In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler, SVMSMOTE


def split_train_test(df):
    X = df.drop(columns=["Acct Closed?"])
    y = df["Acct Closed?"]
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )
    return X_train, X_test, y_train, y_test


def combine_features(df):
    df["Merged_Minutes_Calls_Peak_Hrs"] = df["Minutes Peak Hrs"] * df["Calls Peak H
    df["Merged_Minutes_Calls_Off_Peak"] = df["Minutes Off Peak"] * df["Calls Off Pe
    df["Merged_Minutes_Calls_Night"] = df["Minutes Night"] * df["Bill Night"]
    df["Merged_Minutes_Calls_Trunk"] = df["Trunk Call Minutes"] * df["Trunk Calls"]
    df["Total_Bill"] = (
        df["Bill Peak Hrs"]
        + df["Bill Off Peak"]
        + df["Bill Night"]
        + df["Trunk Call Bill"]
    )
    return df


def remove_features(df):
    columns_to_remove = [
        "Voice Messaging",
        "Minutes Peak Hrs",
        "Calls Peak Hrs",
        "Minutes Off Peak",
        "Calls Off Peak",
        "Minutes Night",
        "Calls Night",
        "Trunk Call Minutes",
        "Trunk Calls",
        "Bill Peak Hrs",
        "Bill Off Peak",
        "Bill Night",
        "Trunk Call Bill",
    ]

    df = df.drop(columns=columns_to_remove)
    return df


def encode_columns_X(df):
    df = pd.get_dummies(df, columns=["Region"])
    df["Trunk Calling Facility"] = df["Trunk Calling Facility"].map({"No": 0, "Yes'
    return df
```

```python
def encode_column_y(df):
    df = df.astype("int")
    return df


def handle_outliers(X, y):
    numeric_columns = X.select_dtypes(include=["float64", "int64"]).columns.differe
        ["Neighborhood", "Contact for Grievances/Changes"]
    )
    threshold = 1.5

    Q1 = X[numeric_columns].quantile(0.25)
    Q3 = X[numeric_columns].quantile(0.75)
    IQR = Q3 - Q1

    outliers_num = np.logical_or(
        X[numeric_columns] < (Q1 - threshold * IQR),
        X[numeric_columns] > (Q3 + threshold * IQR),
    )

    X_filtered = X[~outliers_num.any(axis=1)]
    y_filered = y[~outliers_num.any(axis=1)]

    return X_filtered, y_filered


def perform_scale(X_train, X_test):
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
    return X_train, X_test


def perform_sampling(X, y, sampling_technique):
    sampler = RandomUnderSampler(random_state=42)
    match sampling_technique:
        case "under_sample":
            pass
        case "over_sample":
            sampler = RandomOverSampler(random_state=42)
        case "svmsmote":
            sampler = SVMSMOTE(random_state=42)
    X_resampled, y_resampled = sampler.fit_resample(X, y)
    return X_resampled, y_resampled


def build_model(X, y):
    model = LogisticRegression(
        penalty="l2", solver="lbfgs", max_iter=1023, random_state=500
    )
    model = LogisticRegression()
    model.fit(X, y)
    return model


def show_model_performance(y_test, y_pred):
    accuracy = accuracy_score(y_test, y_pred)
    print(f"\n Accuracy: {accuracy:.2f}")

    print("\n Classification metrics")
    print(classification_report(y_test, y_pred))

    conf_matrix = confusion_matrix(y_test, y_pred)
```

```
    print("\n Confusion Matrix:")
    print(conf_matrix)


def plot_auc_of_roc(model, X_test, y_test):
    # Predict probabilities for the positive class
    y_prob = model.predict_proba(X_test)[:, 1]

    # Calculate the ROC curve
    fpr, tpr, thresholds = roc_curve(y_test, y_prob)

    # Calculate the AUC
    roc_auc = auc(fpr, tpr)

    # Plot the ROC curve
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color="darkorange", lw=2, label=f"AUC = {roc_auc:.2f}")
    plt.plot([0, 1], [0, 1], color="navy", lw=2, linestyle="--")
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("Receiver Operating Characteristic (ROC) Curve")
    plt.legend(loc="lower right")
    plt.show()


def main():
    df_path = "~/code/loyalist-college/ml-1/assignment-3/UserRetentionData.csv"
    original_df = pd.read_csv(df_path)
    df_copy = original_df.copy()

    # Split Train & Test
    X_train, X_test, y_train, y_test = split_train_test(df_copy)

    # Feature Engineering Train
    X_test, y_test = handle_outliers(X_test, y_test)
    X_train = combine_features(X_train)
    X_train = remove_features(X_train)
    X_train = encode_columns_X(X_train)
    y_train = encode_column_y(y_train)

