```
import pandas as pd from sklearn.model_selection import train_test_split from
sklearn.ensemble import RandomForestClassifier from sklearn.metrics import
classification_report, roc_auc_score, roc_curve from sklearn.preprocessing import
LabelEncoder from sklearn.impute import SimpleImputer from sklearn.preprocessing import
StandardScaler import matplotlib.pyplot as plt

import seaborn as sns df = pd.read_csv('machine.csv') df.columns = df.columns.str.strip()
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns
categorical_columns = df.select_dtypes(include=['object']).columns

numerical_imputer = SimpleImputer(strategy='mean')  # Use mean for numerical columns

df[numerical_columns] =

numerical_imputer.fit_transform(df[numerical_columns])

categorical_imputer = SimpleImputer(strategy='most_frequent')  # Use most frequent for
categorical columns

df[categorical_columns] =

categorical_imputer.fit_transform(df[categorical_columns]) label_encoder = LabelEncoder() for
col in categorical_columns:

   df[col] = label_encoder.fit_transform(df[col]) if 'Exited' in df.columns:

   X = df.drop('Exited', axis=1)  # Features     y = df['Exited']  # Target (Exited) else:

   raise KeyError("'Exited' column not found in the dataset")

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train) X_test = scaler.transform(X_test) model =
RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train) y_pred = model.predict(X_test) print("Classification Report:")
print(classification_report(y_test, y_pred)) roc_auc = roc_auc_score(y_test,
model.predict_proba(X_test)[:, 1]) print(f"ROC-AUC Score: {roc_auc}") fpr, tpr, thresholds =
roc_curve(y_test, model.predict_proba(X_test)[:, 1]) plt.figure(figsize=(10, 6)) plt.plot(fpr, tpr,
color='blue', label=f'ROC Curve (AUC = {roc_auc:.2f})') plt.plot([0, 1], [0, 1], color='gray',
linestyle='--') plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate') plt.title('ROC Curve')
plt.legend(loc='lower right') plt.show() importances = model.feature_importances_ indices =
X.columns

plt.figure(figsize=(10, 6)) sns.barplot(x=importances, y=indices, palette='viridis')
plt.title('Feature Importances') plt.xlabel('Importance') plt.ylabel('Feature') plt.show()

new_customer = [[700, 0, 40, 2, 50000, 1, 1, 5, 20000, 2, 1, 1, 45000]]  # Replace with
appropriate values for all 13 features new_customer_scaled = scaler.transform(new_customer)
predicted_exited = model.predict(new_customer_scaled) print(f"Predicted Exited: {'Yes' if
predicted_exited == 1 else 'No'}")
```