```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.utils import resample
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, ConfusionMatrixDisplay
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

warnings.filterwarnings('ignore')


df=pd.read_csv('/content/Churn_Modelling.csv',encoding='Latin-1')
df
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMem
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	

10000 rows × 14 columns

```
df.head()
df.columns = df.columns.str.lower()


df.drop(columns = ['rownumber', 'customerid', 'surname'], inplace = True)


# one-hot encoding categorical values
cat = ['geography', 'gender']

def one_hot(data, col):
    for _ in col:
        data[_] = pd.Categorical(df[_]).codes
    return data

df = one_hot(df, cat)


df
```

	creditscore	geography	gender	age	tenure	balance	numofproducts	hascrcard	isactivemember	estimatedsalary	exited
0	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	608	2	0	41	1	83807.86	1	0	1	112542.58	0

EXPLORATORY DATA ANALYSIS

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   creditscore      10000 non-null  int64  
 1   geography        10000 non-null  int8   
 2   gender           10000 non-null  int8   
 3   age              10000 non-null  int64  
 4   tenure           10000 non-null  int64  
 5   balance          10000 non-null  float64
 6   numofproducts    10000 non-null  int64  
 7   hascrcard        10000 non-null  int64  
 8   isactivemember   10000 non-null  int64  
 9   estimatedsalary  10000 non-null  float64
 10  exited           10000 non-null  int64  
dtypes: float64(2), int64(7), int8(2)
memory usage: 722.8 KB
```

```
df.describe(include = 'all')
```

	creditscore	geography	gender	age	tenure	balance	numofproducts	hascrcard	isactivemember
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000
mean	650.528800	0.746300	0.545700	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100
std	96.653299	0.827529	0.497932	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797
min	350.000000	0.000000	0.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000
25%	584.000000	0.000000	0.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000
50%	652.000000	0.000000	1.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000
75%	718.000000	1.000000	1.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000
max	850.000000	2.000000	1.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000

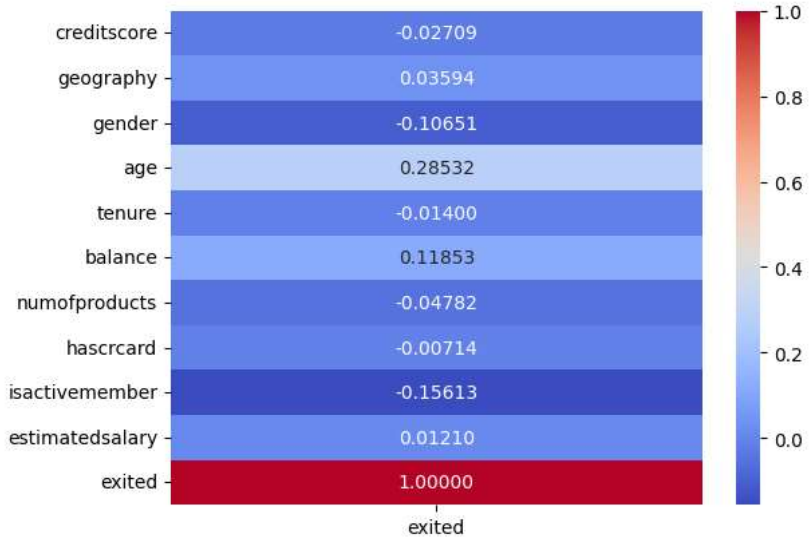
```
print('NA Values: \n',df.isna().sum(),'\n\n Null Values: \n ',df.isnull().sum())
```

```
NA Values:
 creditscore        0
geography          0
gender             0
age                0
tenure             0
balance            0
numofproducts      0
hascrcard          0
isactivemember     0
estimatedsalary    0
exited             0
dtype: int64

 Null Values:
  creditscore        0
geography          0
gender             0
age                0
tenure             0
balance            0
numofproducts      0
hascrcard          0
isactivemember     0
estimatedsalary    0
exited             0
dtype: int64
```

```
def disp_heatmap(df):
    df_corr = df.corr()['exited']
    sns.heatmap(df_corr.to_frame(), annot = True, fmt = '.5f', cmap = 'coolwarm')
```

