In [36]:
```python
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
df=pd.read_excel('F:\DS assignment\DS-Assignment Dataset and instructions\P3- C
```

In [3]:
```python
df.head(10)
```

Out[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	France	Female	42	2	0.00
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86
2	3	15619304	Onio	502	France	Female	42	8	159660.80
3	4	15701354	Boni	699	France	Female	39	1	0.00
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78
6	7	15592531	Bartlett	822	France	Male	50	7	0.00
7	8	15656148	Obinna	376	Germany	Female	29	4	115046.74
8	9	15792365	He	501	France	Male	44	4	142051.07
9	10	15592389	H?	684	France	Male	27	2	134603.88

In [4]:
```python
df.tail()
```

Out[4]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
9995	9996	15606229	Obijiaku	771	France	Male	39	5	(
9996	9997	15569892	Johnstone	516	France	Male	35	10	5736(
9997	9998	15584532	Liu	709	France	Female	36	7	(
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	7507!
9999	10000	15628319	Walker	792	France	Female	28	4	13014:

In [5]:
```python
df.shape
```

Out[5]:
```
(10000, 14)
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
churned            0
dtype: int64
```

In [7]:
```
df.describe()
```

Out[7]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	Nur
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	1
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	

In [8]:
```
df.columns
```

Out[8]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'churned'],
      dtype='object')
```

1. Customer Demographics:

Q. What is the distribution of customers across different age groups?

In [11]:
```
age_distribution = df['Age'].value_counts().sort_index()

print("Distribution of Customers Across Different Age Groups:")
print(age_distribution)
```

```
Distribution of Customers Across Different Age Groups:
Age
18    22
19    27
20    40
21    53
22    84
      ..
83     1
84     2
85     1
88     1
92     2
Name: count, Length: 70, dtype: int64
```

Q. Analyze the gender distribution of customers?

In [12]:
```
gender_distribution = df['Gender'].value_counts()

print("Gender Distribution of Customers:")
print(gender_distribution)
```

```
Gender Distribution of Customers:
Gender
Male      5457
Female    4543
Name: count, dtype: int64
```

2. CHURN ANALYSIS

Q. What percentage of customers have churned?

In [13]:
```
churn_percentage = (df['churned'].sum() / len(df)) * 100

print(f"Percentage of Customers who have churned: {churn_percentage:.2f}%")
```

```
Percentage of Customers who have churned: 20.37%
```

Q. What are the main reasons for customer churn?

In [14]:

```
print("Data Types of Columns:")
print(df.dtypes)

numeric_columns = df.select_dtypes(include=['int64', 'float64']).columns

# Calculate correlations with 'churned'
correlations = df[numeric_columns].corrwith(df['churned']).abs().sort_values(as

print("\nTop Factors Correlated with Churn:")
print(correlations)

geography_counts = df['Geography'].value_counts()
gender_counts = df['Gender'].value_counts()

print("\nGeography Distribution:")
print(geography_counts)

print("\nGender Distribution:")
print(gender_counts)
```

```
Data Types of Columns:
RowNumber             int64
CustomerId            int64
Surname              object
CreditScore           int64
Geography            object
Gender               object
Age                   int64
Tenure                int64
Balance             float64
NumOfProducts         int64
HasCrCard             int64
IsActiveMember        int64
EstimatedSalary     float64
churned               int64
dtype: object

Top Factors Correlated with Churn:
churned            1.000000
Age                0.285323
IsActiveMember     0.156128
Balance            0.118533
NumOfProducts      0.047820
CreditScore        0.027094
RowNumber          0.016571
Tenure             0.014001
EstimatedSalary    0.012097
HasCrCard          0.007138
CustomerId         0.006248
dtype: float64

Geography Distribution:
Geography
France     5014
Germany    2509
Spain      2477
Name: count, dtype: int64

Gender Distribution:
Gender
Male      5457
Female    4543
Name: count, dtype: int64
```

Q. Identify any patterns or trends among customers who have churned.

In [16]:
```
import seaborn as sns
sns.set_theme()  # Reset Seaborn defaults
```

In [17]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


df = pd.read_excel('F:/DS assignment/DS-Assignment Dataset and instructions/P3-

# Filter churned customers
churned_df = df[df['churned'] == 1]
```

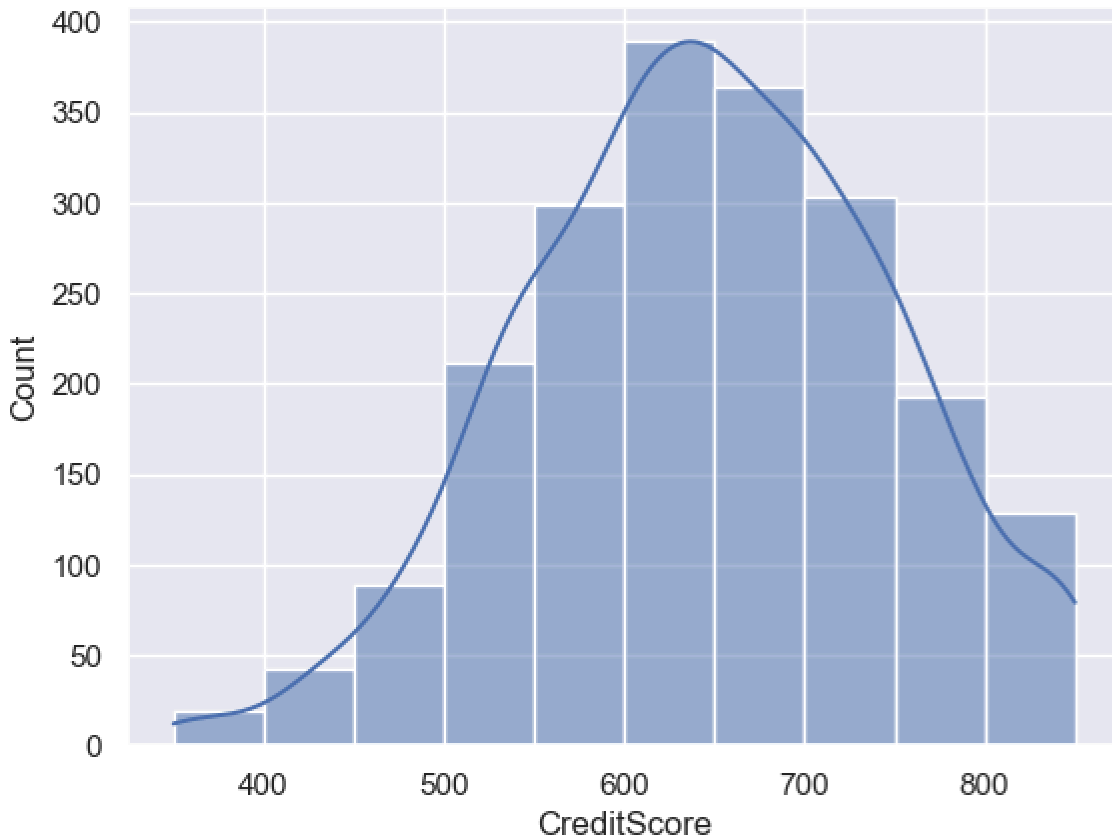
In [20]:
```
churned_df['CreditScore'].value_counts()
```

