1.Import Required Libraries

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

2.Load the Dataset

```
from google.colab import files
uploaded= files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Telco-Customer-Churn.csv to Telco-Customer-Churn.csv
```

```
df = pd.read_csv("Telco-Customer-Churn.csv")
print(df.head())
```

```
   customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No   
1  5575-GNVDE    Male              0      No         No      34          Yes   
2  3668-QPYBK    Male              0      No         No       2          Yes   
3  7795-CFOCW    Male              0      No         No      45           No   
4  9237-HQITU  Female              0      No         No       2          Yes   

      MultipleLines InternetService OnlineSecurity  ... DeviceProtection  \
0  No phone service             DSL             No  ...               No   
1                No             DSL            Yes  ...              Yes   
2                No             DSL            Yes  ...               No   
3  No phone service             DSL            Yes  ...              Yes   
4                No     Fiber optic             No  ...               No   

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes   
1          No          No              No        One year               No   
2          No          No              No  Month-to-month              Yes   
3         Yes          No              No        One year               No   
4          No          No              No  Month-to-month              Yes   
```

```
                PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

Data Precessing

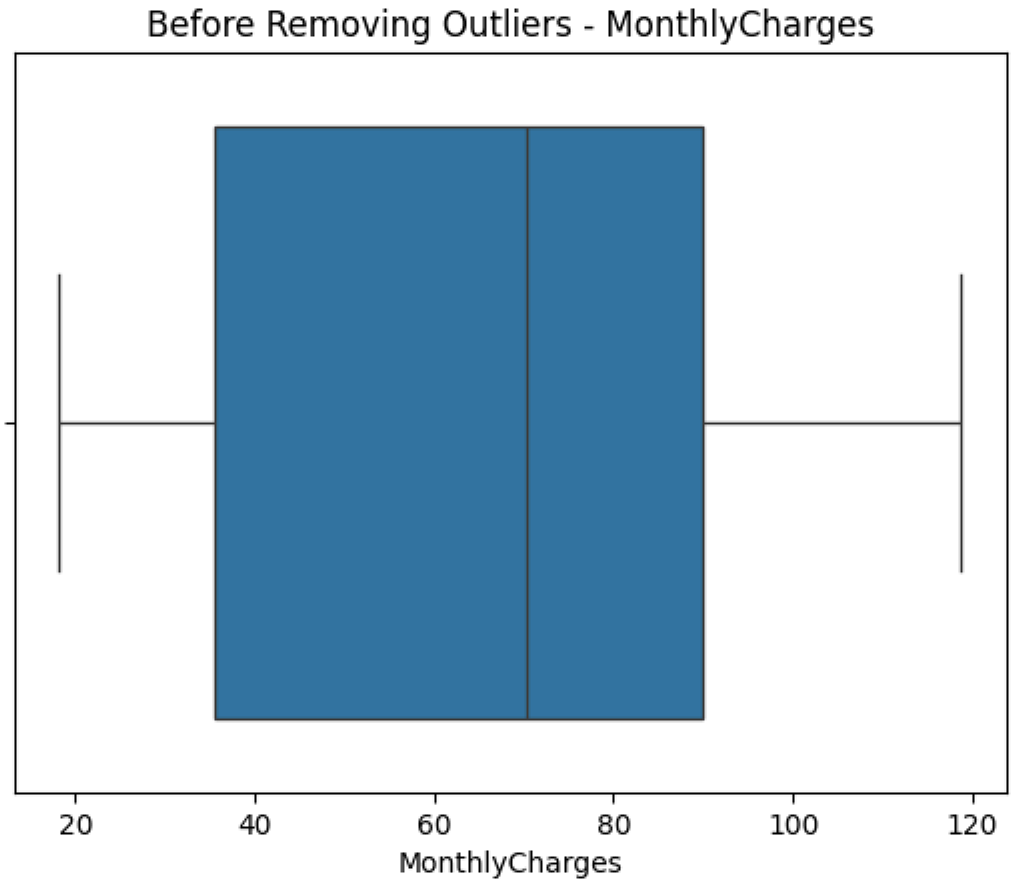
```
# Check for missing values
print(df.isnull().sum())

# Convert TotalCharges to numeric (it has missing/blank values)
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"], errors='coerce')

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

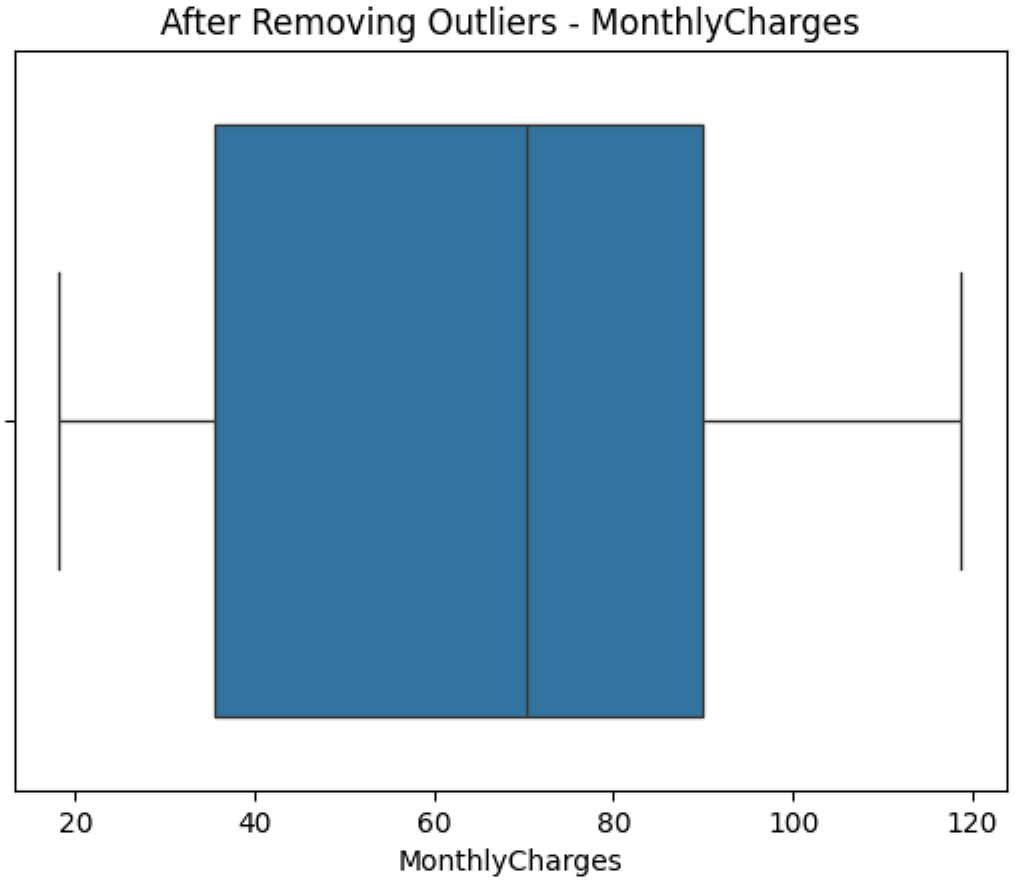
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
<ipython-input-8-29992b9cd5e8>:8: FutureWarning: A value is trying to be set on a copy c
The behavior will change in pandas 3.0. This inplace method will never work because the