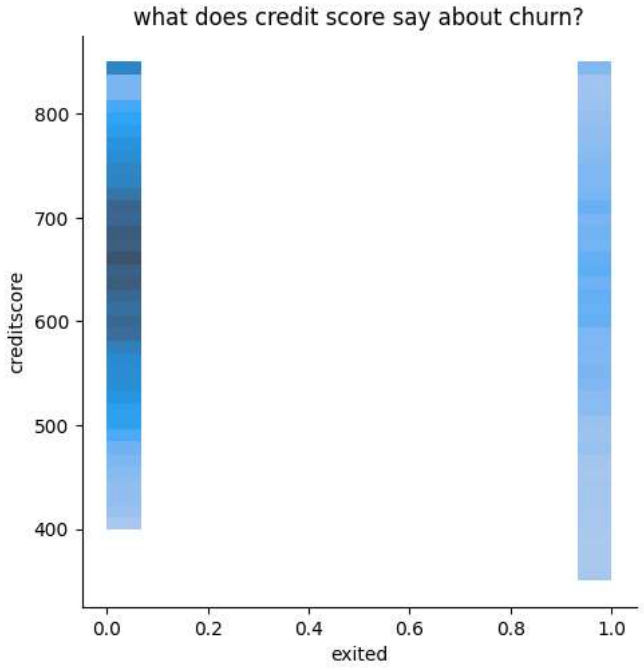
```
disp_heatmap(df)
```




```
exit = df[df['exited']==1]
stay = df[df['exited']==0]
print('Stayed : ', stay.count()[0])
print('Exited : ', exit.count()[0])
```

```
Stayed :  7963
Exited :  2037
```

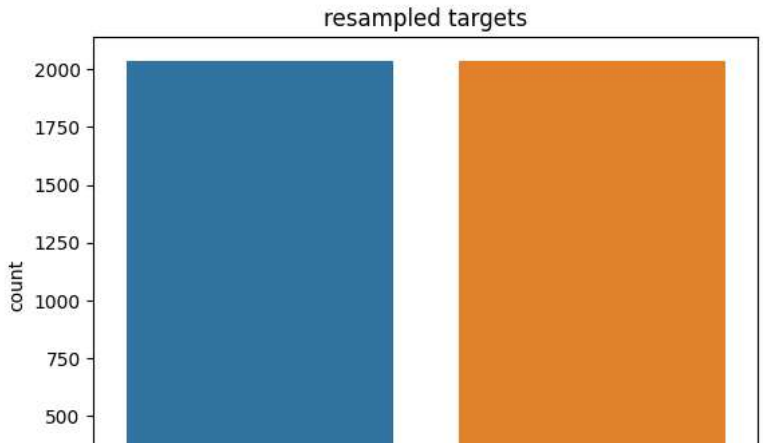
```
sns.displot(x = 'exited',y = 'creditscore', data = df)
plt.title('what does credit score say about churn?')
plt.show()
```



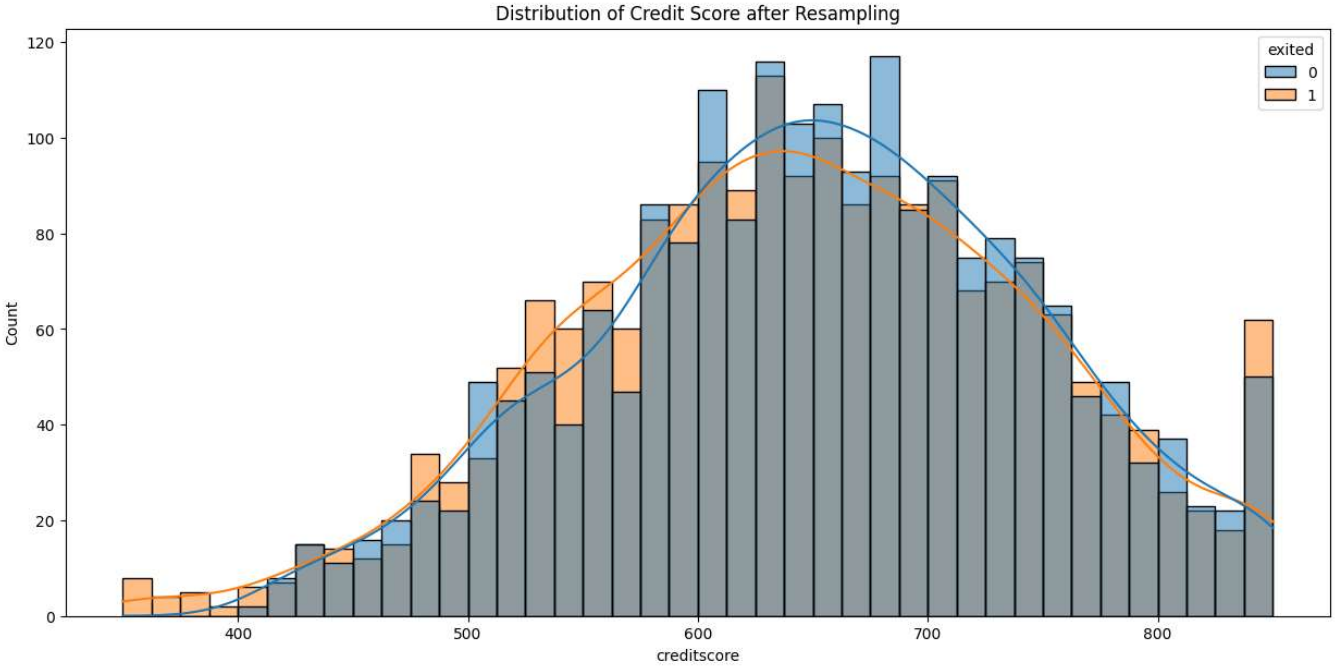
RESAMPLING THE DATA TO REDUCE SKEW-NESS

```
stay_resample = resample(stay, n_samples = exit.count()[0], replace = False, random_state = 42)
df = pd.concat([stay_resample, exit])
```

```
sns.countplot(data = df, x = 'exited')
plt.title('resampled targets')
plt.show()
```




```
plt.figure(figsize = (15,7))
sns.histplot(data = df, x = 'creditscore', hue = 'exited', kde= True, bins = 40 )
plt.title('Distribution of Credit Score after Resampling')
plt.show()
```



FOR SELECTING ONLY TOP 5 FEATURES

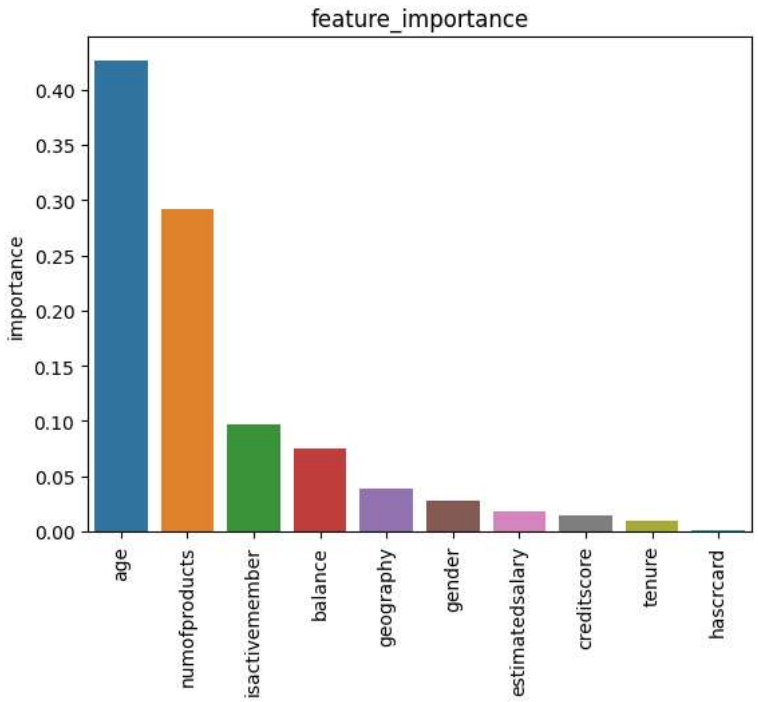
```
y = df['exited']
x = df.drop(columns = 'exited')
```

```
rf = RandomForestClassifier(max_depth = 5)
rf.fit(x, y)
imp = rf.feature_importances_
features = pd.DataFrame(imp, index = x.columns, columns = ['importance']).sort_values('importance', ascending = False)
```

```
features
```

	importance
age	0.426247
numofproducts	0.292001
isactivemember	0.096628
balance	0.075075

```
sns.barplot(y = features.importance, x = features.index)
plt.xticks(rotation = 90)
plt.title('feature_importance')
plt.show()
```



SELECTING ONLY TOP 5

```
x.drop(columns = ['gender', 'creditscore', 'estimatedsalary', 'tenure', 'hascrcard'], inplace = True)
```

CLASSIFYING THE DATA

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, shuffle = True)
```

USING RANDOM FOREST

```
rf = RandomForestClassifier(max_depth = 10, n_estimators = 500)
rf.fit(x_train, y_train)
y_pred = rf.predict(x_test)
```

```
report = classification_report(y_pred, y_test)
print(report)
```

```
                  precision    recall  f1-score   support

               0       0.77      0.75      0.76       624
               1       0.75      0.77      0.76       599

        accuracy                           0.76      1223
       macro avg       0.76      0.76      0.76      1223
    weighted avg       0.76      0.76      0.76      1223
```

```
acc = accuracy_score(y_pred, y_test)
print(f"RF Accuracy: {acc*100}%")
```

```
RF Accuracy: 76.04251839738349%
```

AFTER USING RANDOM FOREST THE ACCURACY IS 76%

Using logistics regression

```
lr = LogisticRegression()
lr.fit(x_train, y_train)
y_pred_lr = lr.predict(x_test)
```

```
report = classification_report(y_pred_lr, y_test)
print(report)
```

```
                   precision    recall  f1-score   support

               0       0.00      0.00      0.00         0
               1       1.00      0.50      0.67      1223

        accuracy                           0.50      1223
       macro avg       0.50      0.25      0.33      1223
    weighted avg       1.00      0.50      0.67      1223
```

```
acc = accuracy_score(y_pred_lr, y_test)
print(f"LR Accuracy: {acc*100}%")
```

```
LR Accuracy: 50.20441537203598%
```

After Using Logostics Regression the Accuracy is 50%