Out[20]:
```
CreditScore
850    43
651    17
705    16
637    14
727    13
       ..
821     1
733     1
804     1
407     1
486     1
Name: count, Length: 420, dtype: int64
```

In [23]:
```
churned_df['CreditScore']
```

Out[23]:
```
0       619
2       502
5       645
7       376
16      653
       ...
9981    498
9982    655
9991    597
9997    709
9998    772
Name: CreditScore, Length: 2037, dtype: int64
```

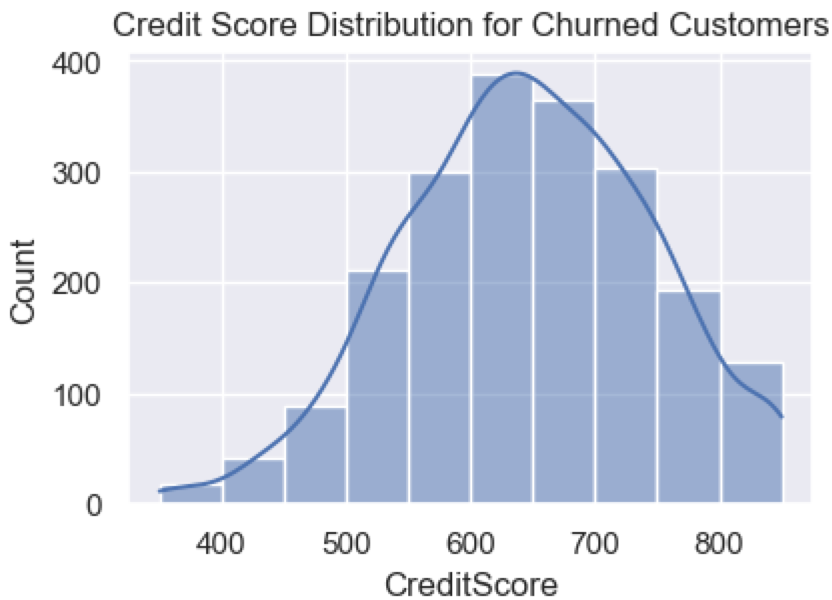
In [23]:
```
sns.histplot(churned_df['CreditScore'],bins = 10 ,kde = True);
```



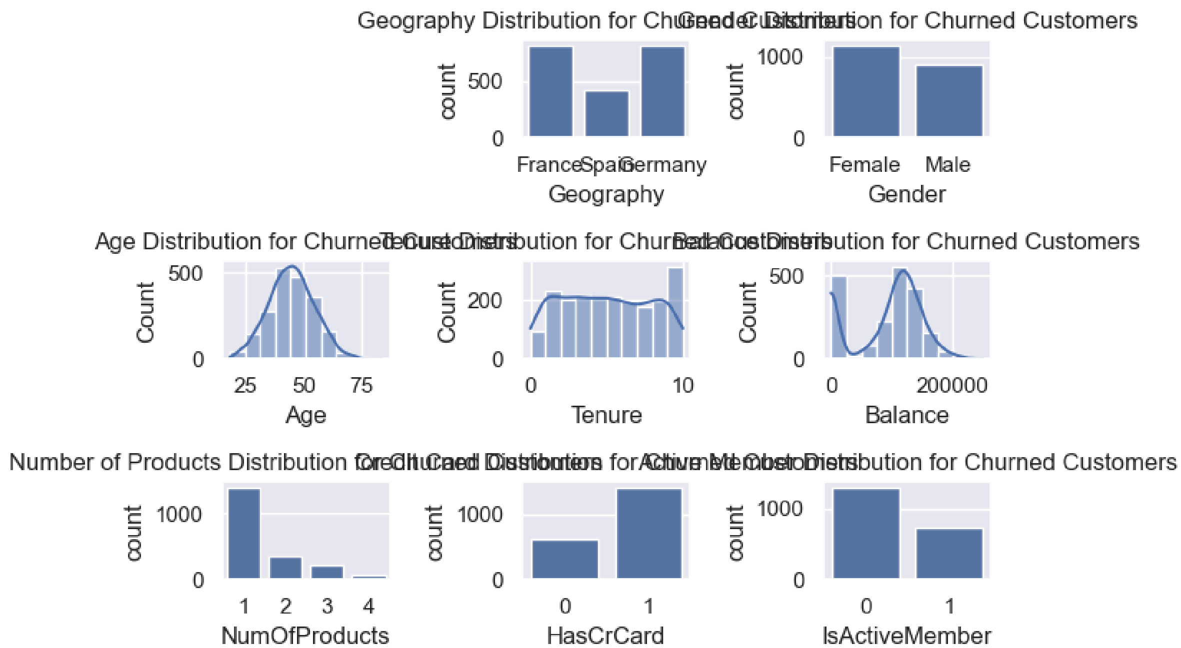
Q. Plotting the distribution of different features for churned customers