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col


  df["TotalCharges"].fillna(df["TotalCharges"].median(), inplace=True)
```

```
# Check for duplicates
print("Duplicates:", df.duplicated().sum())

# Drop duplicates if any
df.drop_duplicates(inplace=True)
```

```
Duplicates: 0
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot before removing outliers
sns.boxplot(x=df["MonthlyCharges"])
plt.title("Before Removing Outliers - MonthlyCharges")
plt.show()

# Remove outliers using IQR
Q1 = df["MonthlyCharges"].quantile(0.25)
Q3 = df["MonthlyCharges"].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df = df[(df["MonthlyCharges"] >= lower_bound) & (df["MonthlyCharges"] <= upper_bound)]

# Boxplot after removing outliers
sns.boxplot(x=df["MonthlyCharges"])
plt.title("After Removing Outliers - MonthlyCharges")
plt.show()
```


Before Removing Outliers - MonthlyCharges
20
40
60
80
100
120
MonthlyCharges


After Removing Outliers - MonthlyCharges
20
40
60
80
100
120
MonthlyCharges

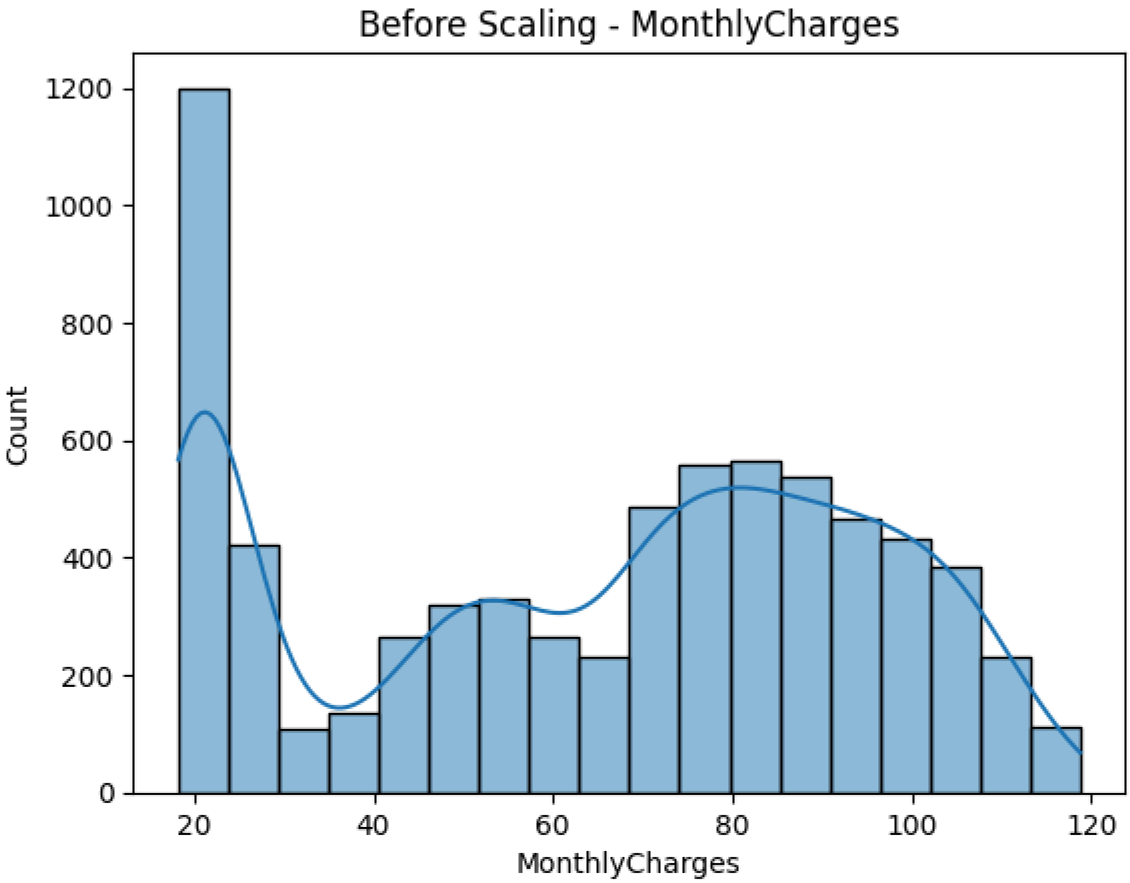
```
from sklearn.preprocessing import LabelEncoder

df_encoded = df.copy()
categorical_cols = df_encoded.select_dtypes(include='object').columns.tolist() # Convert to

# Drop customerID as it's not useful
if 'customerID' in categorical_cols: # Check if 'customerID' is in the list before removing
    categorical_cols.remove('customerID')
    df_encoded.drop("customerID", axis=1, inplace=True)



# Label encode binary categorical features
le = LabelEncoder()
for col in categorical_cols:
    # Ensure the column still exists in df_encoded after potential drops
    if col in df_encoded.columns:
        if df_encoded[col].nunique() == 2:
            df_encoded[col] = le.fit_transform(df_encoded[col])
        # The 'elif col != 'customerID':` is no longer needed since 'customerID' is removed
        else:
            df_encoded = pd.get_dummies(df_encoded, columns=[col], prefix=col, drop_first=Tr

print(df_encoded.head())
```

```
   gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  \
0       0              0        1           0       1             0
1       1              0        0           0      34             1
2       1              0        0           0       2             1
3       1              0        0           0      45             0
4       0              0        0           0       2             1

   PaperlessBilling  MonthlyCharges  TotalCharges  Churn  ...  \
0                 1           29.85         29.85      0  ...
1                 0           56.95       1889.50      0  ...
2                 1           53.85        108.15      1  ...
3                 0           42.30       1840.75      0  ...
4                 1           70.70        151.65      1  ...

   TechSupport_Yes  StreamingTV_No internet service  StreamingTV_Yes  \
0            False                            False            False
1            False                            False            False
2            False                            False            False
3             True                            False            False
4            False                            False            False

   StreamingMovies_No internet service  StreamingMovies_Yes  \
0                                False                False
1                                False                False
2                                False                False
3                                False                False
4                                False                False

   Contract_One year  Contract_Two year  \
0              False              False
1               True              False
2              False              False
3               True              False
4              False              False

   PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic check  \
0                                  False                            True
1                                  False                           False
2                                  False                           False
3                                  False                           False
4                                  False                            True

   PaymentMethod_Mailed check
0                       False
1                        True
2                        True
3                       False
4                       False

[5 rows x 31 columns]
```

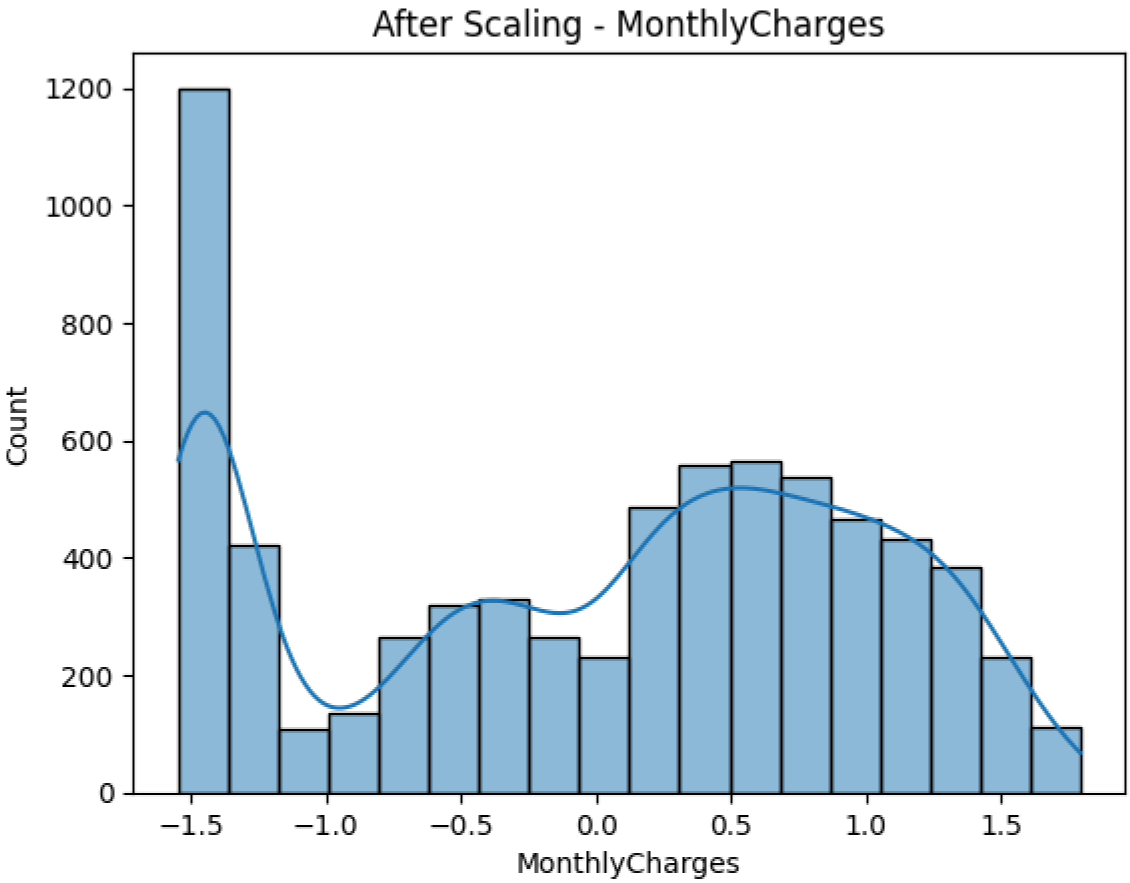
```
from sklearn.preprocessing import StandardScaler

# Scale numerical columns
scaler = StandardScaler()
numeric_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
df_encoded[numeric_cols] = scaler.fit_transform(df_encoded[numeric_cols])

print(df_encoded[numeric_cols].describe())
```

```
             tenure  MonthlyCharges  TotalCharges
count  7.043000e+03    7.043000e+03  7.043000e+03
mean  -2.421273e-17   -6.406285e-17 -1.488074e-17
std    1.000071e+00    1.000071e+00  1.000071e+00
min   -1.318165e+00   -1.545860e+00 -9.991203e-01
25%   -9.516817e-01   -9.725399e-01 -8.298459e-01
50%   -1.372744e-01    1.857327e-01 -3.904632e-01
75%    9.214551e-01    8.338335e-01  6.642871e-01
max    1.613701e+00    1.794352e+00  2.826743e+00
```

```
sns.histplot(df["MonthlyCharges"], kde=True)
plt.title("Before Scaling - MonthlyCharges")
plt.show()
```



```
# Assuming you want to scale numerical features and store the result in df_encoded
# This is a placeholder and might need adjustment based on your full data processing steps

from sklearn.preprocessing import StandardScaler

# Select the numerical column(s) to scale
numerical_cols = ['MonthlyCharges']

# Create a copy of the original DataFrame to avoid modifying it directly
df_encoded = df.copy()

# Initialize the StandardScaler
scaler = StandardScaler()

# Fit and transform the numerical column(s)
df_encoded[numerical_cols] = scaler.fit_transform(df_encoded[numerical_cols])

# Now you can plot the scaled 'MonthlyCharges'
sns.histplot(df_encoded["MonthlyCharges"], kde=True)
plt.title("After Scaling - MonthlyCharges")
plt.show()
```

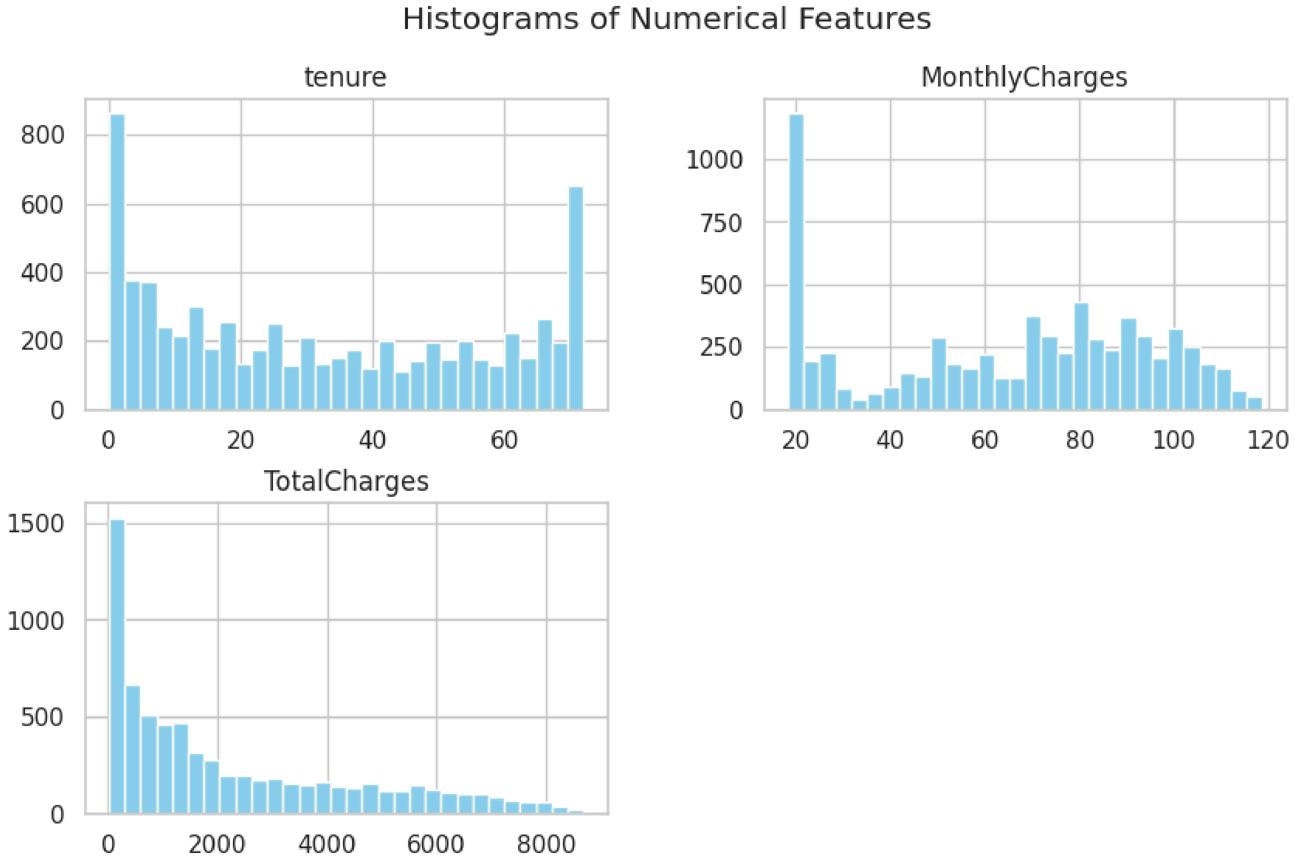


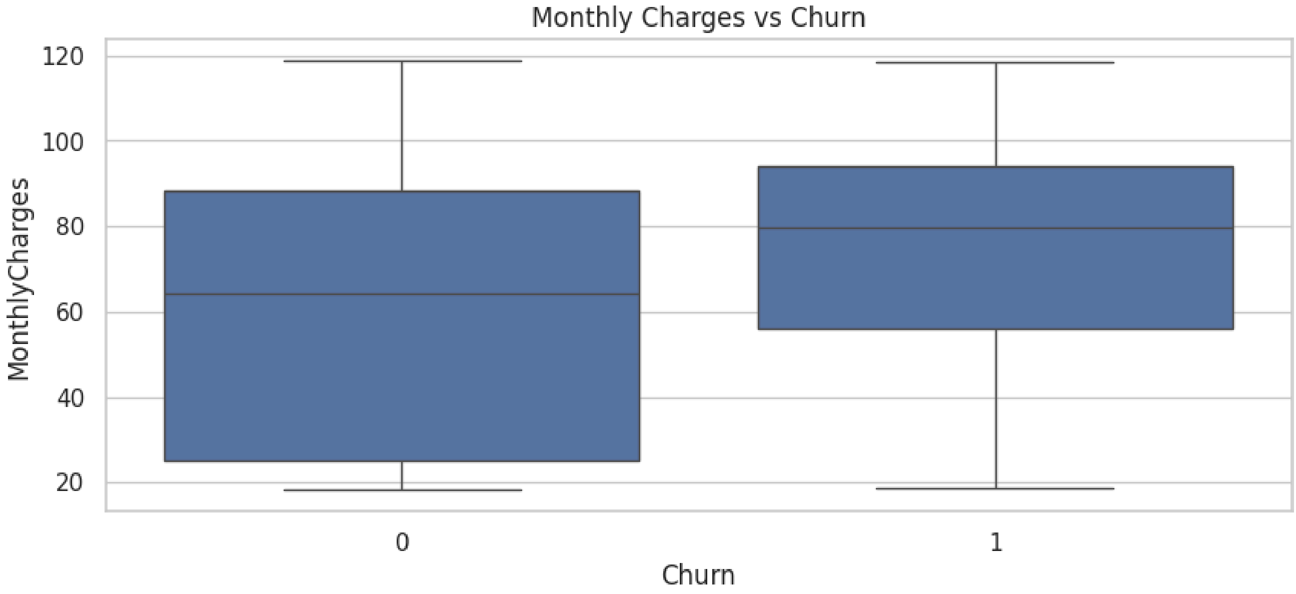
EDA

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("Telco-Customer-Churn.csv")

# Convert TotalCharges to numeric and handle missing values
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"], errors='coerce')
df["TotalCharges"].fillna(df["TotalCharges"].median(), inplace=True)

# Drop customerID
df.drop("customerID", axis=1, inplace=True)

# Encode target variable
df["Churn"] = df["Churn"].map({"Yes": 1, "No": 0})

# Set seaborn style
sns.set(style="whitegrid")

# Histograms
```

```
df[["tenure", "MonthlyCharges", "TotalCharges"]].hist(bins=30, figsize=(10, 6), color='skybl
plt.suptitle("Histograms of Numerical Features")
plt.show()

# Boxplot: Monthly Charges vs Churn
plt.figure(figsize=(10, 4))
sns.boxplot(x='Churn', y='MonthlyCharges', data=df)
plt.title('Monthly Charges vs Churn')
plt.show()
```

```
<ipython-input-16-a131b12f6053>:10: FutureWarning: A value is trying to be set on a cc
The behavior will change in pandas 3.0. This inplace method will never work because th

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({c


  df["TotalCharges"].fillna(df["TotalCharges"].median(), inplace=True)
```



