified as non-churned. Overall, the model performs better at identifying non-churned customers compared to churned ones. This suggest potential areas for improvement in sensitivity toward churned customers through further optimization or rebalancing techniques.

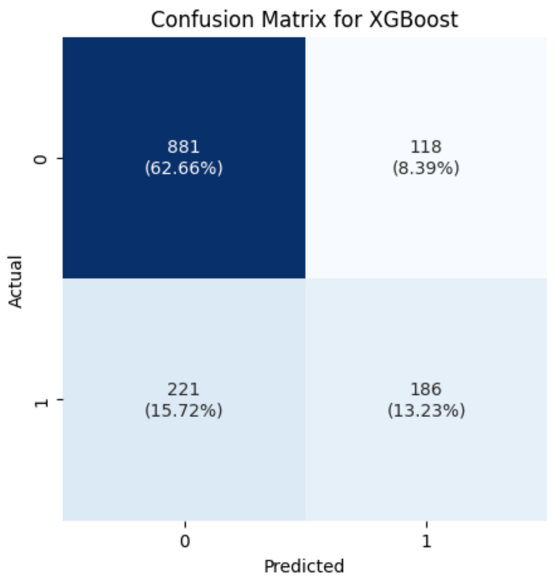


Figure 4.9 XGBoost Confusion Matrix

The classification report inTable 4.2 highlights fXGBoost performance in predicting customer churn. For the non-churned class, the model achieved a precision of 80%, a recall of 88%, and an F1-score of 84%. This demonstrated strong performance to identify non-churned customers. In contrast, the model recorded a precision of 61%, a recall of 46%, and an F1-score of 52% for the churned class. It reflected challenges in detecting churned customers. The overall accuracy of the model is 78%, with a macro-average F1-score of 68% and a weighted-average F1-score of 75%.

Table 4.3 Classification Report of XGBoost

	Precision	Recall	F1-Score	Support
0	0.80	0.88	0.84	999
1	0.61	0.46	0.52	407
Accuracy			0.78	1406
Macro avg	0.71	0.67	0.68	1406
Weighted avg	0.75	0.76	0.75	1406

4.4 Machine Learning Comparison

Table 4.4 Performance Comparison

Machine Learning	Accuracy (%)	Cross Validation (%)	ROC AUC Score (%)	F1 Churn Score (%)
Random Forest Classifier	76.53	79.48	66.38	51
AdaBoost Classifier	78.31	80.67	70.32	58
XGBoost Classifier	78.67	82.62	80.09	52

Based on the performance comparison in Table 4.4, the XGBoost Classifier emerges as the best-performing model. It achieves the highest accuracy (78.67%), cross-validation score (82.62%), and ROC AUC score (80.09%). This indicates that it is the most reliable and effective model for general prediction tasks in this dataset. However, its F1 Score for churn (52%) was slightly lower compared to AdaBoost (58%). This means that it encounters challenges to balance precision and recall for the minority class.

Next, AdaBoost Classifier delivered competitive results with an accuracy of 78.31% and a cross-validation score of 80.67%. Although its ROC-AUC score (70.32%) and F1 Score for churn (58%) were lower than XGBoost in some aspects, it demonstrated better performance in detecting churned customers due to the higher F1 score.

On the other hand, Random Forest Classifier showed slightly lower performance across the board. It achieved an accuracy of 76.53%, a cross-validation score of 79.48%, and the lowest ROC-AUC score of 66.38%. It means that the model is struggled to distinguish between classes. Its F1 Score for churn (51%) was the lowest among the three model reflect challenges in identifying churned customers effectively.

In conclusion, XGBoost emerged as the best overall classifier due to its overall performance across accuracy, cross-validation, and ROC-AUC. Its success might be due to its gradient boosting framework which minimize errors through iterative learning and is highly optimized for both accuracy and generalization. While AdaBoost could be considered for churn-specific tasks, XGBoost is the recommended model for overall predictive performance. In contrast, Random Forest performed lower and may require further optimization to improve its performance.