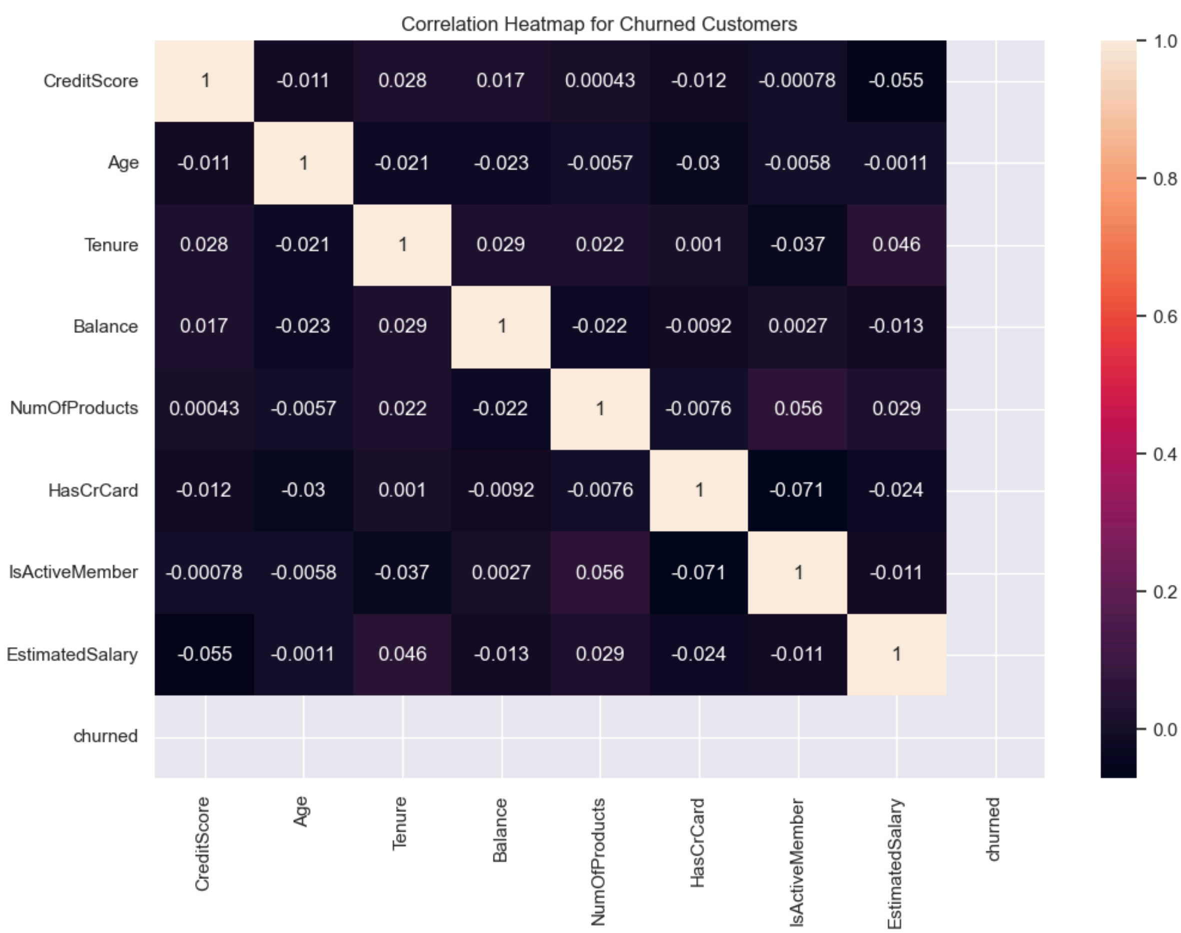
In [24]:
```
# Plotting the distribution of different features for churned customers
plt.figure(figsize=(15, 10))

# Credit Score distribution
plt.subplot(3, 3, 1)
sns.histplot(churned_df['CreditScore'], bins=10, kde=True)
plt.title('Credit Score Distribution for Churned Customers')
```

Out[24]:
```
Text(0.5, 1.0, 'Credit Score Distribution for Churned Customers')
```