    # Feture Engineering Test
    X_test = combine_features(X_test)
    X_test = remove_features(X_test)
    X_test = encode_columns_X(X_test)
    y_test = encode_column_y(y_test)

    # Scaling
    X_train, X_test = perform_scale(X_train, X_test)

    # Sampling
    X_train, y_train = perform_sampling(X_train, y_train, "svmsmote")

    # Build Model
    model = build_model(X_train, y_train)

    # prediction test
    y_pred = model.predict(X_test)

    show_model_performance(y_test, y_pred)
    plot_auc_of_roc(model, X_test, y_test)


main()
```

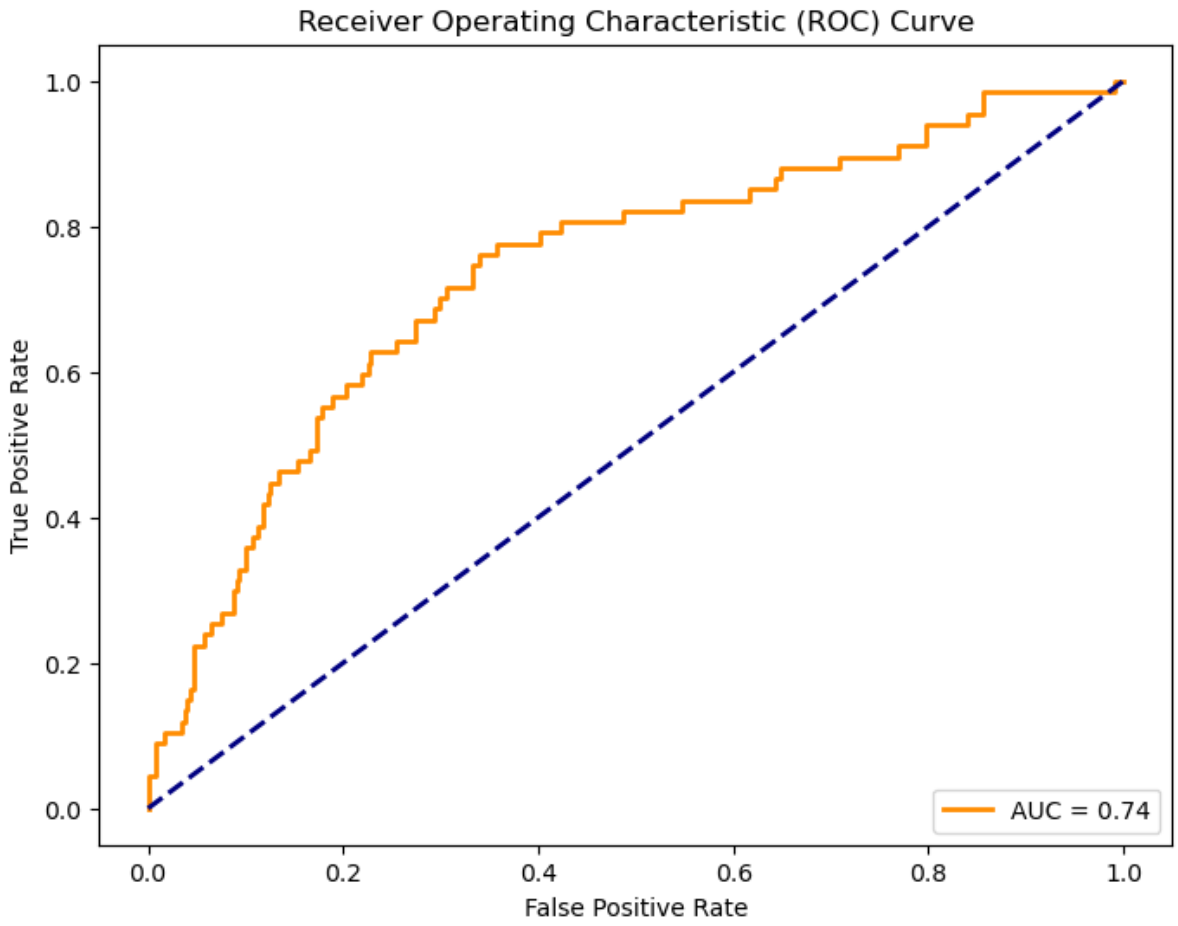
```
Accuracy: 0.79

Classification metrics
              precision    recall  f1-score   support

           0       0.91      0.84      0.87       398
           1       0.34      0.48      0.40        67

    accuracy                           0.79       465
   macro avg       0.62      0.66      0.64       465
weighted avg       0.82      0.79      0.81       465


Confusion Matrix:
[[336  62]
 [ 35  32]]
```




In [ ]: