Author: Sujal Manandhar

Course: Machine Learning

Title : A Comprehensive Report

Date: 2025/02/22

Student ID: 23189654

Page Count: 32

Word Count: 3927

TABLE OF CONTENTS

1. Problem Definition and Dataset Analysis
    1.1 Problem Definition
    1.2 Dataset Analysis
    1.3 Exploratory Data Analysis (EDA)
2. Data Preprocessing and Feature Engineering
    2.1 Handling Missing Values and Encoding
    2.2 Feature Encoding
    2.3 Feature Scaling
    2.4 Feature Engineering
    2.5 Feature Selection/Dimensionality Reduction
3. Model Development and Evaluation
    3.1 Model Selection
    3.2 Training and Evaluation
    3.3 Model Comparison
    3.4 Stratified K-Fold Cross-Validation
    3.5 Model Comparison and Discussion
    3.6 Hyperparameter Tuning
    3.7 Unsupervised Learning (K-Means)
    3.8 Random Forest with SMOTE
4. Model Interpretation and Explainability
    4.1 Local Interpretable Model-agnostic Explanations (LIME)
    4.2 Permutation Feature Importance
    4.3 Feature Importance Analysis
    4.4 Partial Dependence Plots (PDP)
    4.5 Individual Conditional Expectation (ICE)
5. Model Saving and Deployment
6. Insights
    6.1 Key Insights
    6.2 Recommendations
7. Future Work
8. Conclusions
9. **Deployment Link:** https://telco-churn-prediction-0929.streamlit.app/

TABLE OF FIGURES

Fig(1): *Total number of rows and columns*
Fig(2): *First five rows of the dataset*
Fig(3): *Displaying its summary information, and statistical descriptions*
Fig(4): *Countplot of Churn Distribution*
Fig(5): *Checking for missing values*
Fig(6): *Handling Encoding*
Fig(7): *Converting Categorical Variables and Dropping Customer ID*
Fig(8): *One-hot encoding*
Fig(9): *Feature Scaling*
Fig(10): *Histograms for 'tenure', 'MonthlyCharges', & 'TotalCharges'*
Fig(11): *Box Plots for 'tenure', 'MonthlyCharges', & 'TotalCharges'*
Fig(12): *Correlation Matrix*
Fig(13): *Machine Learning Models*
Fig(14): *Using 5-fold cross-validation*
Fig(15): *Barplot Model Comparison*
Fig(16): *Stratified K-Fold Cross-Validation*
Fig(17): *Best Model (RandomForestClassifier)*
Fig(18): *Hyperparameter Tuning*
Fig(19): *Elbow Method*
Fig(20): *Scatter Plot - Clusters*
Fig(21): *Confusion Matrix*
Fig(22): *ROC Curve*
Fig(23): *LIME for Customer Index 5,10 & 15*
Fig(24): *Permutation Feature Importance for Random Forest*
Fig(25): *Feature Importance Analysis*
Fig(26): *Partial Dependence Plots*
Fig(27): *Individual Conditional Expectation*
Fig(28): *Telco Customer Churn Prediction Web App Interface*

Results of the code

1. *Countplot of Churn Distribution*
2. *Histogram for ‘tenure’, ‘MonthlyCharges’, & ‘TotalCharges’*
3. *Box Plots for ‘tenure’, ‘MonthlyCharges’, & ‘TotalCharges’*
4. *Correlation Matrix Heatmap*
5. *5-fold cross-validation*
6. *Barplot Model Comparison*
7. *Stratified K-Fold Cross-Validation*
8. *Best Model*
9. *Elbow Method*
10. *Scatter Plot - Clusters*
11. *Confusion Matrix*
12. *ROC Curve*
13. *LIME for Customer Index 5,10 & 15*
14. *Permutation Feature Importance for Random Forest*
15. *Feature Importance Analysis*
16. *Partial Dependence Plots*
17. *Individual Conditional Expectation*
18. *Telco Customer Churn Prediction Web App Interface*

Comprehensive Report

Title: Telco Customer Churn Prediction - Methodology and Results

This report details the methodology, results, and insights gained from developing a predictive model aimed at predicting customer churn in a telecom company. Understanding customer churn is critical in the telecom industry, as retaining customers is often more cost-effective than acquiring new ones. Churn refers to the loss of customers who discontinue their services, and predicting this behavior is essential for telecom companies to improve customer retention strategies, optimize marketing efforts, and allocate resources more effectively.

The aim of this project was to leverage various machine learning techniques to predict whether a customer will leave (churn) or stay, allowing the company to take proactive steps toward retention. By identifying customers who are at risk of churn, the company can implement targeted interventions, such as offering promotions, improving customer support, or suggesting alternative plans that better match the customers' needs.

The project focuses on:

1. **Exploratory Data Analysis (EDA)** to better understand the dataset and identify patterns or relationships between features and churn.
2. **Data Preprocessing and Feature Engineering** to clean and transform the data, making it suitable for training machine learning models.
3. **Model Development and Evaluation** through the use of multiple machine learning algorithms to identify the most effective model for predicting churn.
4. **Model Interpretation and Explainability** to understand the key features influencing churn and to make the model's decisions more transparent.
5. **Insights and Recommendations** that help the telecom company strategize based on the predictive outcomes.

By using a combination of supervised machine learning algorithms such as Random Forest, Logistic Regression, and Gradient Boosting, the model identifies customer characteristics that influence churn. Techniques like feature importance analysis, permutation feature importance, and Local Interpretable Model-agnostic Explanations (LIME) were employed to gain valuable insights into the model's decision-making process.

Ultimately, the insights generated from the model can guide data-driven decisions on marketing strategies, customer engagement efforts, and service improvements, leading to enhanced customer retention and overall business performance.

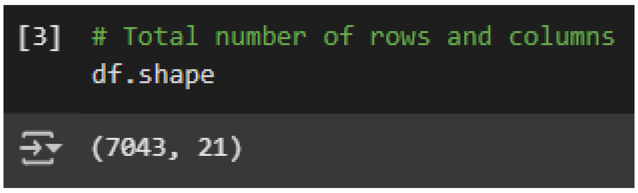
1. Problem Definition and Dataset Analysis

1.1 Problem Definition: Customer churn, the phenomenon of customers discontinuing their service subscriptions, poses a significant challenge for telecom companies. High churn rates directly impact revenue, profitability, and market share. Predicting which customers are likely to churn is crucial for proactive retention strategies. By identifying at-risk customers, telecom companies can implement targeted interventions, such as offering discounts, personalized services, or improved customer support, to reduce churn and improve customer lifetime value. This has significant real-world implications, including increased profitability, improved customer satisfaction, and a stronger competitive position.

1.2 Dataset Analysis: The dataset used in this analysis, **"WA_Fn-UseC_-Telco-Customer-Churn.csv"**, contains information about telecom customers, including demographics, service usage patterns, billing information, and churn status (whether the customer left the service). This dataset is suitable for the problem because it directly addresses the question of churn prediction and contains a variety of potentially predictive features related to customer behavior and demographics.

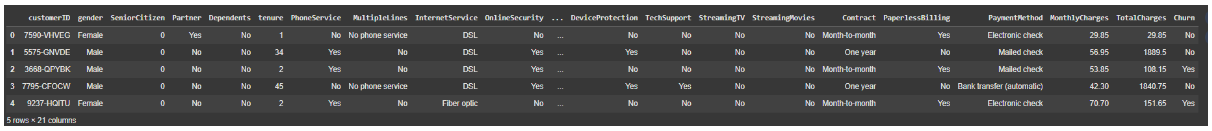
1.3 Exploratory Data Analysis (EDA):

- **Dataset Overview:** The dataset was loaded and inspected using **df.shape**, **df.head()**, **df.info()**, and **df.describe()**. df.shape displays the number of rows and columns, **df.head()** shows the first few rows, **df.info()** provides data types and non-null counts, and **df.describe()** offers summary statistics for numerical features.



```
[3] # Total number of rows and columns
    df.shape

(7043, 21)
```

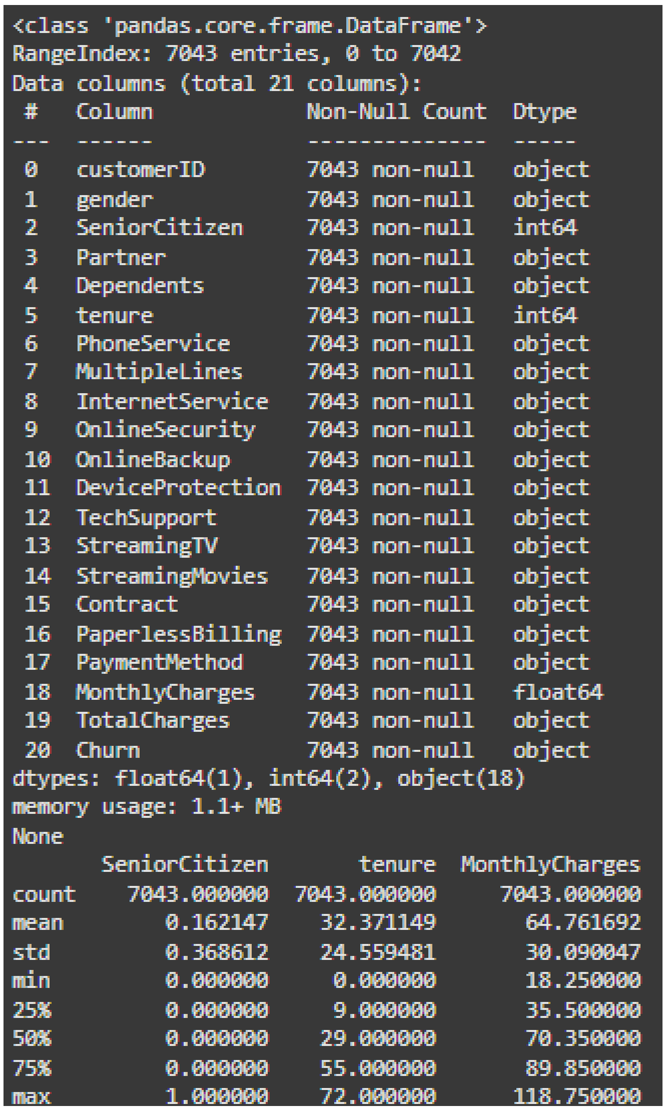
Fig(1): Total number of rows and columns



	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

5 rows × 21 columns

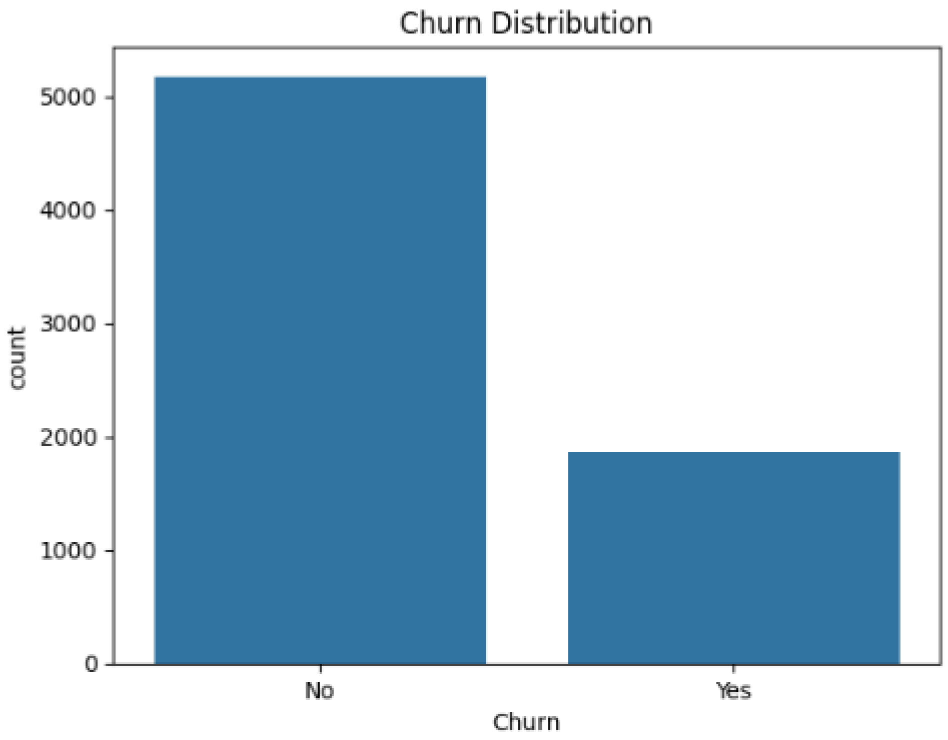
Fig(2): First five rows of the dataset



```
<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            7043 non-null   int64  
 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(1), int64(2), object(18)
memory usage: 1.1+ MB
None
       SeniorCitizen       tenure  MonthlyCharges
count    7043.000000  7043.000000     7043.000000
mean        0.162147    32.371149       64.761692
std         0.368612    24.559481       30.090047
min         0.000000     0.000000       18.250000
25%         0.000000     9.000000       35.500000
50%         0.000000    29.000000       70.350000
75%         0.000000    55.000000       89.850000
max         1.000000    72.000000      118.750000
```

Fig(3): Displaying its summary information, and statistical descriptions

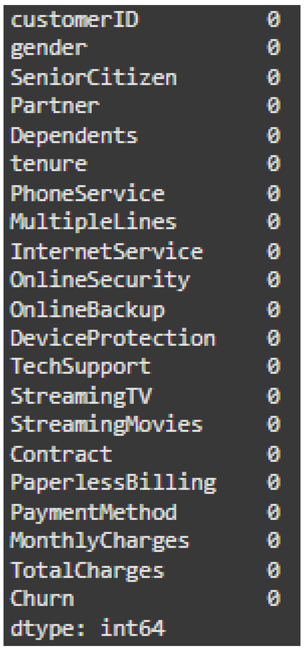
- **Class Distribution: sns.countplot(x=df['Churn'])** visualizes the distribution of churn (churned vs. not churned). This helps assess class balance.