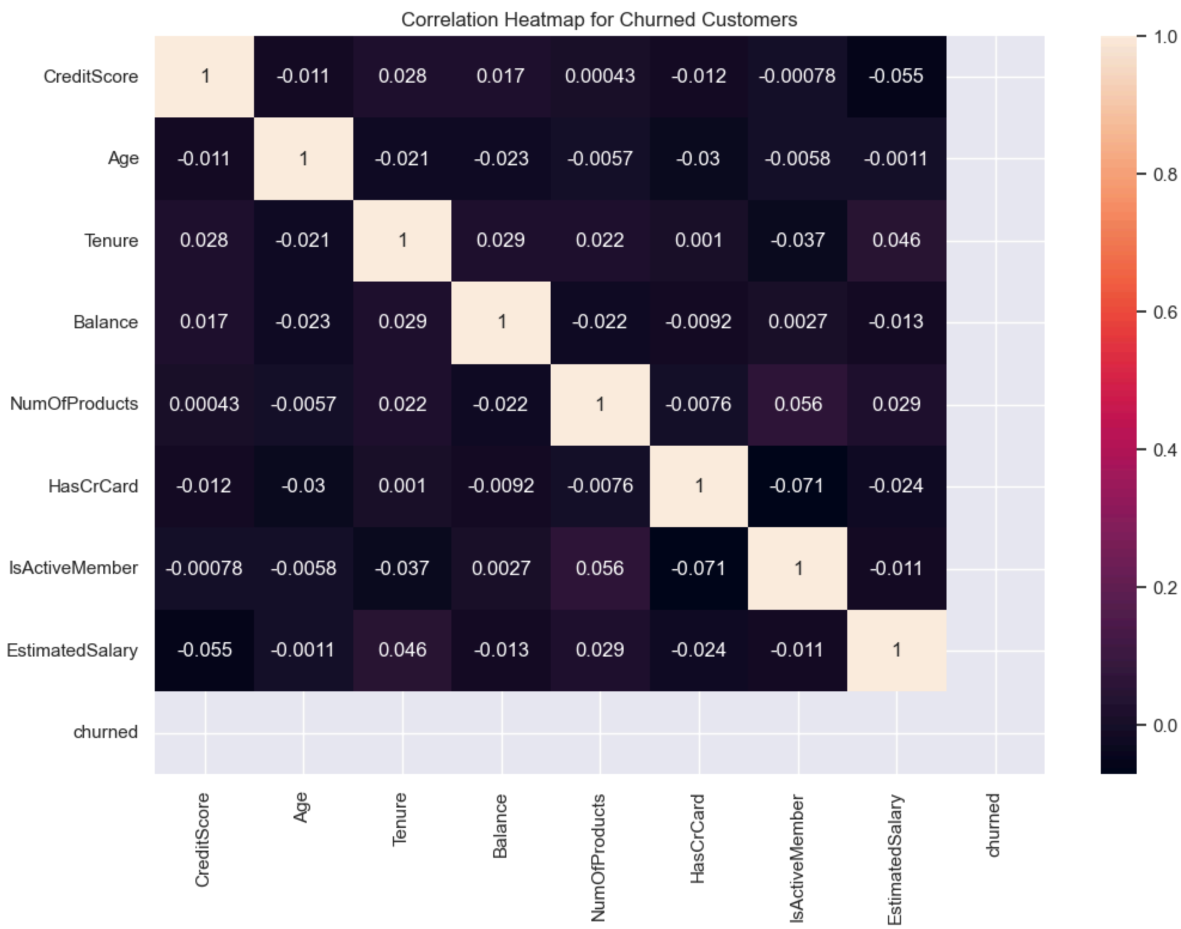


In [35]:
```
# Geography distribution
plt.subplot(3, 3, 2)
sns.countplot(x='Geography', data=churned_df)
plt.title('Geography Distribution for Churned Customers')

# Gender distribution
plt.subplot(3, 3, 3)
sns.countplot(x='Gender', data=churned_df)
plt.title('Gender Distribution for Churned Customers')

# Age distribution
plt.subplot(3, 3, 4)
sns.histplot(churned_df['Age'], bins=10, kde=True)
plt.title('Age Distribution for Churned Customers')

# Tenure distribution
plt.subplot(3, 3, 5)
sns.histplot(churned_df['Tenure'], bins=10, kde=True)
plt.title('Tenure Distribution for Churned Customers')

# Balance distribution
plt.subplot(3, 3, 6)
sns.histplot(churned_df['Balance'], bins=10, kde=True)
plt.title('Balance Distribution for Churned Customers')

# Number of Products distribution
plt.subplot(3, 3, 7)
sns.countplot(x='NumOfProducts', data=churned_df)
plt.title('Number of Products Distribution for Churned Customers')

# Credit Card distribution
plt.subplot(3, 3, 8)
sns.countplot(x='HasCrCard', data=churned_df)
plt.title('Credit Card Distribution for Churned Customers')

# Active Member distribution
plt.subplot(3, 3, 9)
sns.countplot(x='IsActiveMember', data=churned_df)
plt.title('Active Member Distribution for Churned Customers')

plt.tight_layout()
plt.show()

# Correlation heatmap to identify potential patterns
plt.figure(figsize=(12, 8))
sns.heatmap(churned_df[['CreditScore','Age','Tenure','Balance','NumOfProducts',
plt.title('Correlation Heatmap for Churned Customers')
plt.show()
```


Geography Distribution for Churned Customers
Gender Distribution for Churned Customers
count
Geography
France Spain Germany
Gender
Female Male
Age Distribution for Churned Customers
Tenure Distribution for Churned Customers
Balance Distribution for Churned Customers
Count
Age
Tenure
Balance
Number of Products Distribution for Churned Customers
Credit Card Distribution for Churned Customers
Active Member Distribution for Churned Customers
NumOfProducts
HasCrCard
IsActiveMember


Correlation Heatmap for Churned Customers
CreditScore 1 -0.011 0.028 0.017 0.00043 -0.012 -0.00078 -0.055
Age -0.011 1 -0.021 -0.023 -0.0057 -0.03 -0.0058 -0.0011
Tenure 0.028 -0.021 1 0.029 0.022 0.001 -0.037 0.046
Balance 0.017 -0.023 0.029 1 -0.022 -0.0092 0.0027 -0.013
NumOfProducts 0.00043 -0.0057 0.022 -0.022 1 -0.0076 0.056 0.029
HasCrCard -0.012 -0.03 0.001 -0.0092 -0.0076 1 -0.071 -0.024
IsActiveMember -0.00078 -0.0058 -0.037 0.0027 0.056 -0.071 1 -0.011
EstimatedSalary -0.055 -0.0011 0.046 -0.013 0.029 -0.024 -0.011 1
churned
CreditScore Age Tenure Balance NumOfProducts HasCrCard IsActiveMember EstimatedSalary churned

In [34]:
```python
# Correlation heatmap to identify potential patterns
plt.figure(figsize=(12, 8))
sns.heatmap(churned_df[['CreditScore','Age','Tenure','Balance','NumOfProducts',
plt.title('Correlation Heatmap for Churned Customers')
plt.show()
```



In [44]:
```
churned_df
```

Out[44]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Ba
0	1	15634602	Hargrave	619	France	Female	42	2	
2	3	15619304	Onio	502	France	Female	42	8	1596
5	6	15574012	Chu	645	Spain	Male	44	8	1137
7	8	15656148	Obinna	376	Germany	Female	29	4	1150
16	17	15737452	Romeo	653	Germany	Male	58	1	1326
...	...	...	...	...	...	...	...	...	
9981	9982	15672754	Burbidge	498	Germany	Male	42	3	1520
9982	9983	15768163	Griffin	655	Germany	Female	46	7	1371
9991	9992	15769959	Ajuluchukwu	597	France	Female	53	4	883
9997	9998	15584532	Liu	709	France	Female	36	7	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	750

2037 rows × 14 columns

3.Product Usage:

Q.What are the most commonly used products or services?