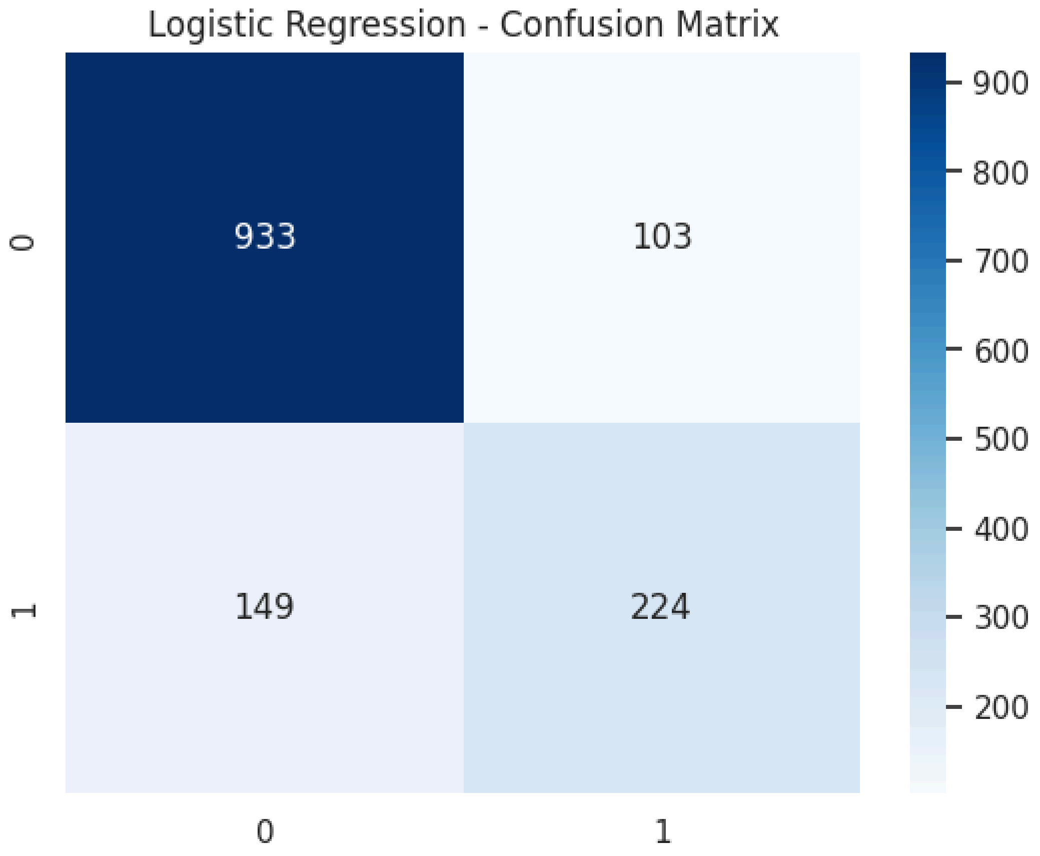


Model Building

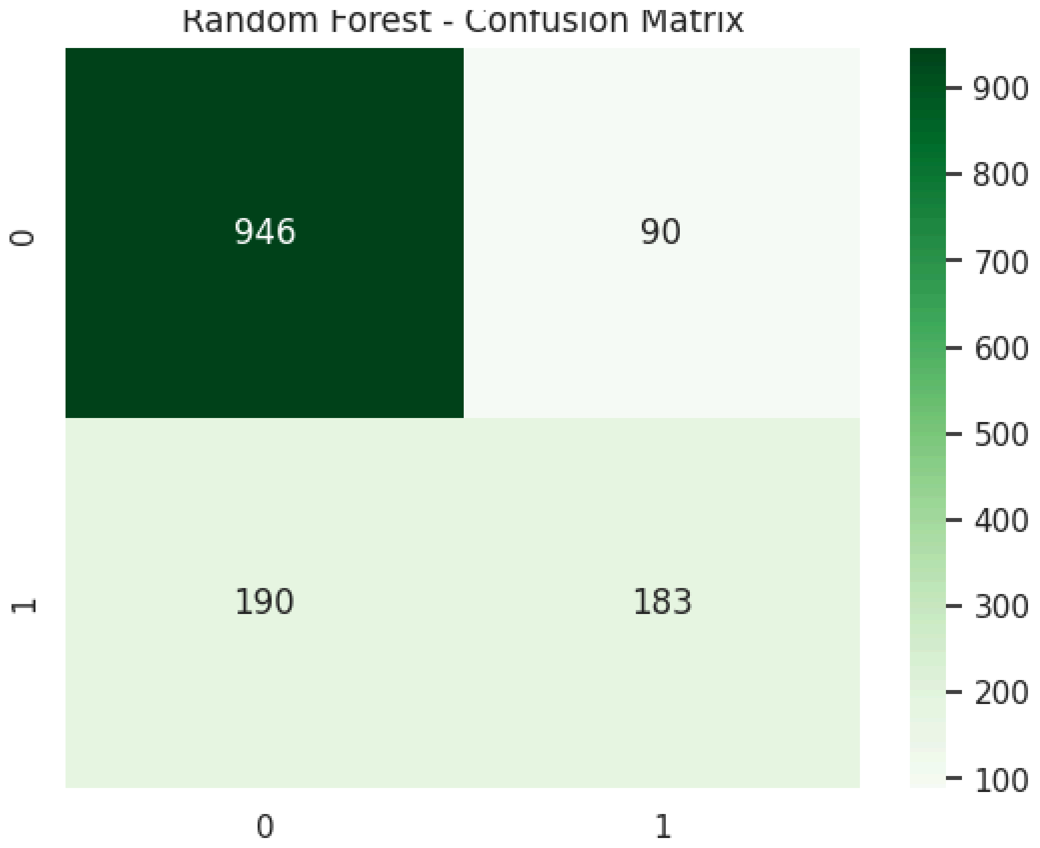
```
# Import required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("Telco-Customer-Churn.csv")

# Data Preprocessing
df.drop('customerID', axis=1, inplace=True)
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df['TotalCharges'].fillna(df['TotalCharges'].median(), inplace=True)

# Encode categorical variables
le = LabelEncoder()
df['Churn'] = le.fit_transform(df['Churn'])  # Yes/No to 1/0
for column in df.select_dtypes(include='object').columns:
    if df[column].nunique() == 2:
        df[column] = le.fit_transform(df[column])
    else:
        df = pd.get_dummies(df, columns=[column])

# Feature Scaling
scaler = StandardScaler()
df[['tenure', 'MonthlyCharges', 'TotalCharges']] = scaler.fit_transform(df[['tenure', 'Mont

# Split data
X = df.drop('Churn', axis=1)
y = df['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# ----------------------------
# Logistic Regression
```

```python
# ----------------------------
log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)
y_pred_log = log_model.predict(X_test)

print("📘 Logistic Regression Results:")
print("Accuracy:", accuracy_score(y_test, y_pred_log))
print(classification_report(y_test, y_pred_log))
sns.heatmap(confusion_matrix(y_test, y_pred_log), annot=True, fmt='d', cmap='Blues')
plt.title("Logistic Regression - Confusion Matrix")
plt.show()

# ----------------------------
# Random Forest
# ----------------------------
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

print("\n📗 Random Forest Results:")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d', cmap='Greens')
plt.title("Random Forest - Confusion Matrix")
plt.show()
```

```
<ipython-input-17-3fb8eed057d3>:17: FutureWarning: A value is trying to set on a cc
The behavior will change in pandas 3.0. This inplace method will never work because th

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({c


  df['TotalCharges'].fillna(df['TotalCharges'].median(), inplace=True)
📘 Logistic Regression Results:
Accuracy: 0.8211497515968772
              precision    recall  f1-score   support

           0       0.86      0.90      0.88      1036
           1       0.69      0.60      0.64       373

    accuracy                           0.82      1409
   macro avg       0.77      0.75      0.76      1409
weighted avg       0.82      0.82      0.82      1409
```



```
📗 Random Forest Results:
Accuracy: 0.801277501774308
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1036
           1       0.67      0.49      0.57       373

    accuracy                           0.80      1409
   macro avg       0.75      0.70      0.72      1409
weighted avg       0.79      0.80      0.79      1409
```



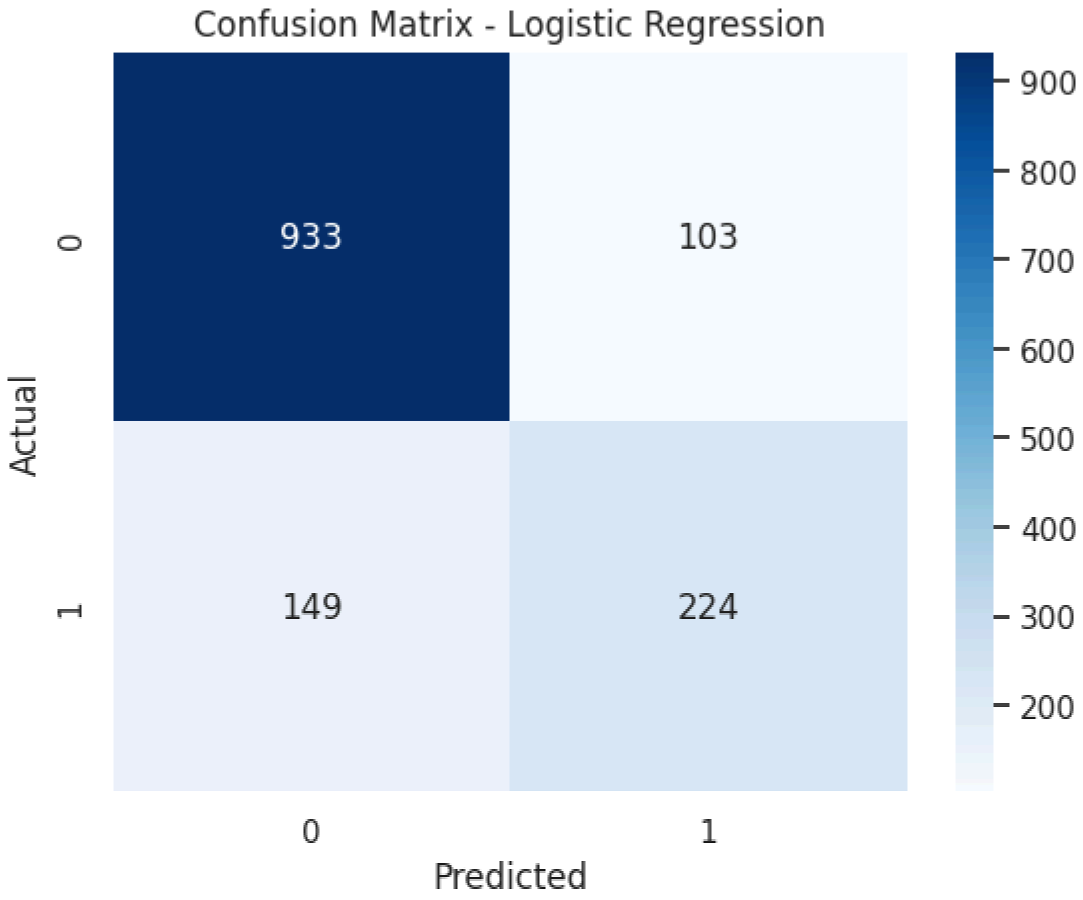
Model Evaluation

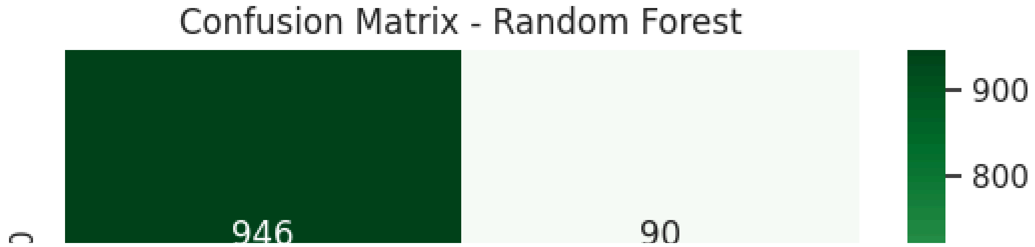
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    accuracy_score, f1_score, roc_auc_score,
    mean_squared_error, confusion_matrix, roc_curve
)

# Load dataset
df = pd.read_csv("Telco-Customer-Churn.csv")

# Data preprocessing
df.drop('customerID', axis=1, inplace=True)
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
```

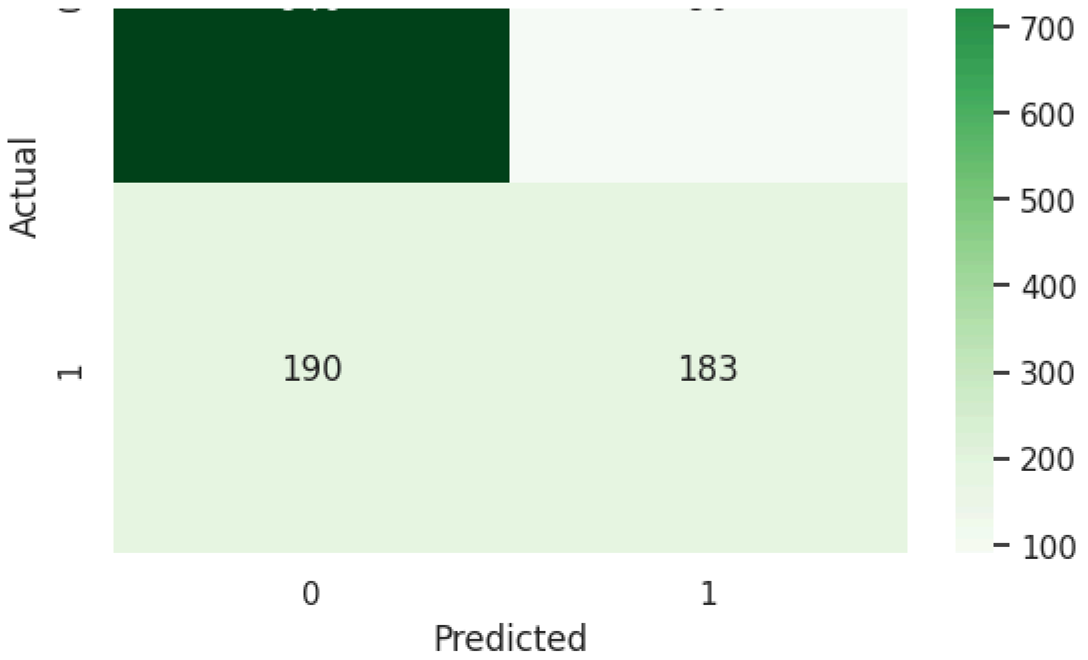
```
    df['TotalCharges'].fillna(df['TotalCharges'].median(), inplace=True)

    le = LabelEncoder()
    df['Churn'] = le.fit_transform(df['Churn'])
    for col in df.select_dtypes(include='object').columns:
        if df[col].nunique() == 2:
            df[col] = le.fit_transform(df[col])
        else:
            df = pd.get_dummies(df, columns=[col])

    scaler = StandardScaler()
    df[['tenure', 'MonthlyCharges', 'TotalCharges']] = scaler.fit_transform(df[['tenure', 'Month

    # Split data
    X = df.drop("Churn", axis=1)
    y = df["Churn"]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Train models
    log_model = LogisticRegression(max_iter=1000)
    rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
    log_model.fit(X_train, y_train)
    rf_model.fit(X_train, y_train)

    # Predictions
    log_preds = log_model.predict(X_test)
    rf_preds = rf_model.predict(X_test)

    # Evaluation metrics
    print("Logistic Regression")
    print("Accuracy:", accuracy_score(y_test, log_preds))
    print("F1 Score:", f1_score(y_test, log_preds))
    print("ROC AUC:", roc_auc_score(y_test, log_model.predict_proba(X_test)[:, 1]))
    print("RMSE:", np.sqrt(mean_squared_error(y_test, log_preds)))
    print(confusion_matrix(y_test, log_preds))

    print("\nRandom Forest")
    print("Accuracy:", accuracy_score(y_test, rf_preds))
    print("F1 Score:", f1_score(y_test, rf_preds))
    print("ROC AUC:", roc_auc_score(y_test, rf_model.predict_proba(X_test)[:, 1]))
    print("RMSE:", np.sqrt(mean_squared_error(y_test, rf_preds)))
    print(confusion_matrix(y_test, rf_preds))

    # Confusion matrix plots
    sns.heatmap(confusion_matrix(y_test, log_preds), annot=True, fmt='d', cmap='Blues')
    plt.title("Confusion Matrix - Logistic Regression")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

    sns.heatmap(confusion_matrix(y_test, rf_preds), annot=True, fmt='d', cmap='Greens')
```

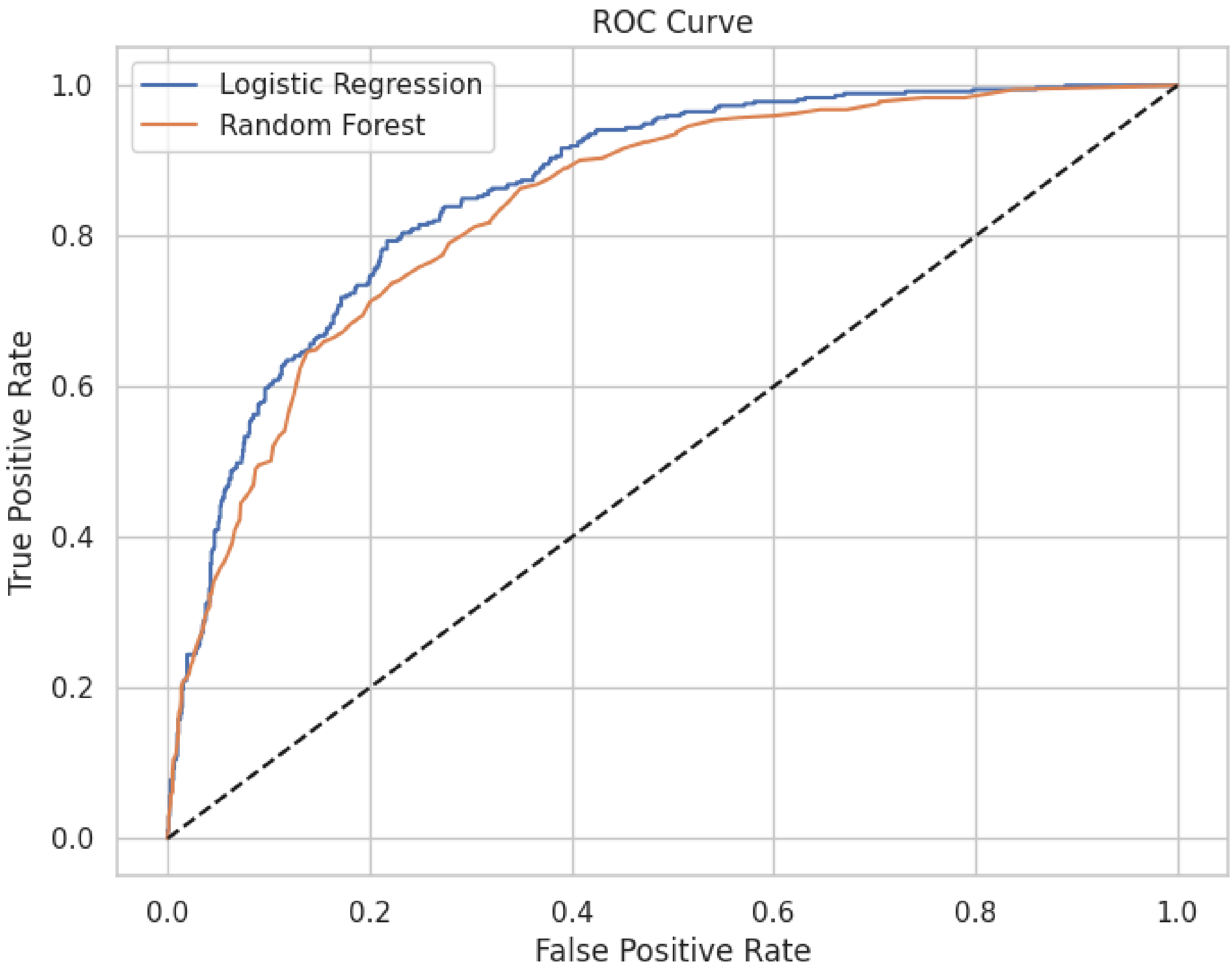
```
plt.title("Confusion Matrix - Random Forest")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# ROC curves
log_fpr, log_tpr, _ = roc_curve(y_test, log_model.predict_proba(X_test)[:, 1])
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_model.predict_proba(X_test)[:, 1])

plt.figure(figsize=(8, 6))
plt.plot(log_fpr, log_tpr, label='Logistic Regression')
plt.plot(rf_fpr, rf_tpr, label='Random Forest')
plt.plot([0, 1], [0, 1], 'k--')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.grid(True)
plt.show()
```

```
<ipython-input-18-84bfb83270b3>:20: FutureWarning: A value is trying to be set on a co
The behavior will change in pandas 3.0. This inplace method will never work because th

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({c


  df['TotalCharges'].fillna(df['TotalCharges'].median(), inplace=True)
Logistic Regression
Accuracy: 0.8211497515968772
F1 Score: 0.64
ROC AUC: 0.8621127351020114
RMSE: 0.42290690276126114
[[933 103]
 [149 224]]

Random Forest
Accuracy: 0.801277501774308
F1 Score: 0.56656346749226
ROC AUC: 0.8385313693624687
RMSE: 0.4457830169776457
[[946  90]
 [190 183]]
```






Actual
1
190
183
0
1
Predicted
700
600
500
400
300
200
100


ROC Curve
Logistic Regression
Random Forest
True Positive Rate
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0