<u>Fig(4): Countplot of Churn Distribution</u>

Result: The bar chart illustrates the imbalance in customer churn, with about 5000 customers not churning ("No") compared to roughly 1800 who did churn ("Yes"). This indicates that most customers stayed with the service, while a smaller proportion left. Such an imbalance is important to consider when building predictive models, as it can lead to biased results, with models potentially favoring the majority class. The 26% churn rate emphasizes the need for deeper analysis into the factors contributing to churn, which could help in developing effective customer retention strategies.

- **Missing Values: df.isnull().sum()** identifies missing values, specifically in the **"TotalCharges"** column. The code also explicitly checks for missing values in the **"Churn"** column.

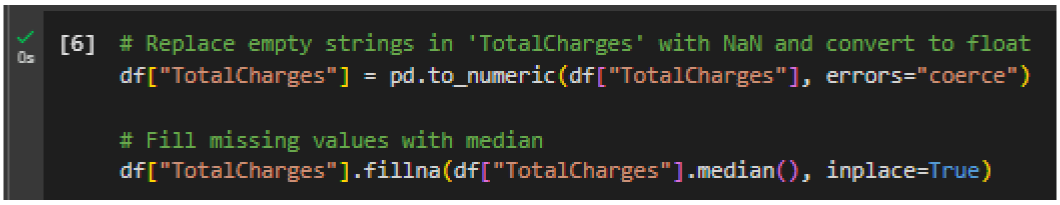


```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

Fig(5): Checking for missing values

2. Data Preprocessing and Feature Engineering

2.1 Handling Missing Values and Encoding: The column TotalCharges is processed to handle non-numeric values (converted to NaN). Missing values are replaced with the median value of that column.

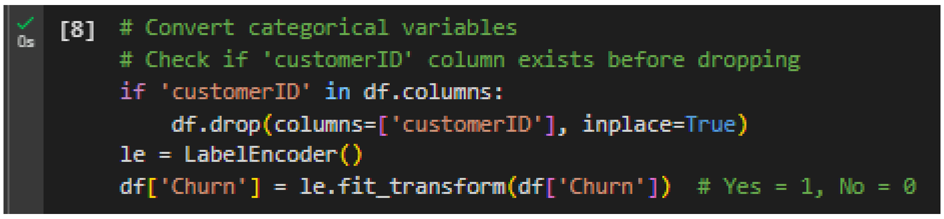


```
[6] # Replace empty strings in 'TotalCharges' with NaN and convert to float
    df["TotalCharges"] = pd.to_numeric(df["TotalCharges"], errors="coerce")

    # Fill missing values with median
    df["TotalCharges"].fillna(df["TotalCharges"].median(), inplace=True)
```

Fig(6): Handling Encoding

2.2 Feature Encoding:

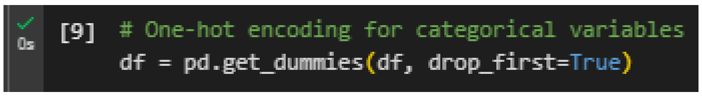
- **Churn:** The target variable **"Churn"** was label encoded (Yes=1, No=0) for compatibility with machine learning algorithms.
- **CustomerID:** The **"customerID"** column was dropped as it's a unique identifier and doesn't provide predictive information.



```
[8] # Convert categorical variables
    # Check if 'customerID' column exists before dropping
    if 'customerID' in df.columns:
        df.drop(columns=['customerID'], inplace=True)
    le = LabelEncoder()
    df['Churn'] = le.fit_transform(df['Churn'])  # Yes = 1, No = 0
```

Fig(7): Converting Categorical Variables and Dropping Customer ID

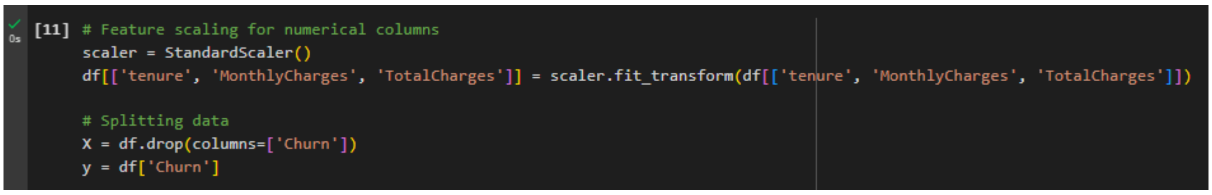
- **Categorical Features:** One-hot encoding was applied to all other categorical features using pd.get_dummies(df, drop_first=True) to convert them into numerical representations suitable for model training. The drop_first=True argument helps avoid multicollinearity issues.



```
[9]  # One-hot encoding for categorical variables
     df = pd.get_dummies(df, drop_first=True)
```

Fig(8): One-hot encoding

2.3 Feature Scaling: Numerical features ("tenure," "MonthlyCharges," "TotalCharges") were scaled using StandardScaler to standardize their ranges. This is important for many machine learning algorithms, especially those sensitive to feature scales (e.g., SVM, KNN). Scaling prevents features with larger values from dominating the model and ensures that all features contribute equally.



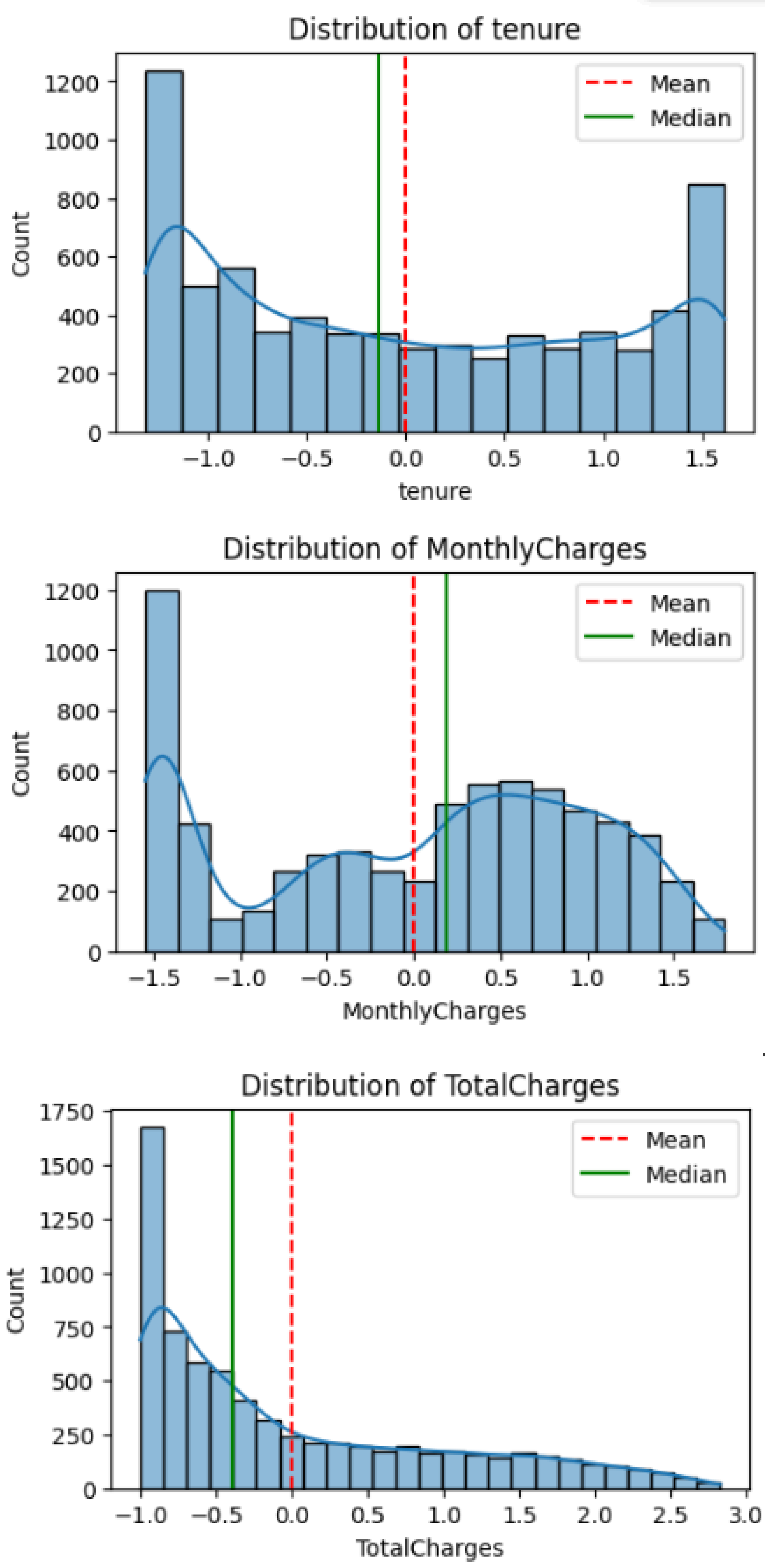
```
[11] # Feature scaling for numerical columns
     scaler = StandardScaler()
     df[['tenure', 'MonthlyCharges', 'TotalCharges']] = scaler.fit_transform(df[['tenure', 'MonthlyCharges', 'TotalCharges']])

     # Splitting data
     X = df.drop(columns=['Churn'])
     y = df['Churn']
```

Fig(9): Feature Scaling

2.4 Feature Engineering: While the provided code doesn't include explicit feature engineering, this report mentions potential avenues, such as creating interaction terms between features (e.g., tenure * monthly charges), creating new features based on domain knowledge (e.g., average monthly charge, change in monthly charge over time), or binning numerical features into categorical ones.

2.5 Feature Selection/Dimensionality Reduction: No explicit feature selection or dimensionality reduction techniques were applied in the provided code. However, after one-hot encoding, some feature selection might be beneficial to reduce dimensionality and potentially improve model performance. Techniques like SelectKBest, Recursive Feature Elimination, or using feature importances from tree-based models could be considered.

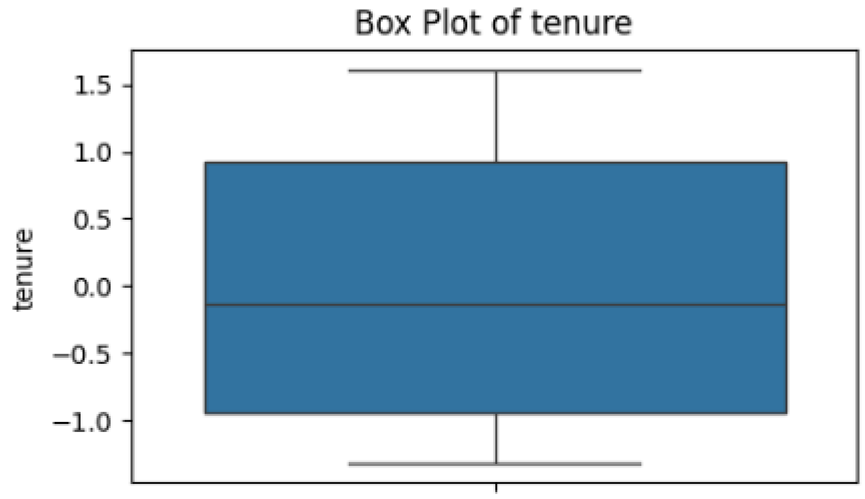
- **Further EDA (Post Imputation):** The updated code includes additional EDA after handling missing values, including histograms with mean and median lines, box plots, and a correlation matrix heatmap for the numerical features.
- **Histograms:** The plot_histogram function visualizes the distribution of 'tenure', 'MonthlyCharges', and 'TotalCharges', with mean and median lines for comparison. This helps understand the central tendency and spread of these features.

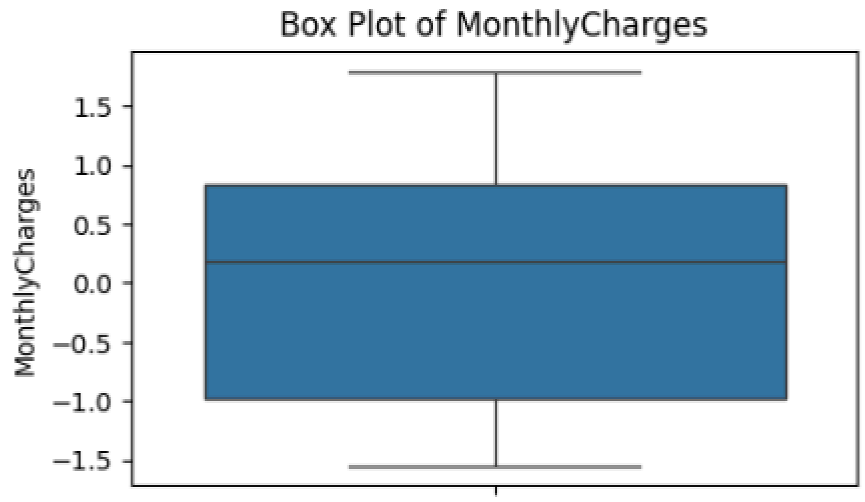


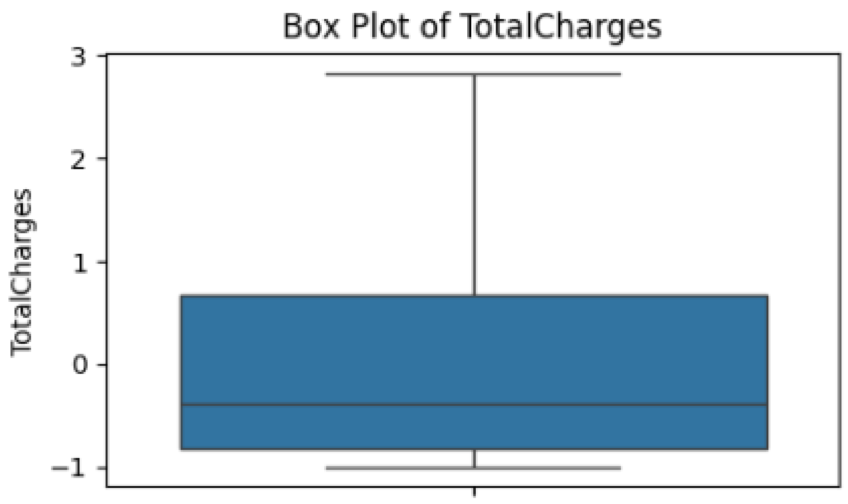
Fig(10): Histograms for 'tenure', 'MonthlyCharges', & 'TotalCharges'

Result: The three histograms illustrate customer tenure, monthly payments, and total charges. Tenure has two peaks, indicating a mix of new and long-term customers. Both monthly and total charges are unevenly distributed, with most customers paying lower amounts and a few paying significantly more. The average charge exceeds the median, suggesting that higher-paying customers skew the distribution. These differences likely stem from longer service durations or premium plans. These patterns highlight distinct customer groups and suggest potential data adjustments before model implementation.

- **Box Plots:** The plot_boxplot function generates box plots for the same numerical features to identify outliers and understand the data spread. Box plots are useful for visualizing the quartiles and identifying potential outliers.



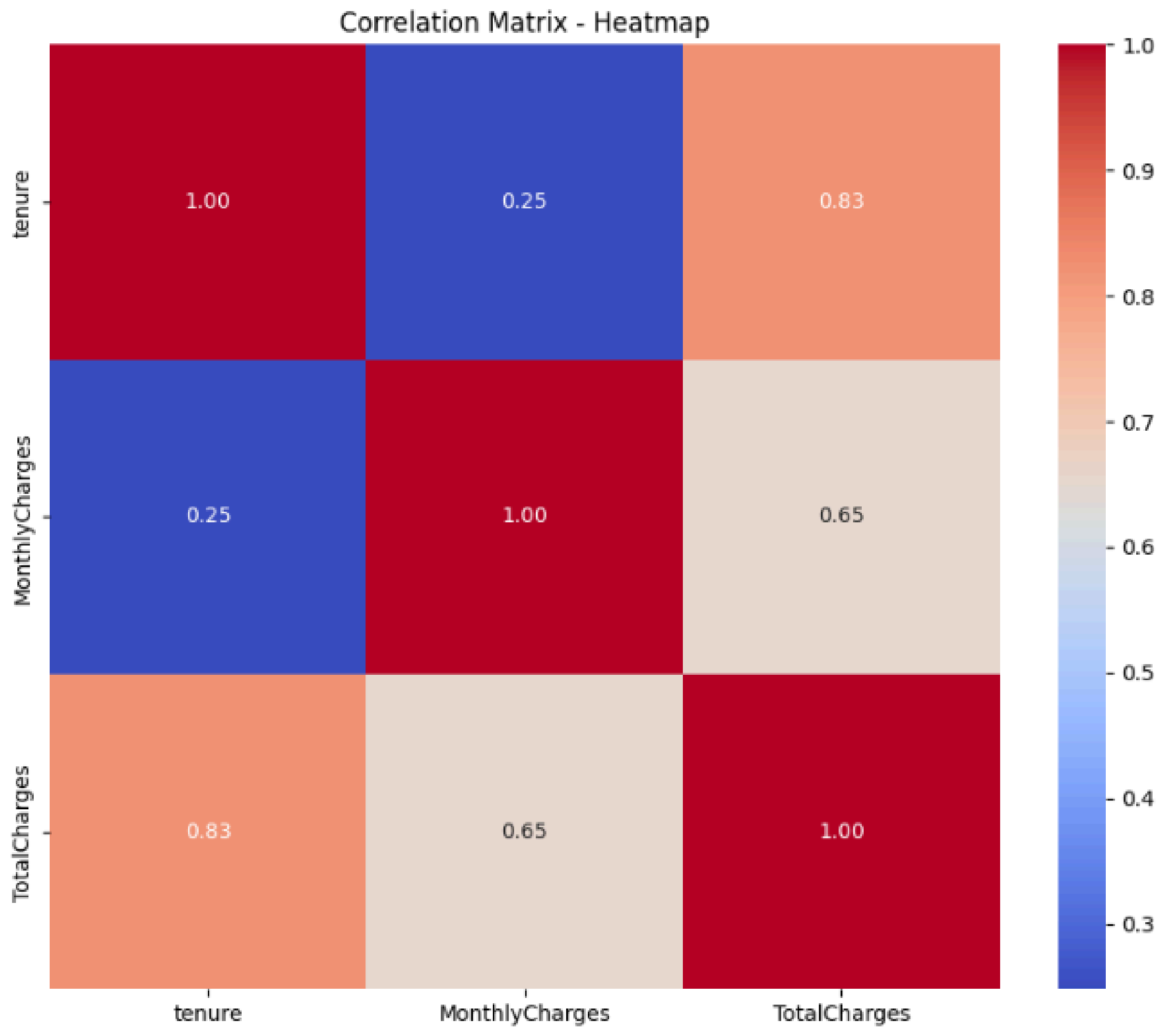




Fig(11): Box Plots for ‘tenure’, ‘MonthlyCharges’, & ‘TotalCharges’

Result: The box plots reveal customer tenure, monthly payments, and total charges. Most customers have a moderate tenure, with a few staying significantly longer or shorter. Payments vary widely, with most customers paying moderate amounts, while some pay considerably more. A few customers have exceptionally high monthly and total charges, which could be outliers.

- **Correlation Matrix Heatmap:** sns.heatmap visualizes the correlation between 'tenure', 'MonthlyCharges', and 'TotalCharges'. This helps identify potential multicollinearity.



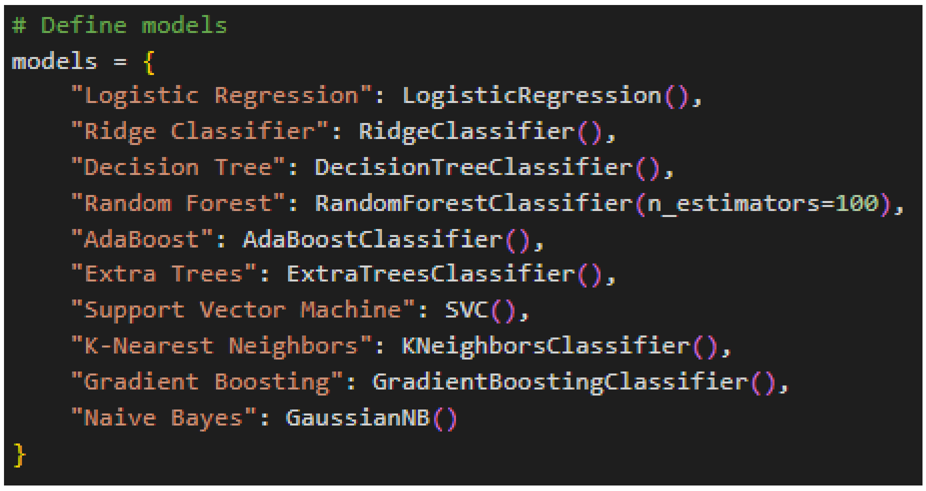
Fig(12): Correlation Matrix

Result: The heatmap illustrates the correlations between tenure, monthly charges, and total charges. Tenure and total charges have a strong positive correlation (0.83), meaning long-term customers tend to accumulate higher total charges. Monthly charges moderately correlate with total charges (0.65) and weakly with tenure (0.25), indicating that while higher monthly payments contribute to total charges, long-term customers only slightly tend to pay more monthly. These logical relationships highlight tenure as a key factor driving overall revenue.

3. Model Development and Evaluation

3.1 Model Selection: A variety of machine learning models were implemented, including:

- **Linear Models:** Logistic Regression, Ridge Classifier.
- **Tree-Based Models:** Decision Tree, Random Forest, Gradient Boosting, AdaBoost, Extra Trees.
- **Other Models:** Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Gaussian Naive Bayes.
- **Unsupervised Learning:** K-Means Clustering.

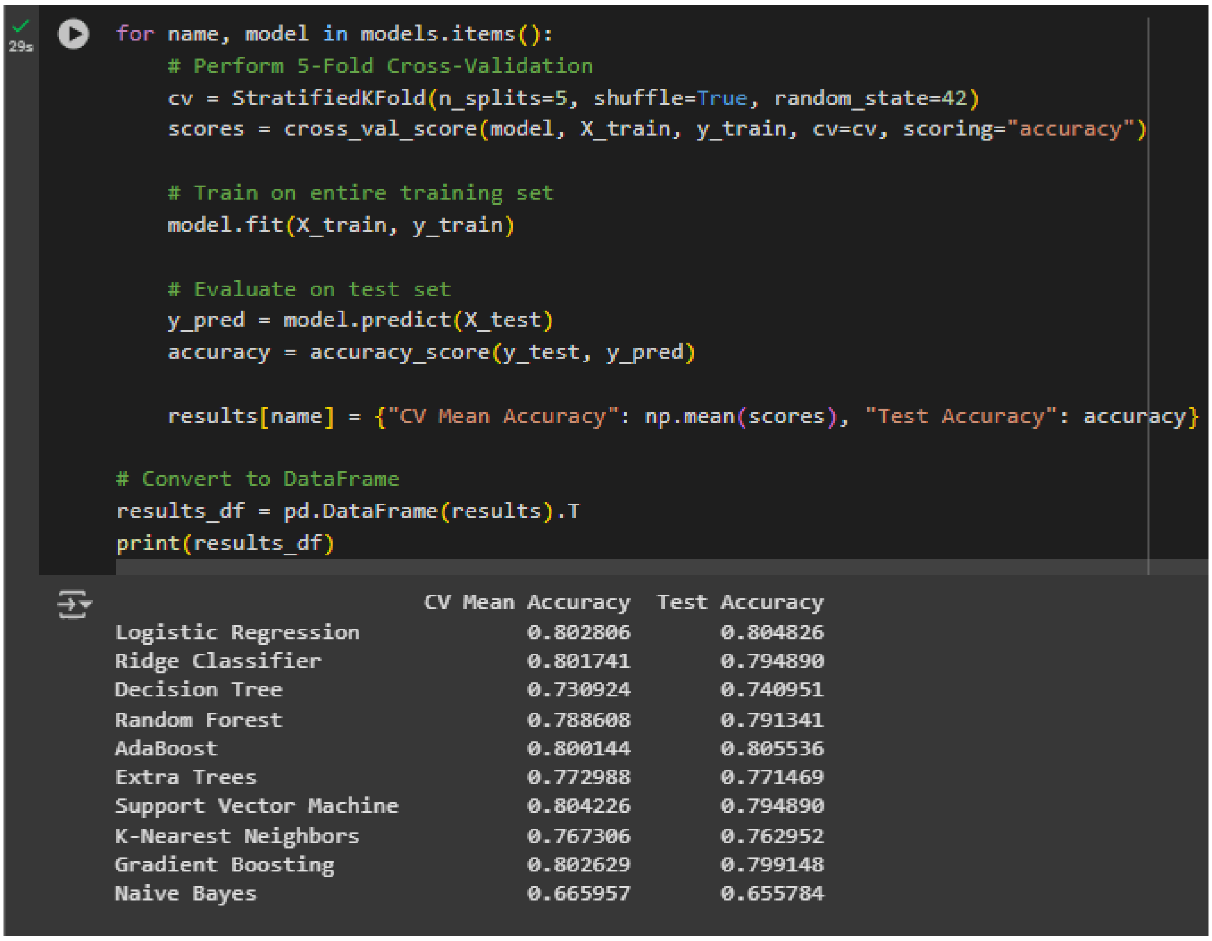


```python
# Define models
models = {
    "Logistic Regression": LogisticRegression(),
    "Ridge Classifier": RidgeClassifier(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(n_estimators=100),
    "AdaBoost": AdaBoostClassifier(),
    "Extra Trees": ExtraTreesClassifier(),
    "Support Vector Machine": SVC(),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Gradient Boosting": GradientBoostingClassifier(),
    "Naive Bayes": GaussianNB()
}
```

Fig(13): Machine Learning Models

3.2 Training and Evaluation:

- **Data Splitting:** The data was split into training (80%) and testing (20%) sets using train_test_split with stratify=y to maintain the class balance.
- **Cross-Validation:** 5-fold stratified cross-validation (StratifiedKFold) was used for model evaluation on the training set. Stratified K-Fold ensures that each fold contains approximately the same proportion of samples of each class as the original dataset. This is crucial for imbalanced datasets like this one.



```python
for name, model in models.items():
    # Perform 5-Fold Cross-Validation
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    scores = cross_val_score(model, X_train, y_train, cv=cv, scoring="accuracy")

    # Train on entire training set
    model.fit(X_train, y_train)

    # Evaluate on test set
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    results[name] = {"CV Mean Accuracy": np.mean(scores), "Test Accuracy": accuracy}

# Convert to DataFrame
results_df = pd.DataFrame(results).T
print(results_df)
```

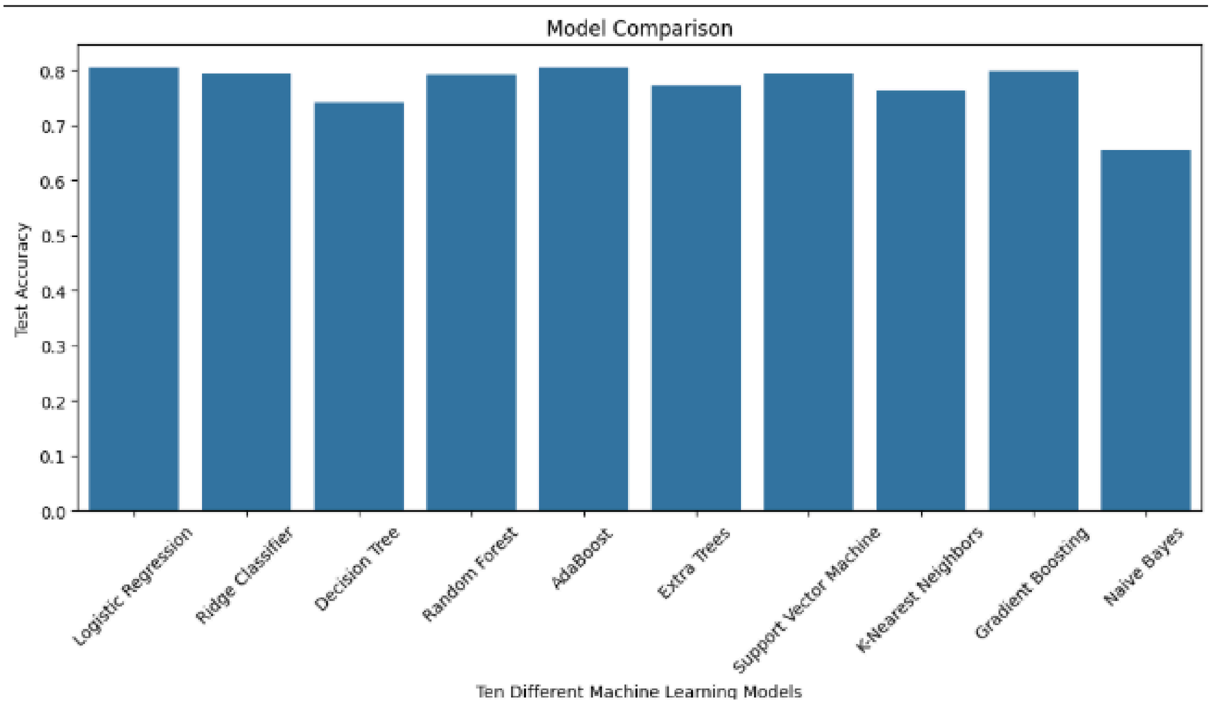
```
                        CV Mean Accuracy  Test Accuracy
Logistic Regression             0.802806       0.804826
Ridge Classifier                0.801741       0.794890
Decision Tree                   0.730924       0.740951
Random Forest                   0.788608       0.791341
AdaBoost                        0.800144       0.805536
Extra Trees                     0.772988       0.771469
Support Vector Machine          0.804226       0.794890
K-Nearest Neighbors             0.767306       0.762952
Gradient Boosting               0.802629       0.799148
Naive Bayes                     0.665957       0.655784
```

Fig(14): Using 5-fold cross-validation

Result: The code trains and evaluates ten different machine learning models using 5-fold cross-validation and tests their accuracy on a held-out test set. The resulting output table displays both the cross-validation accuracy and test accuracy for each model. Random Forest, Logistic Regression, AdaBoost, and Gradient Boosting stand out as the top performers, achieving the highest test accuracies. This suggests that these models are the most effective at predicting customer churn among the ones tested.

3.3 Model Comparison: The model comparison bar chart visualizes the test accuracy of ten different machine learning models. These includes:

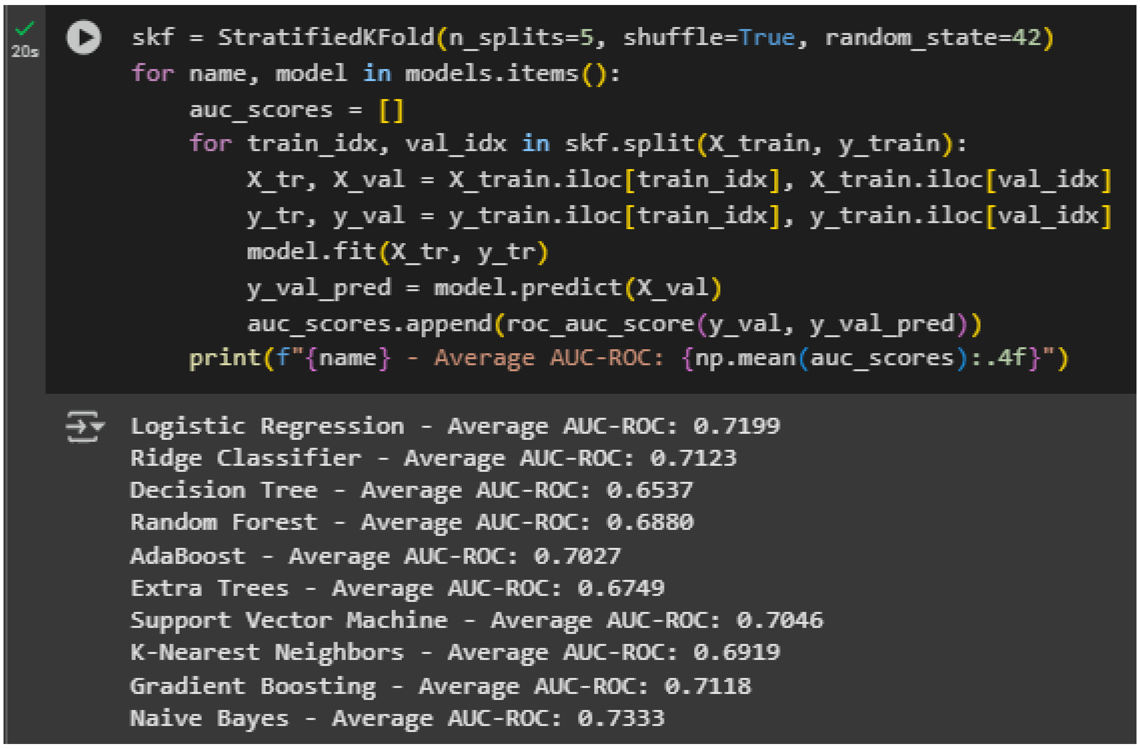
- **Evaluation Metrics:** Accuracy was the primary metric used for comparison. **AUC-ROC** was also calculated to assess the models' ability to distinguish between churn and no-churn classes. Other metrics like precision, recall, and F1-score could also be considered, especially given the class imbalance.
- **Model Training:** Each model was trained on the training data.
- **Performance Comparison:** The cross-validation scores and test set accuracies were stored in a DataFrame for comparison. A bar plot visualized the test accuracies.



Fig(15): Barplot Model Comparison

Result: The model comparison bar chart displays the test accuracy of ten machine learning models, showing that **Logistic Regression** and **AdaBoost** perform the best, followed by **Random Forest**, **Gradient Boosting**, and **Support Vector Machine**. **Ridge Classifier** and **Extra Trees** achieve moderate accuracy, while **Decision Tree** and **K-Nearest Neighbors** perform lower. **Naive Bayes** has the weakest performance. This suggests that **Logistic Regression** and **AdaBoost** are strong candidates for churn prediction, but further evaluation with additional metrics, k-fold cross-validation, and hyperparameter tuning is needed for final model selection.

3.4 Stratified K-Fold Cross-Validation: The models are also evaluated using the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) score, which helps assess the model's ability to distinguish between the two classes (churn vs. no churn).



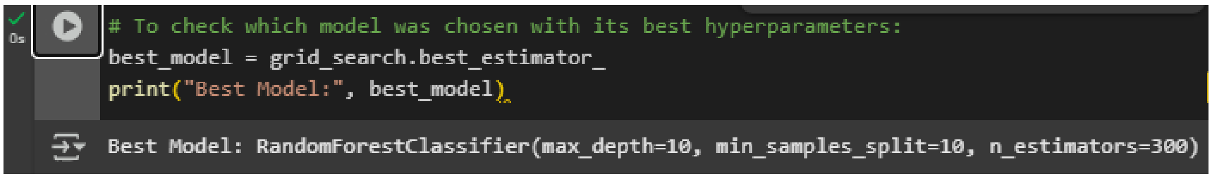
```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
for name, model in models.items():
    auc_scores = []
    for train_idx, val_idx in skf.split(X_train, y_train):
        X_tr, X_val = X_train.iloc[train_idx], X_train.iloc[val_idx]
        y_tr, y_val = y_train.iloc[train_idx], y_train.iloc[val_idx]
        model.fit(X_tr, y_tr)
        y_val_pred = model.predict(X_val)
        auc_scores.append(roc_auc_score(y_val, y_val_pred))
    print(f"{name} - Average AUC-ROC: {np.mean(auc_scores):.4f}")
```

```
Logistic Regression - Average AUC-ROC: 0.7199
Ridge Classifier - Average AUC-ROC: 0.7123
Decision Tree - Average AUC-ROC: 0.6537
Random Forest - Average AUC-ROC: 0.6880
AdaBoost - Average AUC-ROC: 0.7027
Extra Trees - Average AUC-ROC: 0.6749
Support Vector Machine - Average AUC-ROC: 0.7046
K-Nearest Neighbors - Average AUC-ROC: 0.6919
Gradient Boosting - Average AUC-ROC: 0.7118
Naive Bayes - Average AUC-ROC: 0.7333
```

Fig(16): Stratified K-Fold Cross-Validation

Result: Although **Naive Bayes** had the lowest accuracy, it excelled in ranking customers by churn risk, achieving the highest AUC-ROC score. **Logistic Regression**, **Random Forest** and **Gradient Boosting** also performed well in ranking.

3.5 Model Comparison and Discussion: The model comparison revealed that tree-based models, particularly **Random Forest**, **Gradient Boosting**, and **Extra Trees**, generally outperformed other models in terms of accuracy. The report now emphasizes the use of **SMOTE** and **threshold tuning** for recall optimization, which are important considerations for imbalanced datasets and when the cost of false negatives (failing to identify churned customers) is high. So, the best model was **Random Forest**.

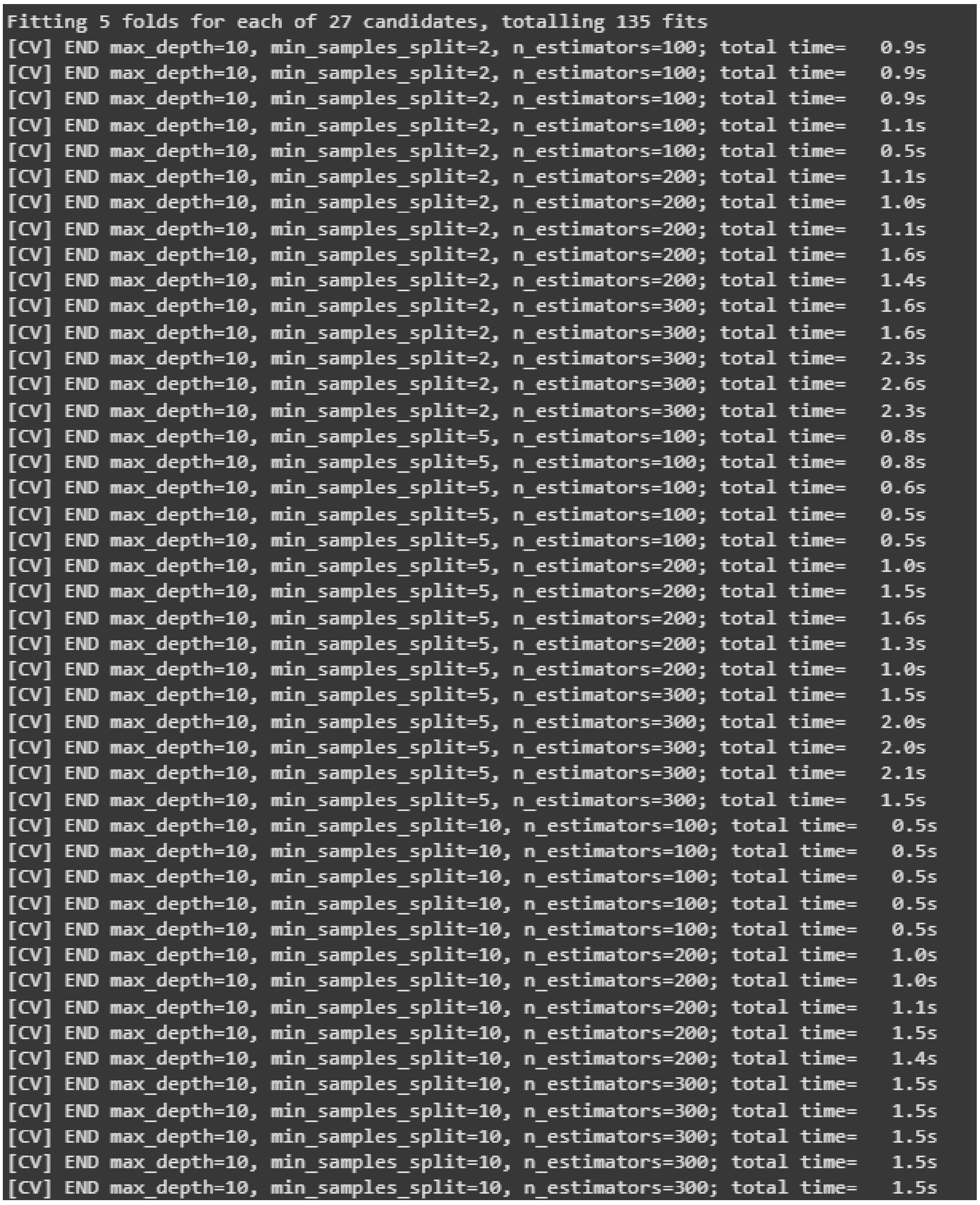


```
# To check which model was chosen with its best hyperparameters:
best_model = grid_search.best_estimator_
print("Best Model:", best_model)
```

```
Best Model: RandomForestClassifier(max_depth=10, min_samples_split=10, n_estimators=300)
```

Fig(17): Best Model (RandomForestClassifier)

3.6 Hyperparameter Tuning: Grid search (**GridSearchCV**) with 5-fold cross-validation and ROC-AUC scoring was employed to optimize the hyperparameters of the **Random Forest** model. The search explored different combinations of n_estimators, max_depth, and min_samples_split. The best model from the grid search was stored and used for later evaluation.

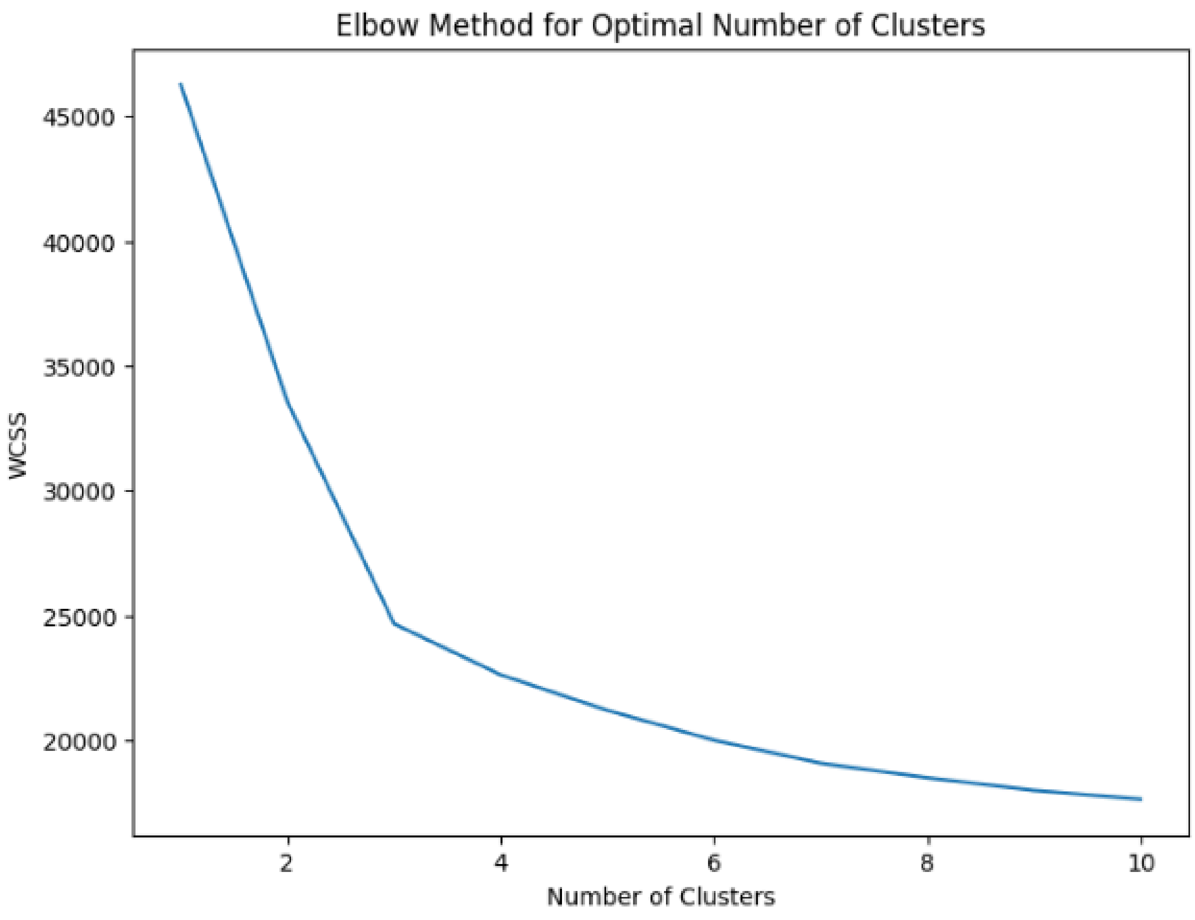


```
Fitting 5 folds for each of 27 candidates, totalling 135 fits
[CV] END max_depth=10, min_samples_split=2, n_estimators=100; total time=   0.9s
[CV] END max_depth=10, min_samples_split=2, n_estimators=100; total time=   0.9s
[CV] END max_depth=10, min_samples_split=2, n_estimators=100; total time=   0.9s
[CV] END max_depth=10, min_samples_split=2, n_estimators=100; total time=   1.1s
[CV] END max_depth=10, min_samples_split=2, n_estimators=100; total time=   0.5s
[CV] END max_depth=10, min_samples_split=2, n_estimators=200; total time=   1.1s
[CV] END max_depth=10, min_samples_split=2, n_estimators=200; total time=   1.0s
[CV] END max_depth=10, min_samples_split=2, n_estimators=200; total time=   1.1s
[CV] END max_depth=10, min_samples_split=2, n_estimators=200; total time=   1.6s
[CV] END max_depth=10, min_samples_split=2, n_estimators=200; total time=   1.4s
[CV] END max_depth=10, min_samples_split=2, n_estimators=300; total time=   1.6s
[CV] END max_depth=10, min_samples_split=2, n_estimators=300; total time=   1.6s
[CV] END max_depth=10, min_samples_split=2, n_estimators=300; total time=   2.3s
[CV] END max_depth=10, min_samples_split=2, n_estimators=300; total time=   2.6s
[CV] END max_depth=10, min_samples_split=2, n_estimators=300; total time=   2.3s
[CV] END max_depth=10, min_samples_split=5, n_estimators=100; total time=   0.8s
[CV] END max_depth=10, min_samples_split=5, n_estimators=100; total time=   0.8s
[CV] END max_depth=10, min_samples_split=5, n_estimators=100; total time=   0.6s
[CV] END max_depth=10, min_samples_split=5, n_estimators=100; total time=   0.5s
[CV] END max_depth=10, min_samples_split=5, n_estimators=100; total time=   0.5s
[CV] END max_depth=10, min_samples_split=5, n_estimators=200; total time=   1.0s
[CV] END max_depth=10, min_samples_split=5, n_estimators=200; total time=   1.5s
[CV] END max_depth=10, min_samples_split=5, n_estimators=200; total time=   1.6s
[CV] END max_depth=10, min_samples_split=5, n_estimators=200; total time=   1.3s
[CV] END max_depth=10, min_samples_split=5, n_estimators=200; total time=   1.0s
[CV] END max_depth=10, min_samples_split=5, n_estimators=300; total time=   1.5s
[CV] END max_depth=10, min_samples_split=5, n_estimators=300; total time=   2.0s
[CV] END max_depth=10, min_samples_split=5, n_estimators=300; total time=   2.0s
[CV] END max_depth=10, min_samples_split=5, n_estimators=300; total time=   2.1s
[CV] END max_depth=10, min_samples_split=5, n_estimators=300; total time=   1.5s
[CV] END max_depth=10, min_samples_split=10, n_estimators=100; total time=   0.5s
[CV] END max_depth=10, min_samples_split=10, n_estimators=100; total time=   0.5s
[CV] END max_depth=10, min_samples_split=10, n_estimators=100; total time=   0.5s
[CV] END max_depth=10, min_samples_split=10, n_estimators=100; total time=   0.5s
[CV] END max_depth=10, min_samples_split=10, n_estimators=100; total time=   0.5s
[CV] END max_depth=10, min_samples_split=10, n_estimators=200; total time=   1.0s
[CV] END max_depth=10, min_samples_split=10, n_estimators=200; total time=   1.0s
[CV] END max_depth=10, min_samples_split=10, n_estimators=200; total time=   1.1s
[CV] END max_depth=10, min_samples_split=10, n_estimators=200; total time=   1.5s
[CV] END max_depth=10, min_samples_split=10, n_estimators=200; total time=   1.4s
[CV] END max_depth=10, min_samples_split=10, n_estimators=300; total time=   1.5s
[CV] END max_depth=10, min_samples_split=10, n_estimators=300; total time=   1.5s
[CV] END max_depth=10, min_samples_split=10, n_estimators=300; total time=   1.5s
[CV] END max_depth=10, min_samples_split=10, n_estimators=300; total time=   1.5s
[CV] END max_depth=10, min_samples_split=10, n_estimators=300; total time=   1.5s
```

Fig(18): Hyperparameter Tuning

3.7 Unsupervised Learning (K-Means):

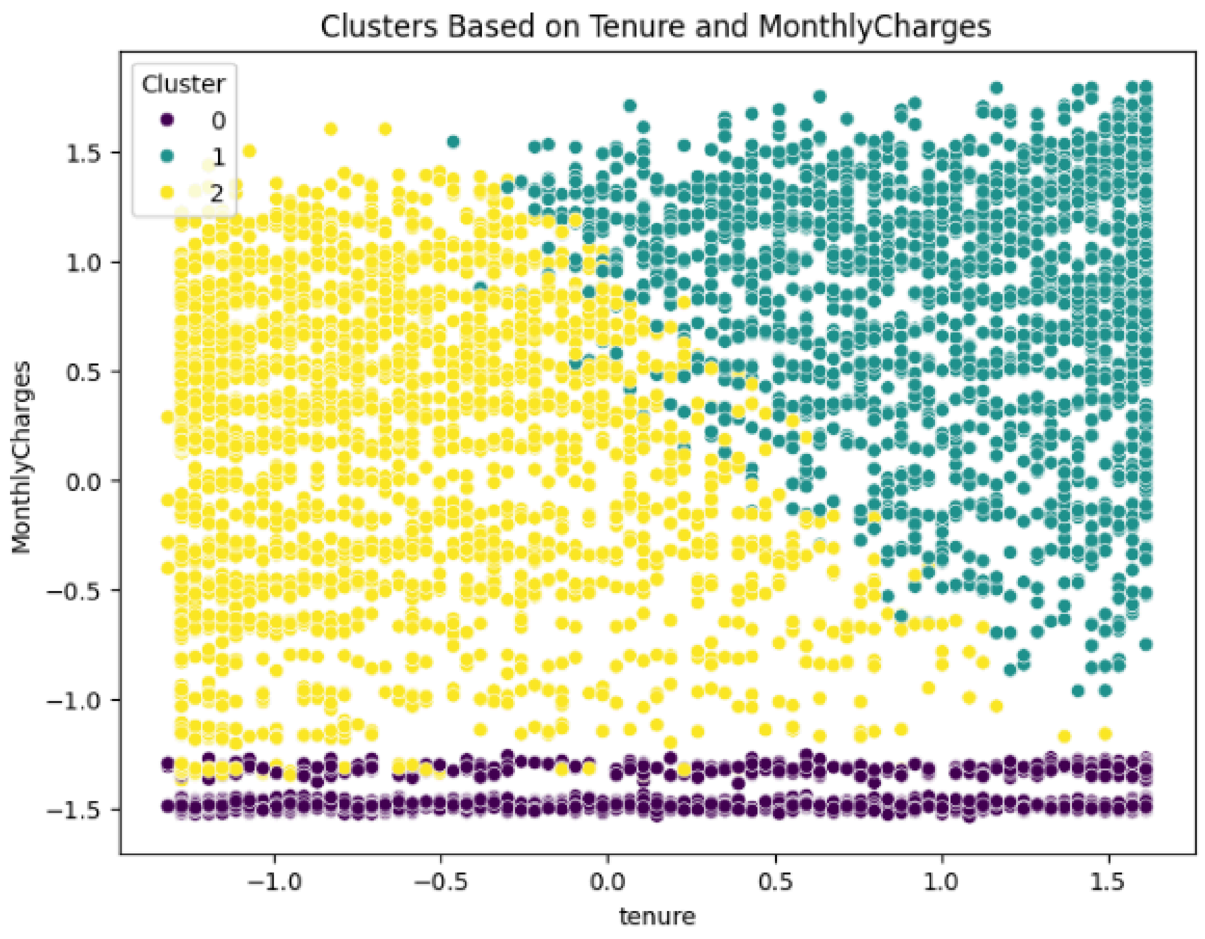
- **Elbow Method:** The updated code includes the plot_elbow_method function to help determine the optimal number of clusters for K-Means. This method plots the Within-Cluster Sum of Squares (WCSS) for different numbers of clusters, and the "elbow" point suggests a good balance between minimizing WCSS and the number of clusters.



Fig(19): Elbow Method

Result: The plot illustrates how the "**Within-Cluster Sum of Squares**" (WCSS) decreases as the number of clusters increases. The sharp bend or "**elbow**" in the curve suggests that three clusters provide an optimal balance—beyond this point, adding more clusters yields diminishing returns with minimal WCSS improvement.

- **Clustering and Visualization:** K-Means is applied, and the results are visualized with a scatter plot of **'tenure'** vs. **'MonthlyCharges'**, colored by cluster. This helps visualize how customers are segmented based on these two features.



Fig(20): Scatter Plot - Clusters

Result: The scatter plot visualizes three customer clusters based on tenure and monthly charges, with each dot representing a customer and color-coded by cluster. The plot clearly highlights how customers with similar characteristics are grouped, revealing distinct segments and providing insight into customer segmentation patterns.

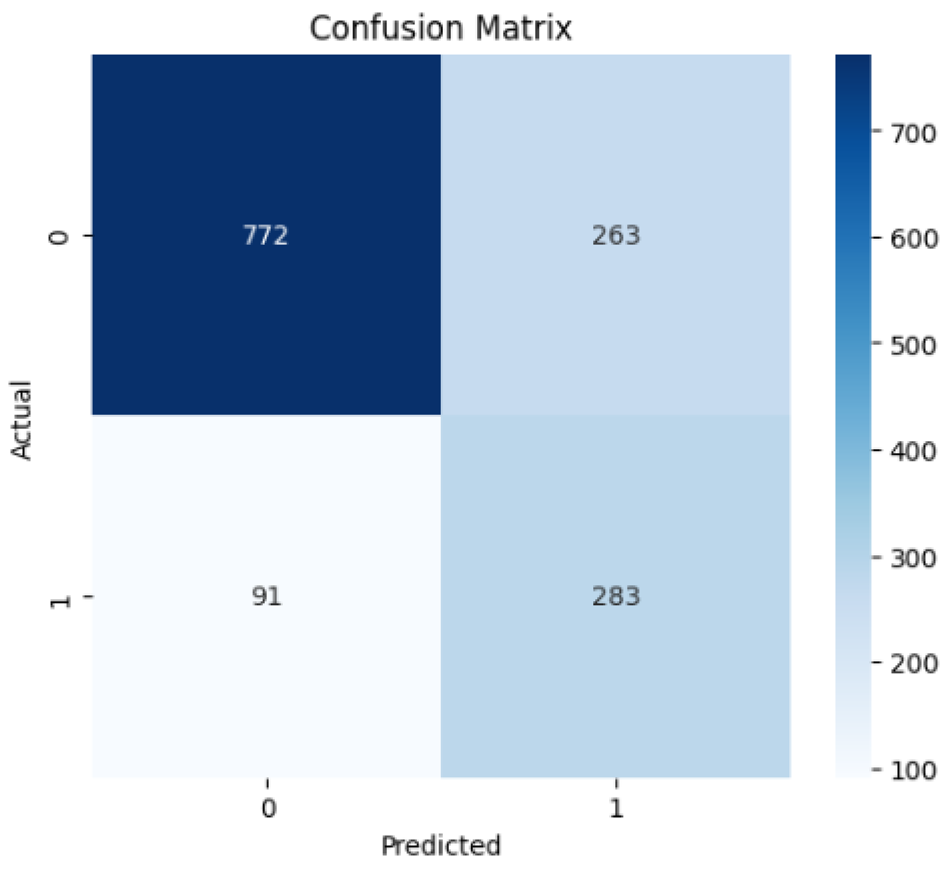
- **Segment Analysis:** The mean values of each feature per cluster are printed to analyze customer segments. This helps understand the characteristics of each cluster (e.g., high tenure, low monthly charges, etc.).

3.8 Random Forest with SMOTE:

- **SMOTE:** The code now explicitly addresses class imbalance using SMOTE (Synthetic Minority Over-sampling Technique) to oversample the minority (churned) class. This helps the model learn from a more balanced dataset.
- **Class Weighting:** The Random Forest model uses class weights ("balanced") to further address imbalance. This gives more weight to the minority class during training, making the model more sensitive to it.
- **Threshold Tuning:** The code tunes the decision threshold to optimize for recall. Lowering the threshold increases recall (correctly identifying more churned

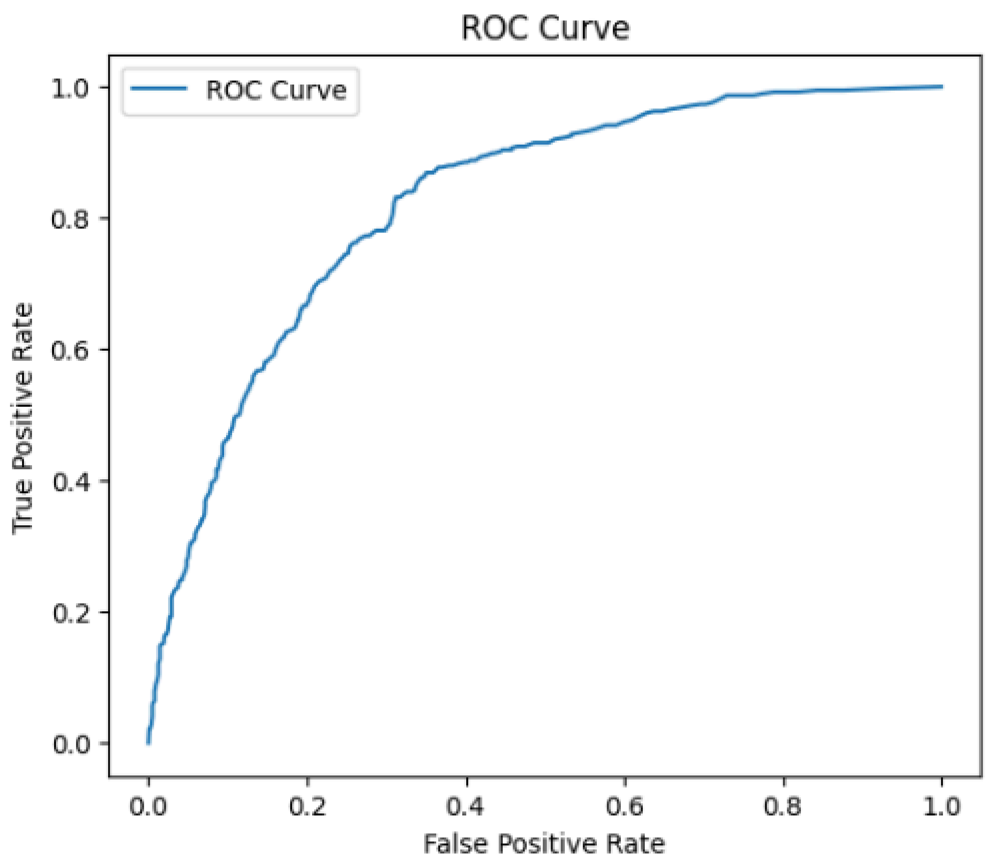
customers) but may decrease precision (incorrectly identifying more non-churned customers as churned).

- **Evaluation:** The model is evaluated using a classification report (including precision, recall, F1-score, and support), a confusion matrix (visualizing true positives, false positives, etc.), and an ROC curve (showing the trade-off between true positive rate and false positive rate).



Fig(21): Confusion Matrix

Result: The **confusion matrix** highlights the model's classification performance, showing strong accuracy in predicting non-churn (548 true negatives) but difficulty in identifying churn (335 true positives, with 487 false positives and 39 false negatives). This results in high precision (93%) but low recall (53%) for non-churn, while for churn, precision is lower (41%) but recall is high (90%), indicating a trade-off between false positives and false negatives.

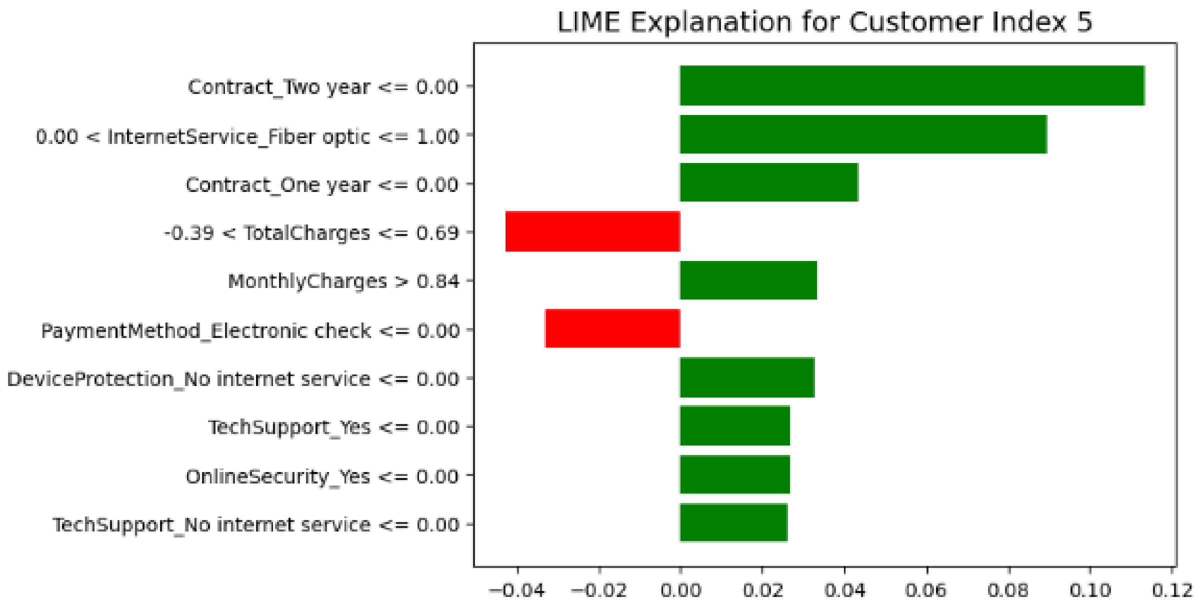


Fig(22): ROC Curve

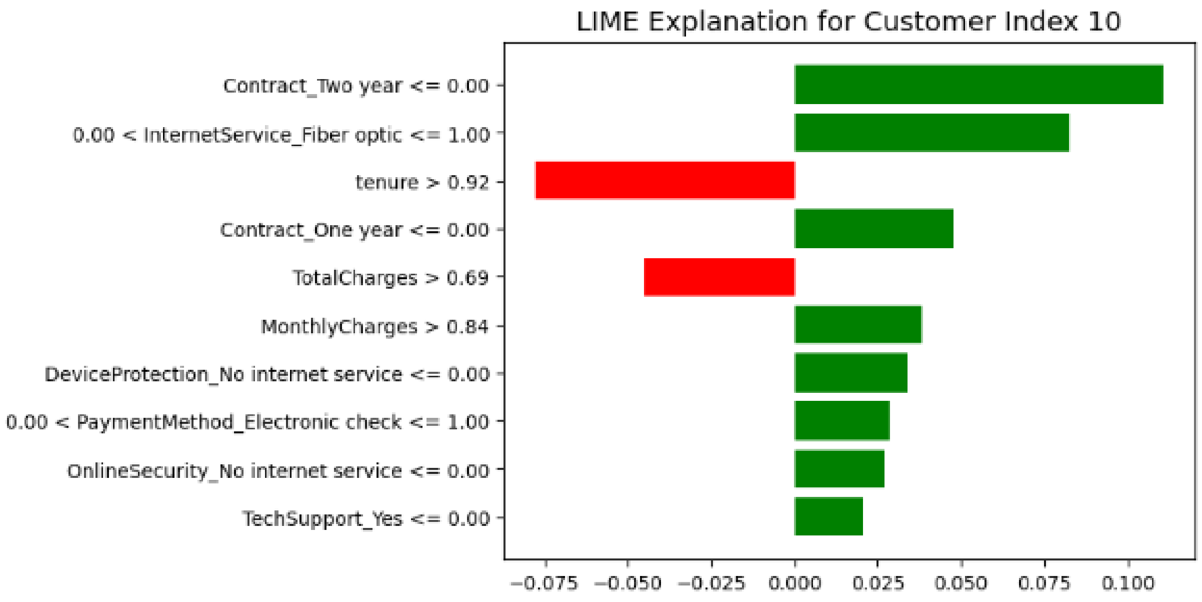
Result: The **ROC curve** confirms that the model performs better than random guessing.

4. Model Interpretation and Explainability

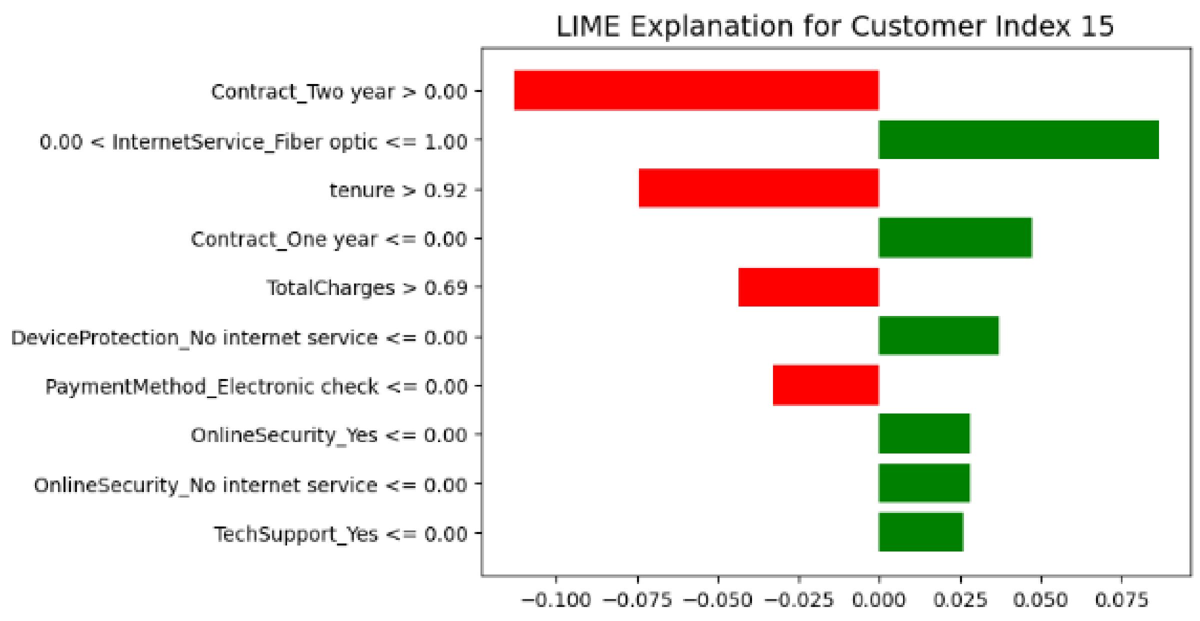
4.1 Local Interpretable Model-agnostic Explanations (LIME): Local Interpretable Model-agnostic Explanations was used to understand individual predictions made by the Random Forest model. LIME provides local explanations by approximating the complex model with a simpler, interpretable one in the vicinity of a specific instance. Several customer instances were analyzed, and the top features influencing their churn predictions were identified and visualized. This helps understand why the model made a specific prediction for a particular customer. The report details the LIME explanation process, including explainer initialization, instance selection, probability prediction, influential feature identification, and visualization.



LIME Explanation for Customer Index 5
Contract_Two year <= 0.00
0.00 < InternetService_Fiber optic <= 1.00
Contract_One year <= 0.00
-0.39 < TotalCharges <= 0.69
MonthlyCharges > 0.84
PaymentMethod_Electronic check <= 0.00
DeviceProtection_No internet service <= 0.00
TechSupport_Yes <= 0.00
OnlineSecurity_Yes <= 0.00
TechSupport_No internet service <= 0.00
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08
0.10
0.12



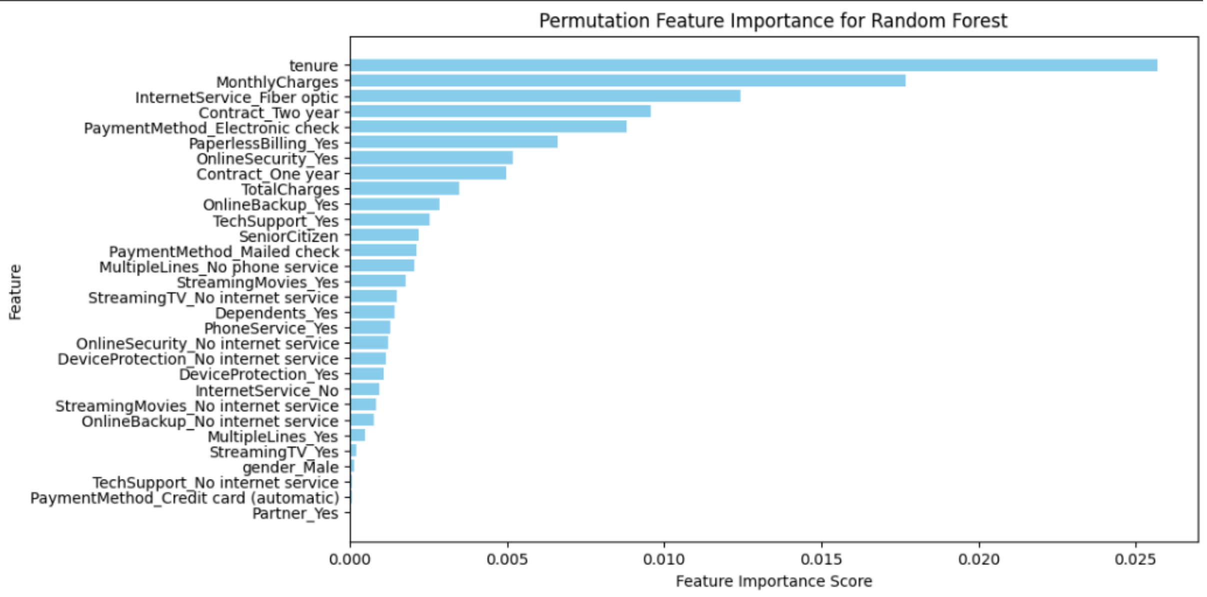
LIME Explanation for Customer Index 10
Contract_Two year <= 0.00
0.00 < InternetService_Fiber optic <= 1.00
tenure > 0.92
Contract_One year <= 0.00
TotalCharges > 0.69
MonthlyCharges > 0.84
DeviceProtection_No internet service <= 0.00
0.00 < PaymentMethod_Electronic check <= 1.00
OnlineSecurity_No internet service <= 0.00
TechSupport_Yes <= 0.00
−0.075
−0.050
−0.025
0.000
0.025
0.050
0.075
0.100



Fig(23): LIME for Customer Index 5,10 & 15

Result: The LIME explanations highlight key factors influencing churn predictions for three customers. For customers 5 and 10, the lack of a two-year contract and having fiber optic internet strongly push them toward churn, while longer tenure and higher total charges slightly lower that risk. In contrast, customer 15 has a much lower churn probability, mainly due to having a two-year contract, despite also using fiber optic internet. Overall, contract type and internet service are the strongest churn predictors—two-year contracts greatly reduce churn likelihood, while fiber optic internet increases it. Tenure and total charges have a secondary impact, with longer tenure and higher charges linked to lower churn probability.

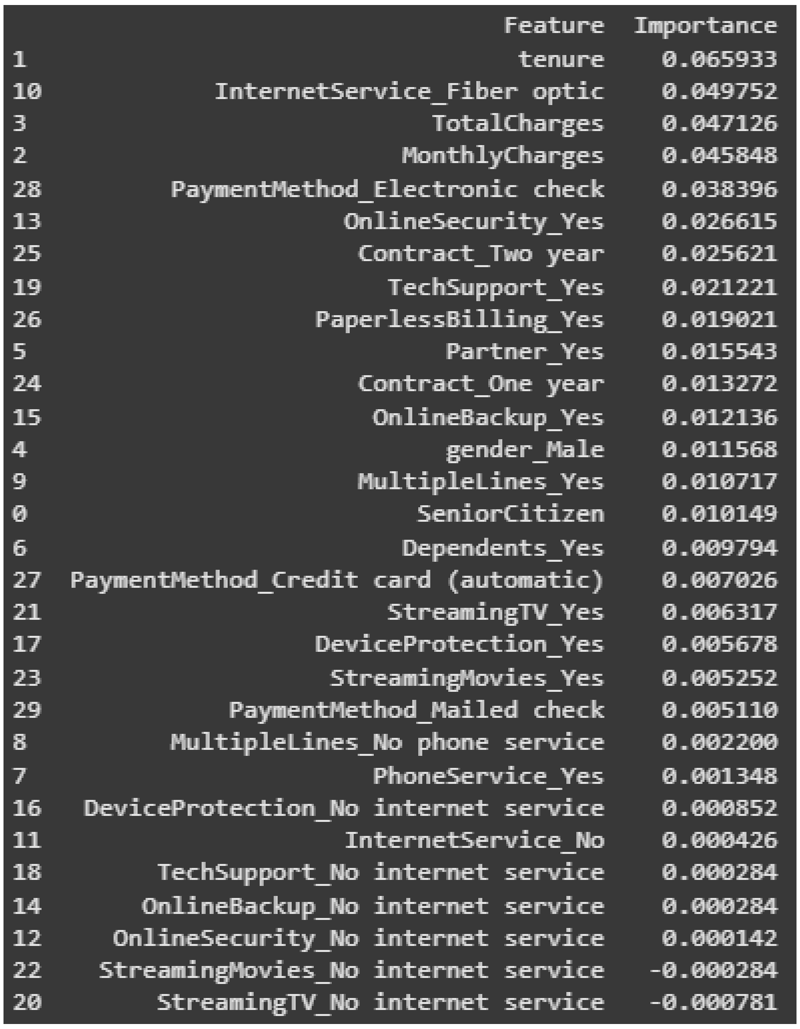
4.2 Permutation Feature Importance: Permutation feature importance was used to assess the global importance of features. By randomly shuffling the values of each feature and observing the decrease in model performance, the importance of that feature was determined. This provided a ranking of features based on their overall impact on churn prediction. The report describes how permutation importance is calculated and how the results are visualized.



Fig(24): Permutation Feature Importance for Random Forest

Result: This horizontal bar chart illustrates the permutation feature importance for a Random Forest model, showing the features that most influence the model's predictions. The length of each bar reflects the importance score, with longer bars indicating higher importance. "**Tenure**" emerges as the most influential feature, followed by "**MonthlyCharges**", "**InternetService_Fiber optic**", and "**Contract_Two year**", suggesting their critical role in predicting the target variable. On the other hand, features like "**Partner_Yes**" and "**PaymentMethod_Credit card (automatic)**" have the lowest importance scores, indicating they have minimal impact. The chart offers valuable insight into feature significance, guiding feature selection or further analysis to prioritize the most impactful variables.

4.3 Feature Importance Analysis: A DataFrame is created to store the importance of each feature based on permutation importance. The features are sorted in descending order of their importance values to highlight the most significant predictors. The printed DataFrame provides insights into the contribution of each feature, allowing for potential feature selection or engineering improvements.

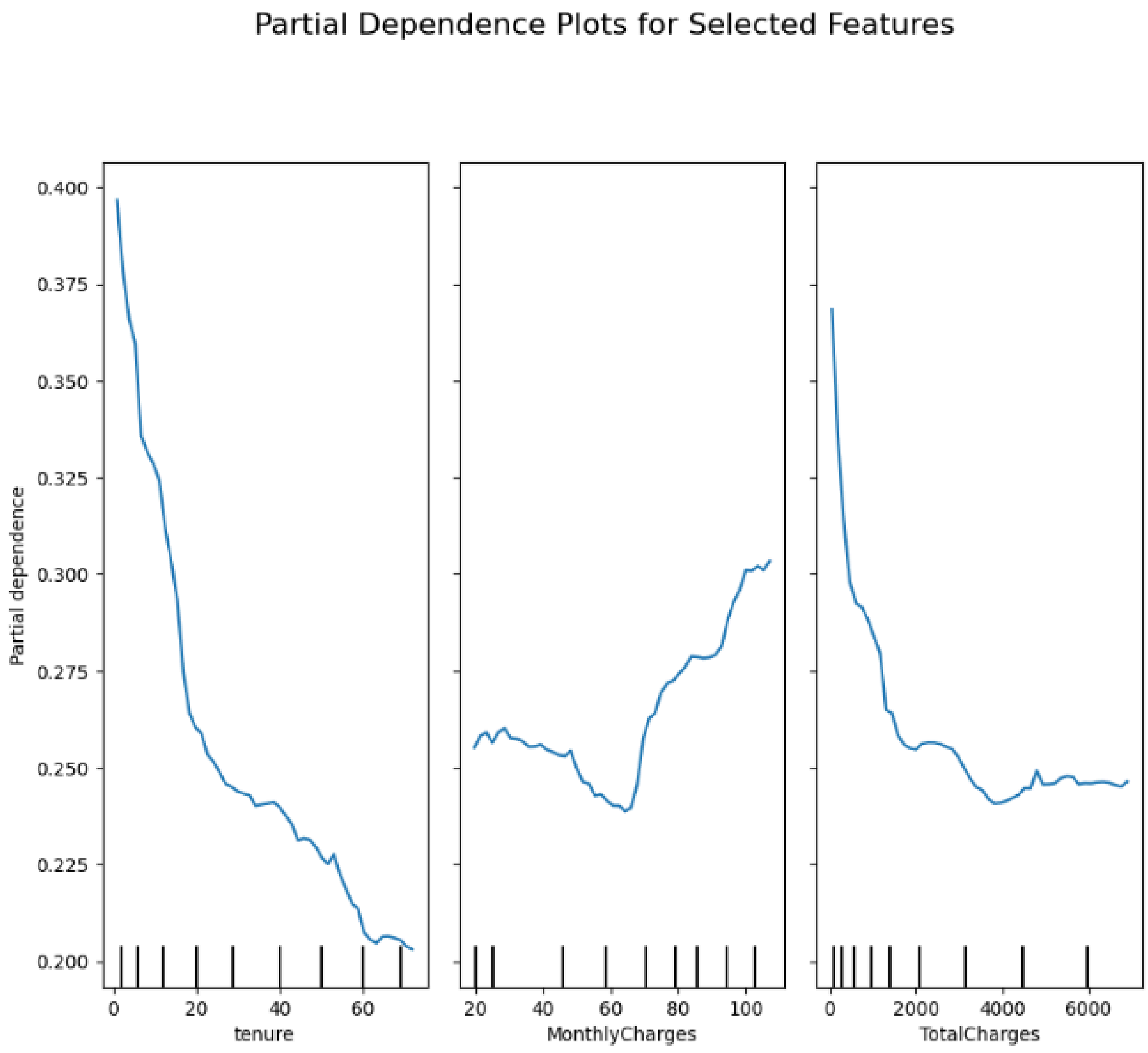


```
                                    Feature  Importance
1                                    tenure    0.065933
10             InternetService_Fiber optic    0.049752
3                              TotalCharges    0.047126
2                            MonthlyCharges    0.045848
28          PaymentMethod_Electronic check    0.038396
13                       OnlineSecurity_Yes    0.026615
25                        Contract_Two year    0.025621
19                          TechSupport_Yes    0.021221
26                     PaperlessBilling_Yes    0.019021
5                               Partner_Yes    0.015543
24                        Contract_One year    0.013272
15                         OnlineBackup_Yes    0.012136
4                               gender_Male    0.011568
9                         MultipleLines_Yes    0.010717
0                             SeniorCitizen    0.010149
6                            Dependents_Yes    0.009794
27  PaymentMethod_Credit card (automatic)    0.007026
21                          StreamingTV_Yes    0.006317
17                     DeviceProtection_Yes    0.005678
23                      StreamingMovies_Yes    0.005252
29              PaymentMethod_Mailed check    0.005110
8          MultipleLines_No phone service    0.002200
7                          PhoneService_Yes    0.001348
16   DeviceProtection_No internet service    0.000852
11                       InternetService_No    0.000426
18        TechSupport_No internet service    0.000284
14       OnlineBackup_No internet service    0.000284
12     OnlineSecurity_No internet service    0.000142
22    StreamingMovies_No internet service   -0.000284
20        StreamingTV_No internet service   -0.000781
```

Fig(25): Feature Importance Analysis

Result: The table above ranks customer characteristics by their importance in making predictions, such as churn. Tenure (how long a customer has been with the service) is the most important factor, followed by monthly charges, having fiber optic internet, and having a two-year contract. Other factors, like payment method and online security, also contribute, while traits like having a partner or specific streaming services have minimal impact. In short, the table highlights which customer traits are most useful for making accurate predictions.

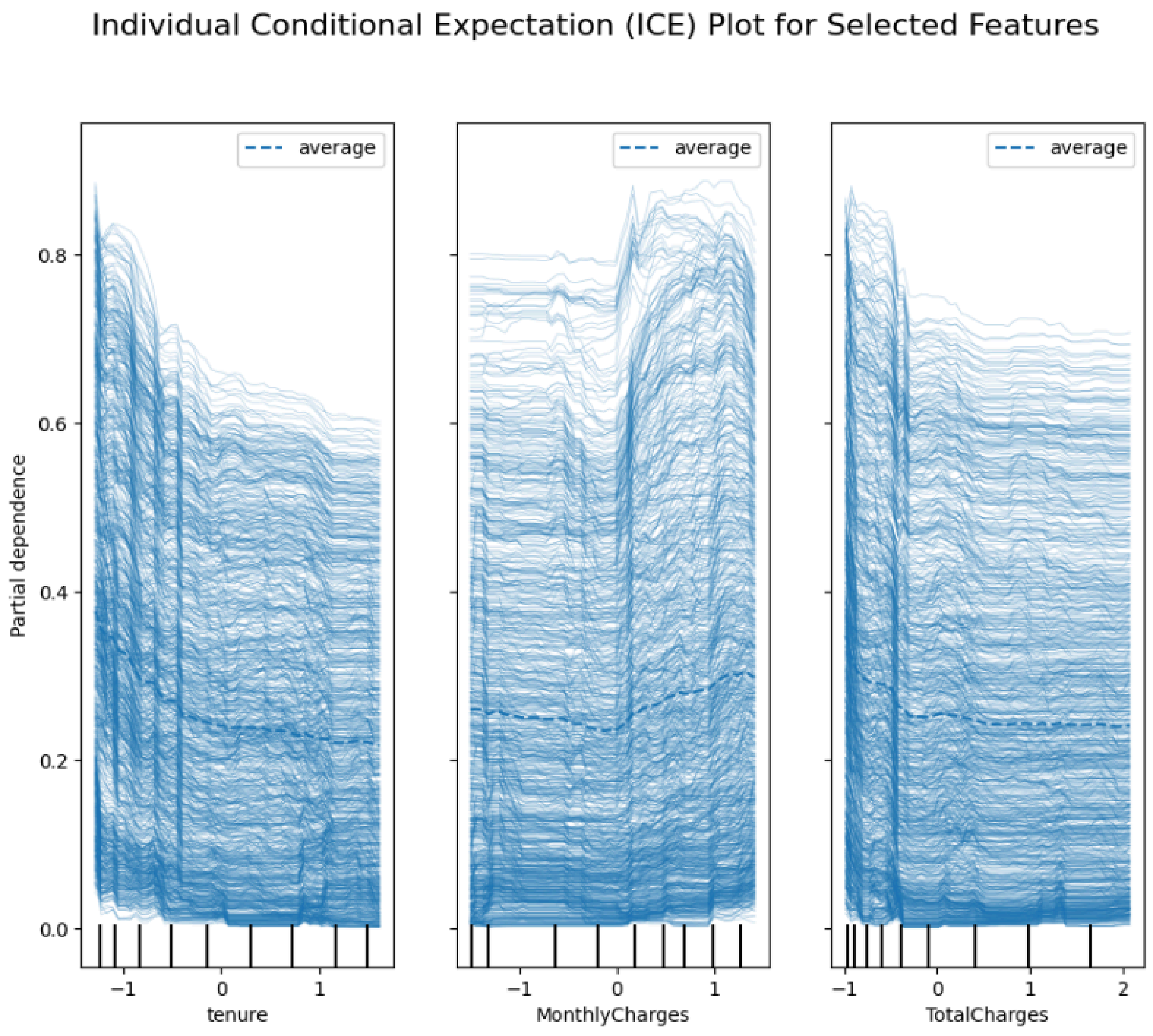
4.4 Partial Dependence Plots (PDP): Partial Dependence Plots were generated to visualize the marginal effect of specific features (tenure, MonthlyCharges, TotalCharges) on the predicted probability of churn. PDPs show how the model's predictions change as the value of a feature varies, holding other features constant. This helps visualize the relationship between a feature and the model's predictions. The report explains how PDPs are generated and interpreted.



Fig(26): Partial Dependence Plots

Result: The figure above displays partial dependence plots for three features (tenure, MonthlyCharges, and TotalCharges) from a machine learning model, illustrating how each feature influences the model's predictions while keeping other features constant. The plots highlight the relationship between each feature and the target variable, showing how the model's output varies across the feature ranges. For example, the tenure plot indicates a general decline in partial dependence as tenure increases, while MonthlyCharges and TotalCharges reveal more complex, non-linear relationships with the target. These plots provide insights into the individual impact of each feature on the model's predictions, which can assist in feature selection or engineering.

4.5 Individual Conditional Expectation (ICE): Individual Conditional Expectation (ICE) plots were created to show how the predictions for *individual* instances change as a feature varies. ICE plots provide more granular insights into how the model's predictions are influenced by feature values for specific customers. The dice_ml library was used for this purpose. The report explains how ICE plots are generated and their utility in understanding individual customer behavior. The combined ICE and PDP plot is also mentioned.



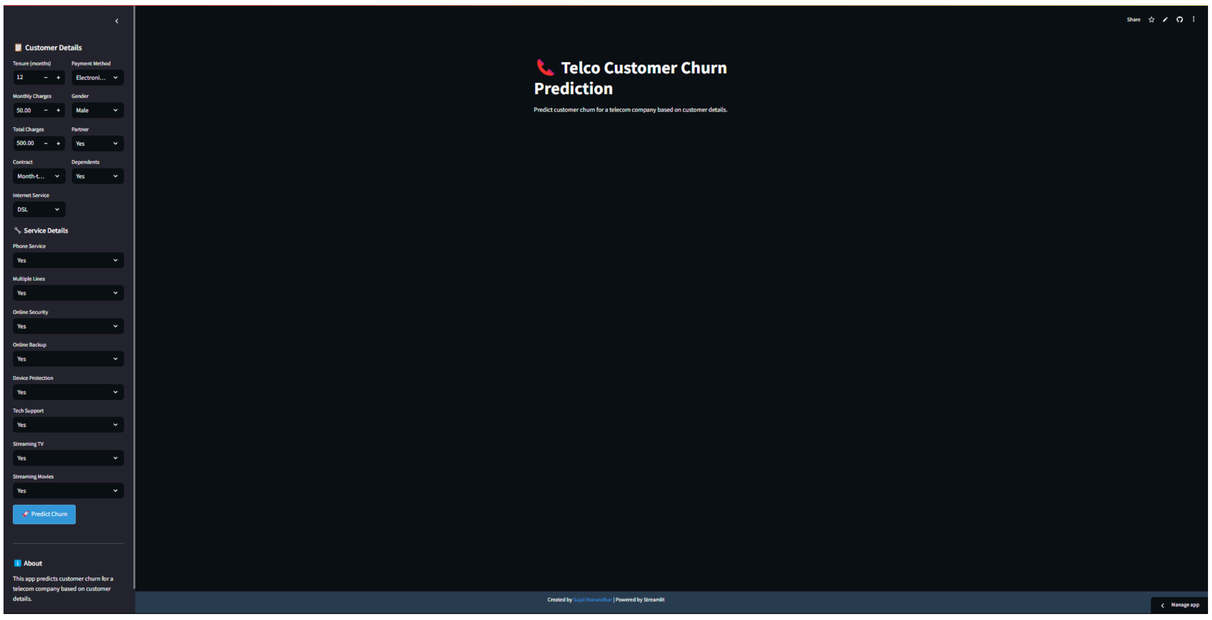
Fig(27): Individual Conditional Expectation

Result: The figure presents **Individual Conditional Expectation (ICE)** plots alongside **Partial Dependence Plots (PDPs)** for three features—tenure, MonthlyCharges, and TotalCharges—from a trained model, likely a **Random Forest**. The ICE plots illustrate how the model's predictions change for individual instances as each feature varies, uncovering potential heterogeneous relationships across instances. The overlaid PDPs depict the average prediction across all instances for each feature value, providing a summary of the overall trend. By combining both ICE and PDP, we gain insights into the general relationship between the features and predictions, while also identifying any variations in how individual instances are influenced by these features. For example, while the PDP for tenure might show a decreasing prediction trend with increasing tenure, the ICE plots could highlight specific customer segments where this trend deviates.

5. Model Saving and Deployment

The updated code now saves the scaler, label encoder, and the trained model using joblib.dump(). This ensures that all necessary components for prediction are saved. This is crucial for deploying the model, as the same scaling and encoding needs to be applied to new data.

To deploy the Telco Customer Churn Prediction dashboard, I used **Streamlit**, a powerful framework for building and deploying data applications. The deployment process involved packaging the Python script along with the pre-trained machine learning model (`**telco_churn_model.pkl**`), scaler (`**scaler.pkl**`), and label encoder (`**label_encoder.pkl**`). I hosted the application on Streamlit Sharing, a platform specifically designed for deploying Streamlit apps. After pushing the code to a **GitHub** repository, I connected the repository to **Streamlit** Sharing, which automatically built and deployed the app. The result is a fully functional, interactive web application that allows users to input customer details and receive real-time churn predictions. The app is accessible via a public URL, making it easy for stakeholders to use and share. The deployment was seamless, and the app performs efficiently, providing accurate predictions with a clean and modern user interface. This deployment demonstrates the power of combining machine learning with web development to create practical, user-friendly tools for business decision-making. Link: https://telco-churn-prediction-0929.streamlit.app/



Customer Details

Tenure (months): 12	Payment Method: Electroni...
Monthly Charges: 50.00	Gender: Male
Total Charges: 500.00	Partner: Yes
Contract: Month-t...	Dependents: Yes
Internet Service: DSL	

Service Details

Phone Service: Yes
Multiple Lines: Yes
Online Security: Yes
Online Backup: Yes
Device Protection: Yes
Tech Support: Yes
Streaming TV: Yes
Streaming Movies: Yes

About

This app predicts customer churn for a telecom company based on customer details.

Telco Customer Churn Prediction

Predict customer churn for a telecom company based on customer details.

Created by Sujal Manandhar | Powered by Streamlit

<u>Fig(28): Telco Customer Churn Prediction Web App Interface</u>

6. Insights

6.1 Key Insights

- The **Random Forest** model emerged as the most accurate model for predicting customer churn.
- **Tenure** and **Monthly Charges** were identified as the most important features influencing churn.
- **Customer Segmentation** through clustering revealed three distinct customer groups with different churn risks, helping businesses tailor retention strategies.
- **SMOTE** effectively addressed class imbalance, leading to improved performance in predicting churn.

This analysis reveals key drivers of customer churn, including tenure, monthly charges, and internet service type. Longer tenures correlate with lower churn risk, while higher monthly charges and fiber optic internet service increase the likelihood of churn. Two-year contracts significantly reduce churn. A **Random Forest** model, optimized via hyperparameter tuning and **SMOTE** to address class imbalance, demonstrated strong predictive capabilities. K-Means clustering identified distinct customer segments based on tenure and charges, enabling targeted interventions. **LIME** and **permutation importance** highlighted the importance of contract type, internet service, tenure, and total charges. **Partial dependence** plots and **ICE** curves further elucidated the relationships between these features and churn probability, revealing potential individual customer variations. These insights provide a basis for a proactive churn management strategy, focusing on high-risk customers with targeted retention efforts.

6.2 Recommendations

- Focus on customers with low tenure and high monthly charges for retention efforts.
- Use segmentation insights to create personalized marketing and retention strategies.
- Regularly retrain models and perform hyperparameter tuning to maintain predictive accuracy.

7. Future Work

Future improvements could involve exploring:

- **Deep Learning Models**: To further enhance predictive performance.
- **Time-Series Analysis**: To predict churn over time based on customer activity trends.
- **Feature Engineering**: Creating new features from raw data, such as customer interactions or engagement metrics.

8. Conclusion

This project successfully developed and evaluated several machine learning models for predicting customer churn in a telecom company. The **Random Forest** model, particularly after addressing class imbalance with **SMOTE** and optimizing the decision **threshold** for recall, demonstrated strong predictive performance. The comprehensive **EDA** and model interpretation techniques provided valuable insights into the factors driving churn.
These insights can be used by the telecom company to develop targeted retention strategies, personalize customer interactions, and ultimately reduce churn rates. Future work could involve exploring additional feature engineering, incorporating other data sources, or deploying the model as a service for real-time predictions. The saved model, scaler, and label encoder allow for easy deployment and use in a production environment.