In [52]:
```
product_counts = df['NumOfProducts'].value_counts().sort_index()

# Displaying the most commonly used products or services
most_common_products = product_counts.idxmax()
most_common_count = product_counts.max()

print(f"The most commonly used number of products or services is {most_common_p
print("\nFrequency distribution of products or services:")
print(product_counts)
```

```
The most commonly used number of products or services is 1 with 5084 customer
s.

Frequency distribution of products or services:
NumOfProducts
1    5084
2    4590
3     266
4      60
Name: count, dtype: int64
```

Q.Analyze the usage patterns of different customer segments.

In [53]:
```python
# Function to analyze customer segments by geography
def analyze_by_geography(df):
    geography_analysis = df.groupby('Geography').agg({
        'CustomerId': 'count',
        'NumOfProducts': 'mean',
        'Balance': 'mean',
        'EstimatedSalary': 'mean'
    }).rename(columns={'CustomerId': 'CustomerCount'})

    return geography_analysis

# Function to analyze customer segments by gender
def analyze_by_gender(df):
    gender_analysis = df.groupby('Gender').agg({
        'CustomerId': 'count',
        'NumOfProducts': 'mean',
        'Balance': 'mean',
        'EstimatedSalary': 'mean'
    }).rename(columns={'CustomerId': 'CustomerCount'})

    return gender_analysis

# Function to analyze customer segments by age group
def analyze_by_age_group(df, bins, labels):
    df['AgeGroup'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)
    age_group_analysis = df.groupby('AgeGroup').agg({
        'CustomerId': 'count',
        'NumOfProducts': 'mean',
        'Balance': 'mean',
        'EstimatedSalary': 'mean'
    }).rename(columns={'CustomerId': 'CustomerCount'})

    return age_group_analysis

# Defining age bins and labels
age_bins = [0, 30, 40, 50, 60]
age_labels = ['<30', '30-39', '40-49', '50-59']

# Perform analysis
geography_analysis = analyze_by_geography(df)
gender_analysis = analyze_by_gender(df)
age_group_analysis = analyze_by_age_group(df, age_bins, age_labels)

# Display the analysis
print("Geography Analysis:\n", geography_analysis)
print("\nGender Analysis:\n", gender_analysis)
print("\nAge Group Analysis:\n", age_group_analysis)
```

```
Geography Analysis:
           CustomerCount  NumOfProducts        Balance  EstimatedSalary
Geography
France              5014       1.530913   62092.636516     99899.180814
Germany             2509       1.519729  119730.116134    101113.435102
Spain               2477       1.539362   61818.147763     99440.572281

Gender Analysis:
        CustomerCount  NumOfProducts       Balance  EstimatedSalary
Gender
Female           4543       1.544134  75659.369139    100601.541382
Male             5457       1.518600  77173.974506     99664.576931

Age Group Analysis:
          CustomerCount  NumOfProducts       Balance  EstimatedSalary
AgeGroup
<30                1641       1.556977  73698.718635    100855.247818
30-39              4346       1.539347  75071.796781     98616.317653
40-49              2618       1.518717  78479.240768    103543.082063
50-59               869       1.481013  83632.942486     97144.930759
```

4.Financial Analysis:

Q. What is the average account balance of customers?

In [54]:

```python
average_balance = df['Balance'].mean()

print(f"The average account balance of customers is {average_balance:.2f}")
```

```
The average account balance of customers is 76485.89
```

Q. Compare the financial characteristics of churned vs. non-churned customers.

In [55]:
```python
def analyze_by_churn_status(df):
    churn_analysis = df.groupby('churned').agg({
        'Balance': 'mean',
        'EstimatedSalary': 'mean',
        'NumOfProducts': 'mean'
    }).rename(index={0: 'Non-Churned', 1: 'Churned'})

    return churn_analysis


churn_analysis = analyze_by_churn_status(df)


print("Financial Characteristics of Churned vs. Non-Churned Customers:\n")
print(churn_analysis)
```

```
Financial Characteristics of Churned vs. Non-Churned Customers:

                  Balance  EstimatedSalary  NumOfProducts
churned
Non-Churned  72745.296779     99738.391772       1.544267
Churned      91108.539337    101465.677531       1.475209
```

5. Predictive Modeling

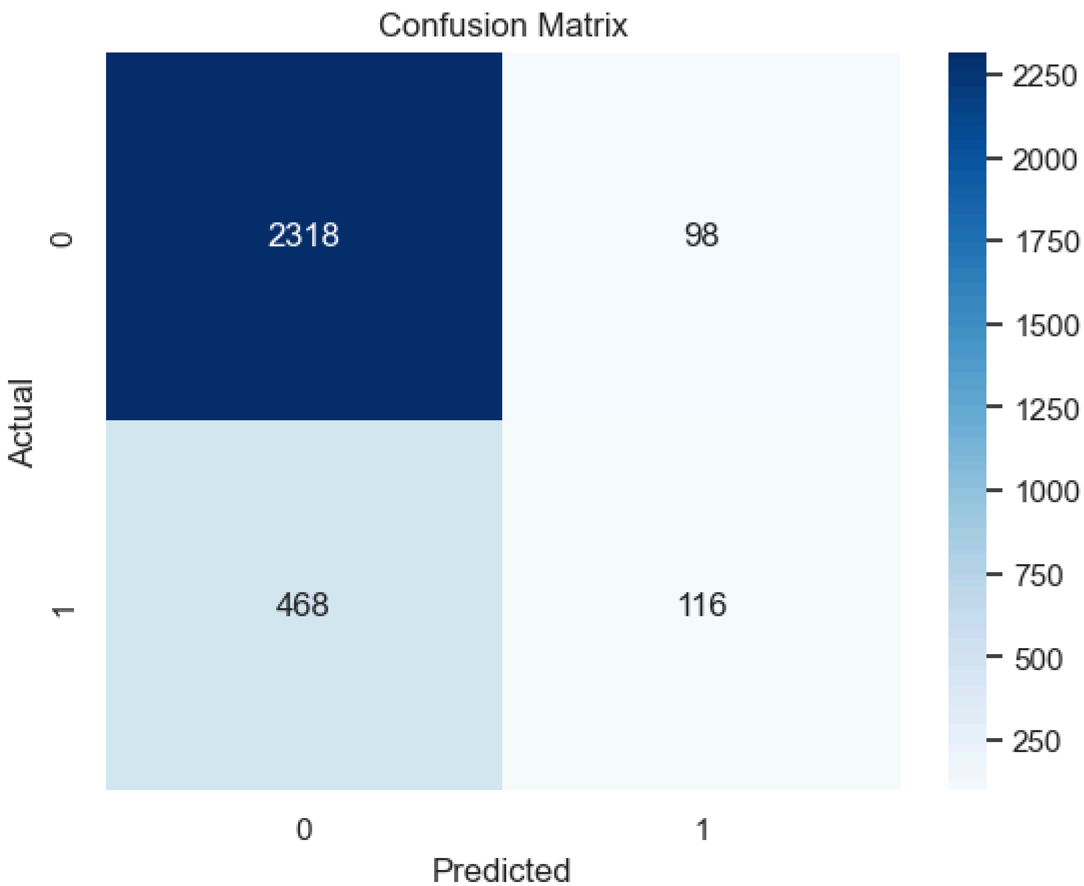
Q.Which factors are the most significant predictors of customer churn?

In [59]:

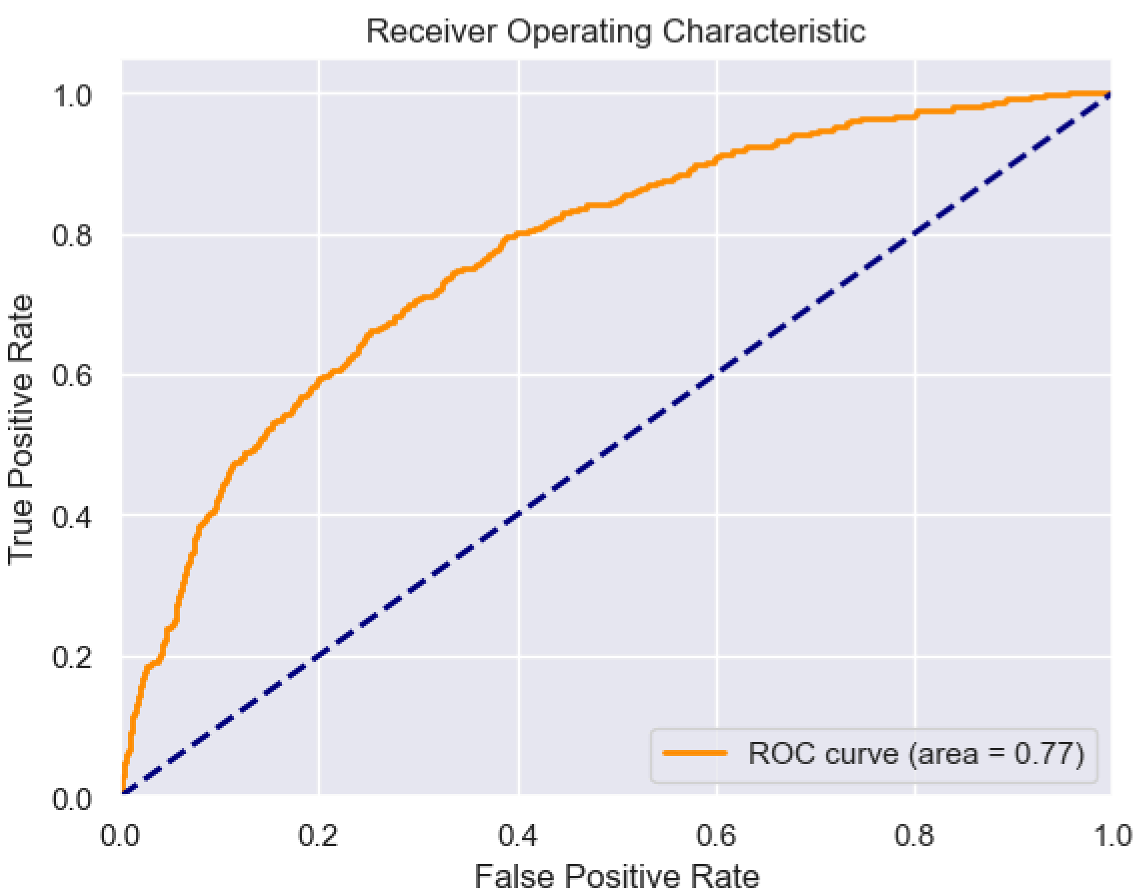
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_
import seaborn as sns
import matplotlib.pyplot as plt


# Define the feature matrix X and the target vector y
X = df.drop(['RowNumber', 'CustomerId', 'Surname', 'churned', 'AgeGroup'], axis
y = df['churned']

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3,

# Fit the logistic regression model
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)

# Make predictions
y_pred = log_reg.predict(X_test)
y_prob = log_reg.predict_proba(X_test)[:, 1]

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)
conf_matrix = confusion_matrix(y_test, y_pred)

# Print evaluation metrics
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"ROC-AUC Score: {roc_auc:.2f}")
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Plot confusion matrix
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Plot ROC curve
from sklearn.metrics import roc_curve

fpr, tpr, _ = roc_curve(y_test, y_prob)
plt.figure()
```

```python
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' %
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

```
Accuracy: 0.81
Precision: 0.54
Recall: 0.20
ROC-AUC Score: 0.77

Confusion Matrix:
[[2318   98]
 [ 468  116]]

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      2416
           1       0.54      0.20      0.29       584

    accuracy                           0.81      3000
   macro avg       0.69      0.58      0.59      3000
weighted avg       0.78      0.81      0.77      3000
```



Confusion Matrix
Actual
Predicted
0
1
0
1
2318
98
468
116
2250
2000
1750
1500
1250
1000
750
500
250



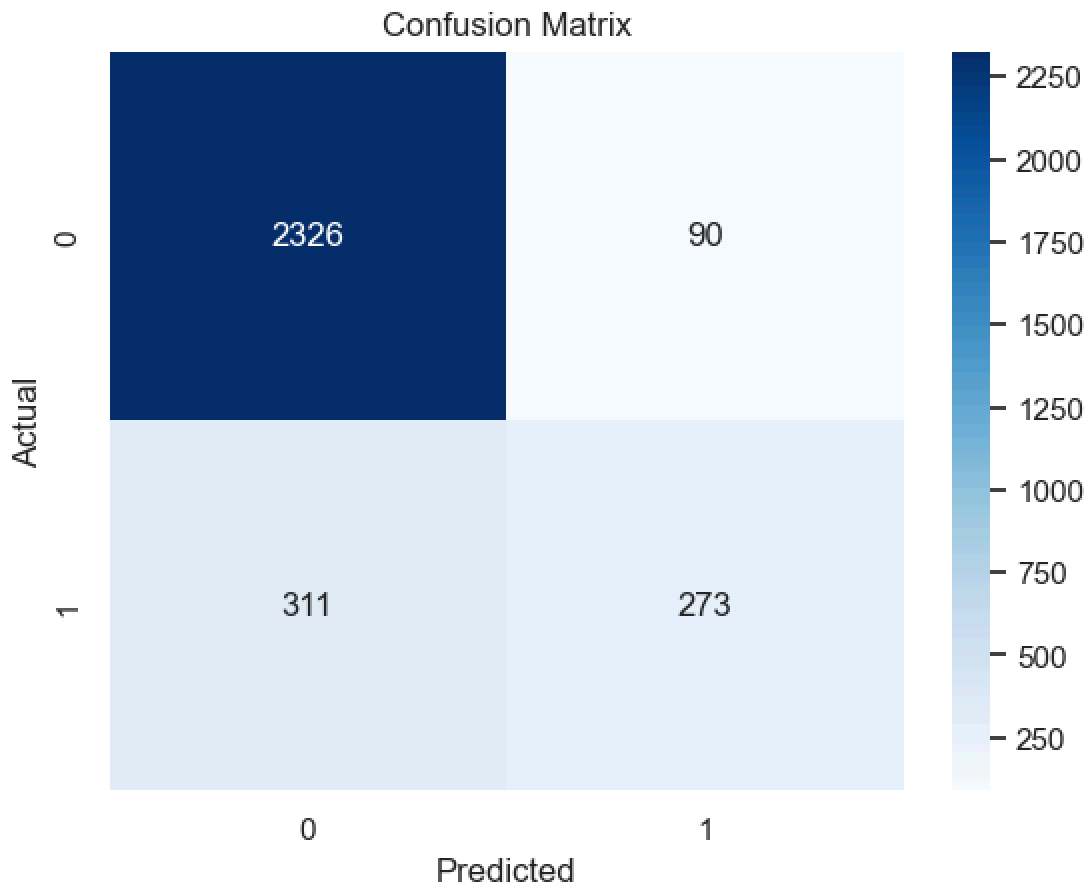
Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.77)

Q.Develop a predictive model to identify at-risk customers

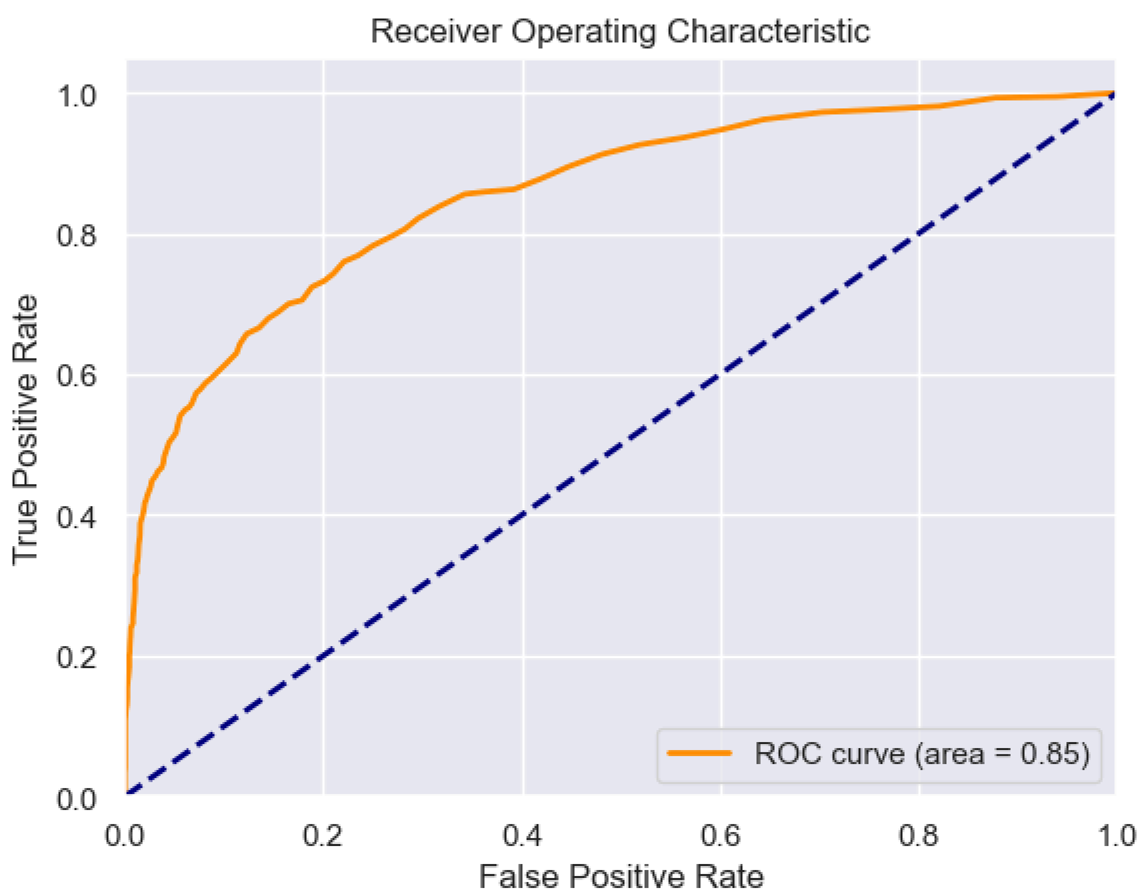
In [60]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_
import seaborn as sns
import matplotlib.pyplot as plt

# Define the feature matrix X and the target vector y
X = df.drop(['RowNumber', 'CustomerId', 'Surname', 'churned', 'AgeGroup'], axis
y = df['churned']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, randon

# Initialize the Random Forest Classifier
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_clf.fit(X_train, y_train)

# Make predictions
y_pred = rf_clf.predict(X_test)
y_prob = rf_clf.predict_proba(X_test)[:, 1]

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)
conf_matrix = confusion_matrix(y_test, y_pred)

# Print evaluation metrics
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"ROC-AUC Score: {roc_auc:.2f}")
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Plot confusion matrix
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Plot ROC curve
from sklearn.metrics import roc_curve

fpr, tpr, _ = roc_curve(y_test, y_prob)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' %
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
```

```python
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()

# Feature importance
feature_importances = pd.Series(rf_clf.feature_importances_, index=X.columns)
feature_importances.nlargest(10).plot(kind='barh')
plt.title('Top 10 Most Important Features')
plt.xlabel('Feature Importance Score')
plt.show()
```

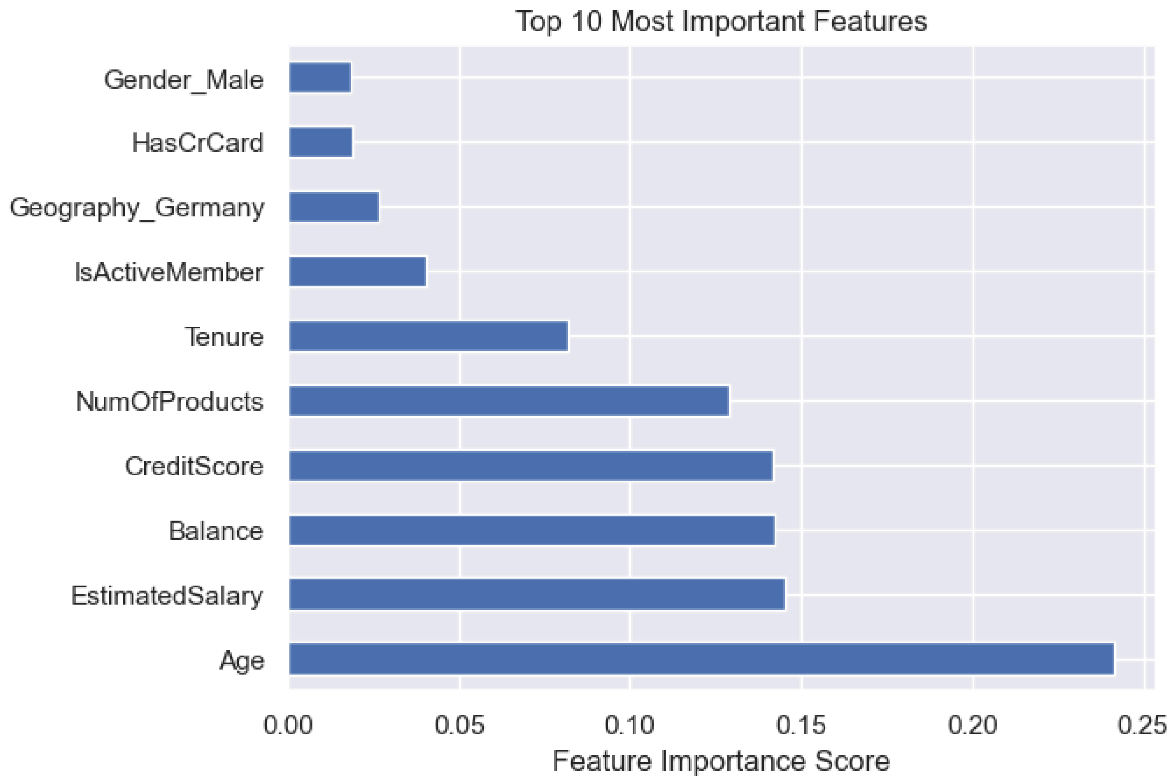
```
Accuracy: 0.87
Precision: 0.75
Recall: 0.47
ROC-AUC Score: 0.85

Confusion Matrix:
[[2326   90]
 [ 311  273]]

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.96      0.92      2416
           1       0.75      0.47      0.58       584

    accuracy                           0.87      3000
   macro avg       0.82      0.72      0.75      3000
weighted avg       0.86      0.87      0.85      3000
```



Confusion Matrix
2326
90
311
273
Actual
0
1
Predicted
2250
2000
1750
1500
1250
1000
750
500
250



Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.85)


Top 10 Most Important Features
Gender_Male
HasCrCard
Geography_Germany
IsActiveMember
Tenure
NumOfProducts
CreditScore
Balance
EstimatedSalary
Age
0.00
0.05
0.10
0.15
0.20
0.25
Feature Importance Score

In [ ]: