import pandas as pd

from sklearn.model\_selection import train\_test\_split, cross\_val\_predict

from sklearn.metrics import accuracy\_score, confusion\_matrix, roc\_curve, auc,classification\_report

from sklearn.naive\_bayes import GaussianNB

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

from imblearn.over\_sampling import SMOTE

# Load the dataset

data = pd.read\_csv("updated\_data.csv")

# Encode categorical features

label\_encoders = {}

for column in data.select\_dtypes(include=['object']).columns:

le = LabelEncoder()

data[column] = le.fit\_transform(data[column])

label\_encoders[column] = le

data.drop('CLIENTNUM', axis=1, inplace=True)

data.drop('Naive\_Bayes\_Classifier\_Attrition\_Flag\_Card\_Category\_Contacts\_Count\_12\_mon\_Dependent\_count\_Education\_Level\_Months\_Inactive\_12\_mon\_1',axis=1,inplace=True)

data.drop('Naive\_Bayes\_Classifier\_Attrition\_Flag\_Card\_Category\_Contacts\_Count\_12\_mon\_Dependent\_count\_Education\_Level\_Months\_Inactive\_12\_mon\_2',axis=1,inplace=True)

# Display the remaining columns

print(data.columns)

# Define features and target

X = data.drop(columns=['Attrition\_Flag'])

y = data['Attrition\_Flag']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42, stratify=y)

# Handle imbalance with SMOTE (Synthetic Minority Oversampling Technique).

smote = SMOTE(random\_state=42)

X\_train\_balanced, y\_train\_balanced = smote.fit\_resample(X\_train, y\_train)

#Train a Naive Bayes classifier

model = GaussianNB()

model.fit(X\_train\_balanced, y\_train\_balanced)

# Make predictions

y\_pred = model.predict(X\_test)

# Check accuracy score

accuracy\_gnb = accuracy\_score(y\_test, y\_pred)

print(f"GaussianNB Accuracy: {accuracy\_gnb}")

# Confusion matrix with cross-validation

y\_pred\_cv = cross\_val\_predict(model, X, y, cv=58)

conf\_matrix = confusion\_matrix(y, y\_pred\_cv)

print("Confusion Matrix (GaussianNB with cross-validation):")

print(conf\_matrix)

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

# ROC curve

fpr, tpr, \_ = roc\_curve(y, y\_pred\_cv)

roc\_auc = auc(fpr, tpr)

# Plot ROC Curve

plt.figure()

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {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()

# Interpretation based on AUC value

if roc\_auc == 1.0:

interpretation = "The model achieves perfect classification with no errors."

elif roc\_auc > 0.8:

interpretation = "The model effectively distinguishes between classes with high accuracy."

elif 0.7 <= roc\_auc <= 0.8:

interpretation = "The model shows a reasonable ability to separate the classes."

elif 0.6 <= roc\_auc < 0.7:

interpretation = "The model has limited discriminative power and may require improvement."

elif 0.5 <= roc\_auc < 0.6:

interpretation = "The model struggles to differentiate between classes, similar to random guessing."

else:

interpretation = "The model is no better than random guessing and requires significant improvement."

print(f"AUC: {roc\_auc:.2f} - {interpretation}")

**Output:**









