Project: Predicting Churn for Bank Customers

Dan Hua Li

Introduction:

    Customer Churn prediction is for knowing which customers are likely to leave or unsubscribe from your service. This is because acquiring new customers often costs more than retaining existing ones. Once We identified customers at risk of churn, can help to know exactly what marketing efforts We should make with customer to maximize their likelihood of staying. Customer churn prediction can help to make marketing strategies effectively in business. The purpose is to make customer retention increases the customer's average lifetime value, making all future sales more valuable and improving unit margins.

    Dataset source: https://www.kaggle.com/datasets/adammaus/predicting-churn-for-bank-customers

Outcome and predictor attributes:

- Outcome attributes: Exited
- Predictors: this dataset has 10000 entries and 14 columns, and may be selected 3 categories columns and 7 numerical columns. Which are:

  1. Gender—it's interesting to explore whether gender plays a role in a customer leaving the bank.
  2. Age—this is certainly relevant, since older customers are less likely to leave their bank than younger ones.
  3. Tenure—refers to the number of years that the customer has been a client of the bank. Normally, older clients are more loyal and less likely to leave a bank.
  4. Balance—also a very good indicator of customer churn, as people with a higher balance in their accounts are less likely to leave the bank compared to those with lower balances.
  5. NumOfProducts—refers to the number of products that a customer has purchased through the bank.
  6. HasCrCard—denotes whether or not a customer has a credit card. This column is also relevant since people with a credit card are less likely to leave the bank.
  7. IsActiveMember—active customers are less likely to leave the bank.
  8. EstimatedSalary—as with balance, people with lower salaries are more likely to leave the bank compared to those with higher salaries.
  9. Exited—whether or not the customer left the bank.

Importing Libraries & Dataset

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from xgboost import XGBClassifier
from sklearn.metrics import confusion_matrix, recall_score, f1_score, accuracy_score, precision_score
from sklearn.model_selection import train_test_split , cross_val_score, RandomizedSearchCV
import warnings  # ignore all warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
!pip install -U pandas
```

```
Requirement already satisfied: pandas in c:\users\yy\appdata\local\programs\python\python310\lib\site-packages (2.0.0)
Requirement already satisfied: tzdata>=2022.1 in c:\users\yy\appdata\local\programs\python\python310\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: pytz>=2020.1 in c:\users\yy\appdata\local\programs\python\python310\lib\site-packages (from pandas) (2022.2.1)
Requirement already satisfied: numpy>=1.21.0 in c:\users\yy\appdata\local\programs\python\python310\lib\site-packages (from pandas) (1.23.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\yy\appdata\local\programs\python\python310\lib\site-packages (from pandas) (2.8.
2)
Requirement already satisfied: six>=1.5 in c:\users\yy\appdata\local\programs\python\python310\lib\site-packages (from python-dateutil>=2.8.2->pand
as) (1.16.0)
```

```
WARNING: Ignoring invalid distribution -ip (c:\users\yy\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\yy\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\yy\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\yy\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\yy\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\yy\appdata\local\programs\python\python310\lib\site-packages)

[notice] A new release of pip is available: 23.0.1 -> 23.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [3]:
```
df = pd.read_csv('project data- bank1.csv')
```

Data Exploration

In [4]:
```
df.head()
```

Out[4]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42.0	2	0.00	1	1	1	101348.88	1
1	2	15647311	NaN	608	Spain	Female	NaN	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42.0	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39.0	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43.0	2	125510.82	1	1	1	79084.10	0

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          9999 non-null   object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              9999 non-null   float64
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(3), int64(8), object(3)
memory usage: 1.1+ MB
```

Handle missing value

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

Out[6]:
```
RowNumber          0
CustomerId         0
Surname            1
CreditScore        0
Geography          0
Gender             0
Age                1
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

Drop the column 'Surname' and replace the missing value with mean of Age.

In [7]:
```
df=df.drop('Surname',axis=1)
```

In [8]:
```
df['Age'].replace(np.nan, df['Age'].mean(), inplace=True)
df.isnull().sum()
```

Out[8]:

```
RowNumber          0
CustomerId         0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

statistical summary

In [9]:

```
df.describe()
```

Out[9]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921592	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487786	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

In [10]:

```
df.nunique()
```

Out[10]:

```
RowNumber          10000
CustomerId         10000
CreditScore          460
Geography              3
Gender                 2
Age                   71
Tenure                11
Balance             6382
NumOfProducts          4
HasCrCard              2
IsActiveMember         2
EstimatedSalary     9999
Exited                 2
dtype: int64
```

Feature Enginering

We will be dropping the first 3 columns 'RowNumber', 'CustomerId', as it seems usless for further use.

In [11]:

```
df.drop(columns=['RowNumber', 'CustomerId'], axis=1, inplace=True)
df_clean=df
```

In [12]:

```
##checking catigorical columns
df.select_dtypes(include='object').head()
```

Out[12]:

	Geography	Gender
0	France	Female
1	Spain	Female
2	France	Female
3	France	Female
4	Spain	Female

In [13]:

```
print(f'Geography unique',df.Geography.unique())
print(f'Gender unique', df.Gender.unique())
```

```
Geography unique ['France' 'Spain' 'Germany']
Gender unique ['Female' 'Male']
```

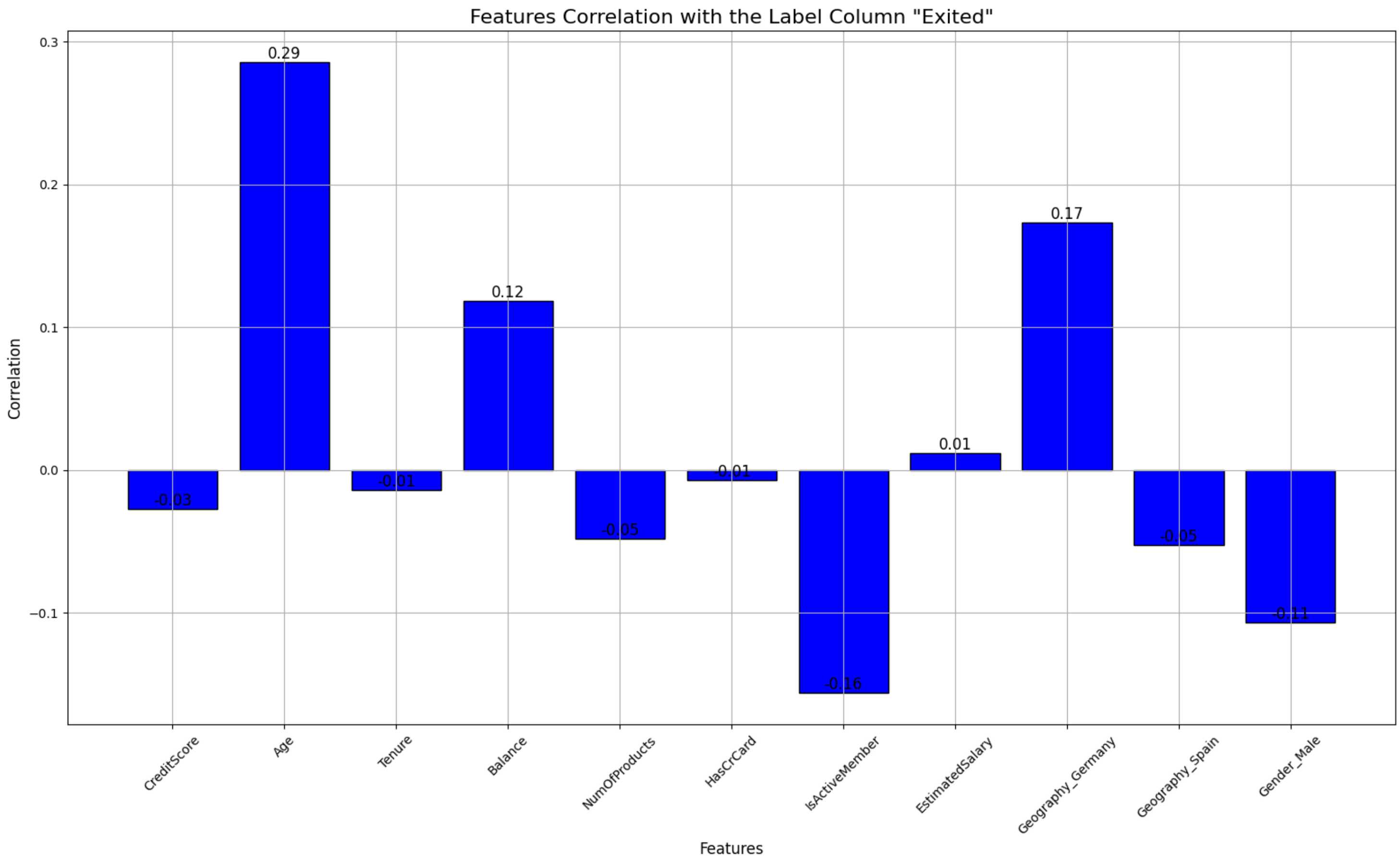
Plot the Corolations between variables

In [14]:

```
#Creating a new variable witout he label column 'Exited'
df_plt = pd.get_dummies(data=df, drop_first=True) # One Hot Encoding for ploting
```

```
df2 = df_plt.drop(columns='Exited') #and droping a unwanted column
```

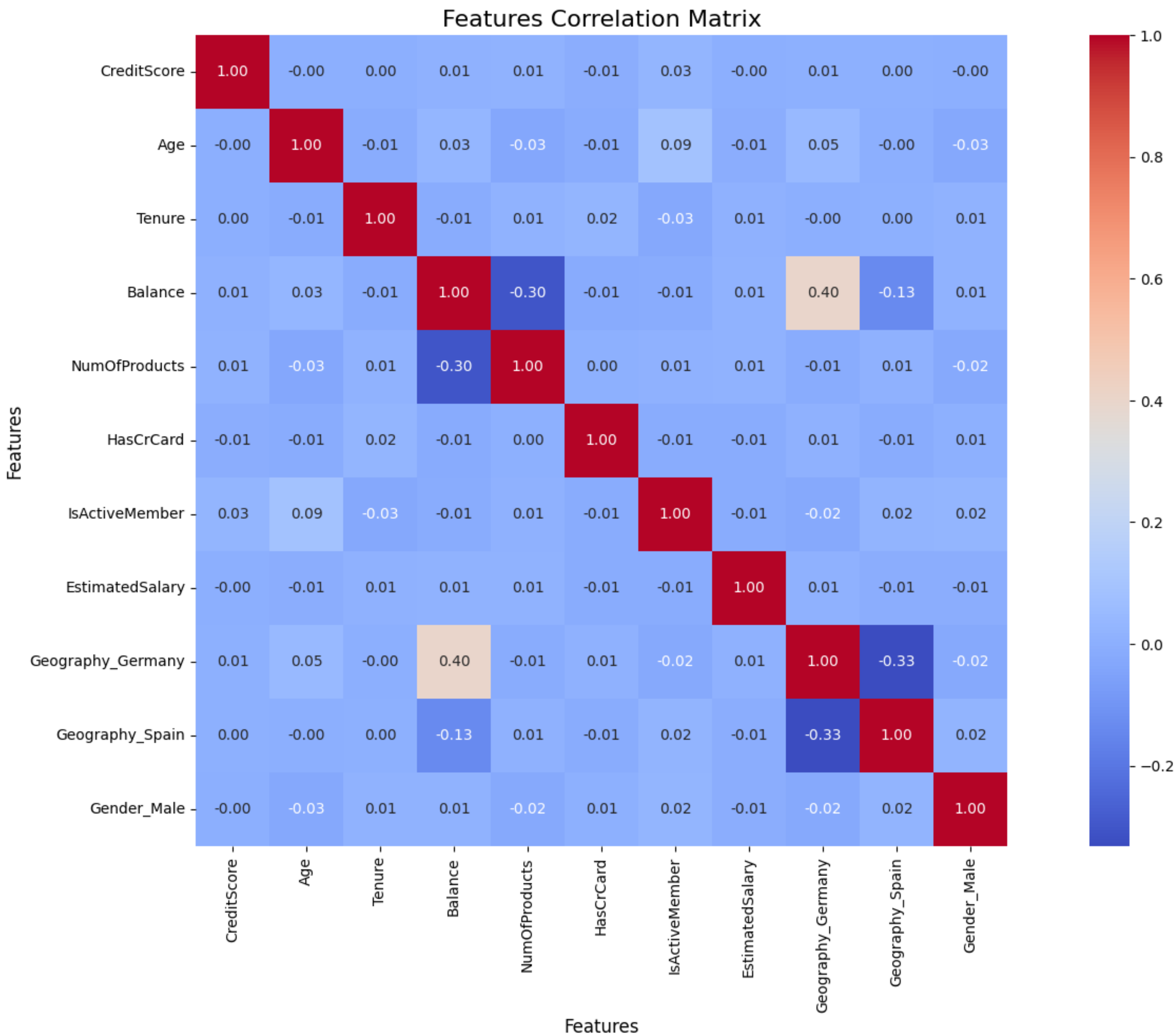
In [15]:
```
# Computing the correlation between the features and the label column
corrw = df2.corrwith(df['Exited'])

# Plotting the bar chart using matplotlib
plt.figure(figsize=(19, 10))
plt.bar(corrw.index, corrw.values, color='blue', edgecolor='black')

# Adding title, labels, and grid to the plot
plt.title('Features Correlation with the Label Column "Exited"', fontsize=16)
plt.xlabel('Features', fontsize=12)
plt.ylabel('Correlation', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True)

# Adding annotations to the bars
for i, value in enumerate(corrw.values):
    label = f"{value:.2f}"
    plt.annotate(label, (i, value), ha='center', va='bottom', fontsize=12)

# Displaying the plot
plt.show()
```



From the correlation plot above, we can see Geography_Germany, Gender_Female are positively related to Exited, whereas IsActiveMember, Gender_Male, Geography_France, Geography_Spain are negatively related to Exited. HasCrCard has little correlation with Exited.

In [54]:
```
# Plotting a Heatmap to see in depth corrolation
corr = df2.corr()  #Defining a corrolation variavle (Corrolation Matrix) corr = df.corr()
plt.figure(figsize=(20,10))
sns.heatmap(corr, annot=True, cmap='coolwarm', cbar=True, fmt='.2f', square=True)

# Adding title and labels to the plot
plt.title('Features Correlation Matrix', fontsize=16)
plt.xlabel('Features', fontsize=12)
plt.ylabel('Features', fontsize=12)
```

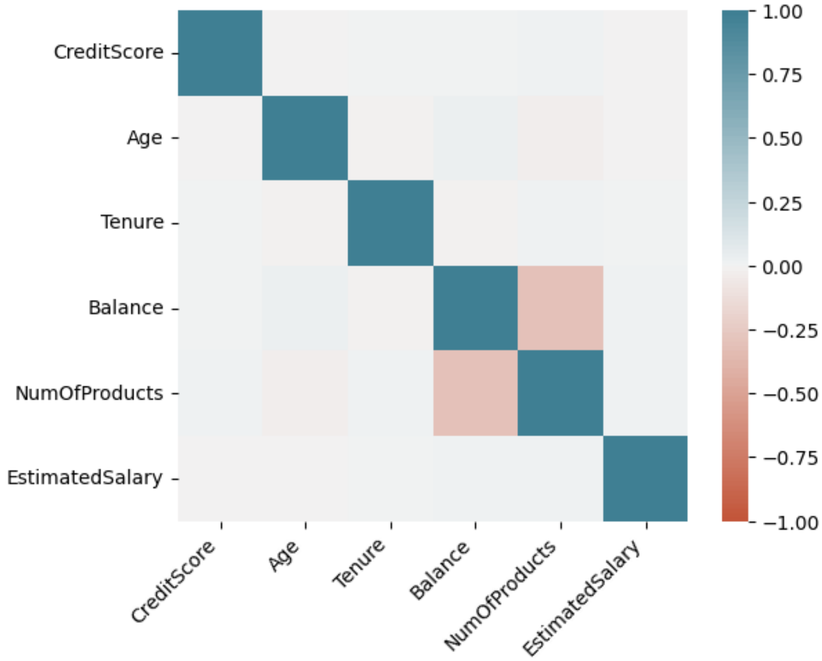
Out[54]:
```
Text(690.5815972222225, 0.5, 'Features')
```



From the whole variable correlation map, we can tell the correlations between the Balance and germany have corelation but not high, so we do not need to worry about multicollinearity.

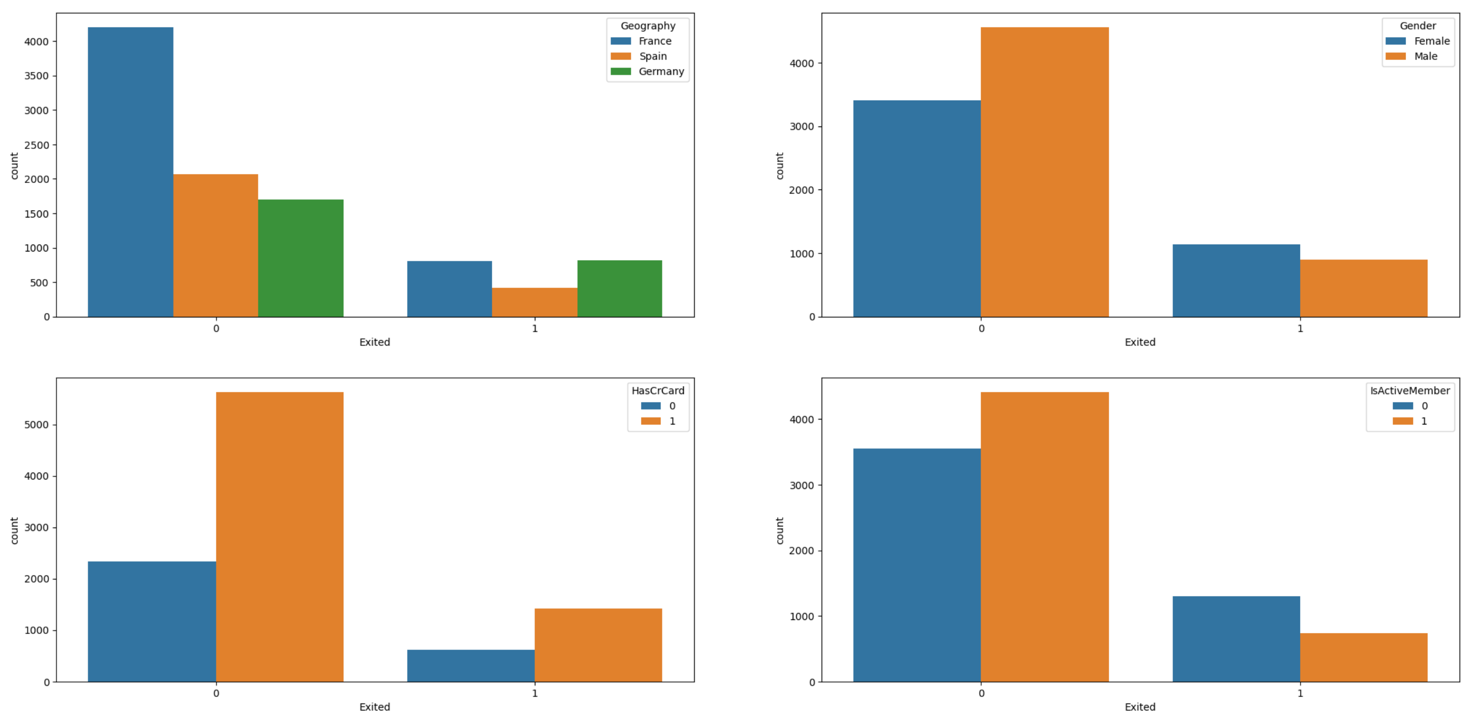
In [17]:
```
# Person correlation between variables
num_cols = ["CreditScore", "Age", "Tenure", "Balance", "NumOfProducts", "EstimatedSalary"]
corr = df[num_cols].corr()
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    square=True
)

ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right')
```

Out[17]:
```
[Text(0.5, 0, 'CreditScore'),
 Text(1.5, 0, 'Age'),
 Text(2.5, 0, 'Tenure'),
 Text(3.5, 0, 'Balance'),
 Text(4.5, 0, 'NumOfProducts'),
 Text(5.5, 0, 'EstimatedSalary')]
```



From the Pearson correlation coefficient map, we can tell the correlations between the numerical variables are not high, so we do not need to worry about multicollinearity

In [18]:
```
df_graphy = pd.read_csv('project data- bank1.csv')
_,ax1 = plt.subplots(2,2, figsize=[25,12])
cur_row = 0
cur_col = 0
for col in ['Geography', 'Gender', 'HasCrCard', 'IsActiveMember']:
    sns.countplot(x='Exited', hue=col, data=df_graphy, ax=ax1[cur_row][cur_col])
    cur_col += 1
    if cur_col == 2:
        cur_row += 1
        cur_col = 0
```



From Geography, we can see German are more likely to churn than Spanish and Franch, which is consistent to the positive correlation between Geography_Germany and Exited. From Gender, we can see more Female churn than Male, also consistent to the positive correlation between Gender_Female and Exited. HasCrCard has no obvious pattern. From IsActiveMember, we can see non-active customers are more likely to churn, whereas active customers are less likely to churn. In general, our observation is consistent with our correlation plot above.

Check outliers

In [19]:
```
df = pd.read_csv('project data- bank1.csv')
df.head(2)
```

Out[19]:

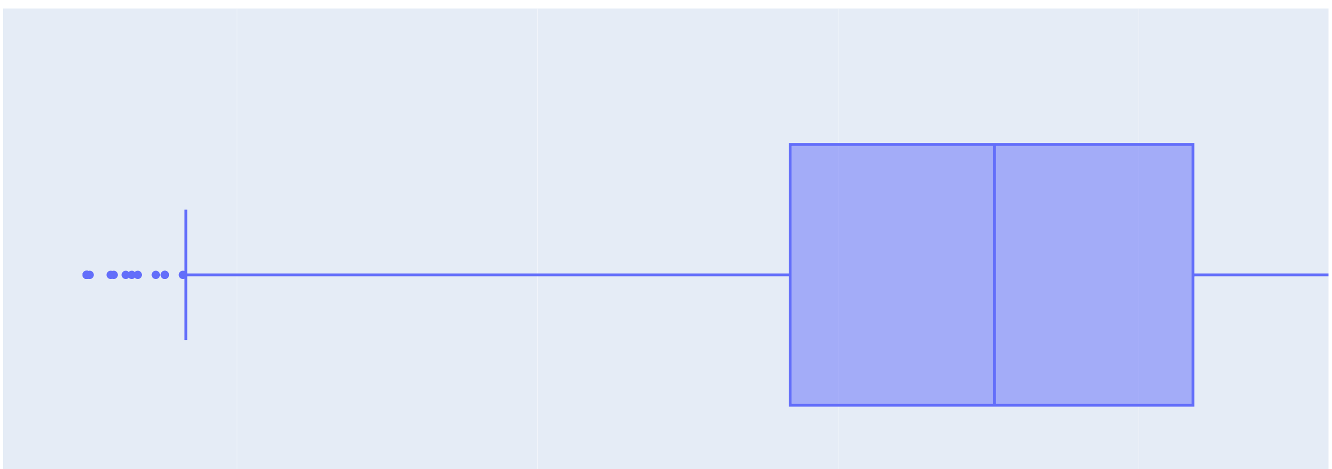
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42.0	2	0.00	1	1	1	101348.88	1
1	2	15647311	NaN	608	Spain	Female	NaN	1	83807.86	1	0	1	112542.58	0

In [20]:
```
from sklearn.ensemble import IsolationForest
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score
import numpy as np
```

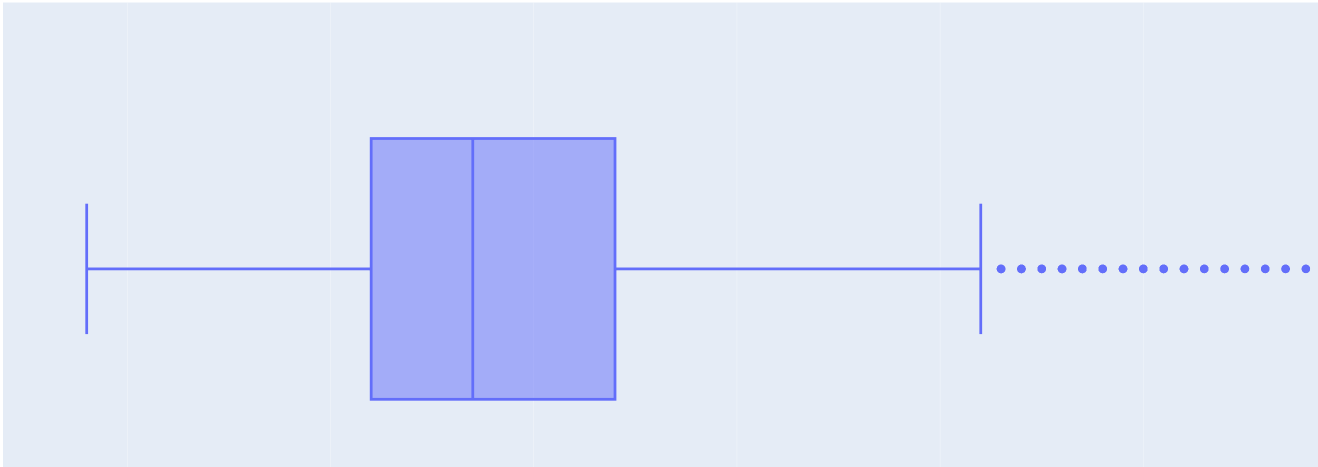
In [21]:
```
X = df[['Age','CreditScore', 'Balance', 'NumOfProducts', 'EstimatedSalary']]
y = df['Exited']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)
```

In [22]:
```
import matplotlib.pyplot as plt
import plotly.express as px
```

In [23]:
```
px.box(data_frame=df,x='CreditScore')
```

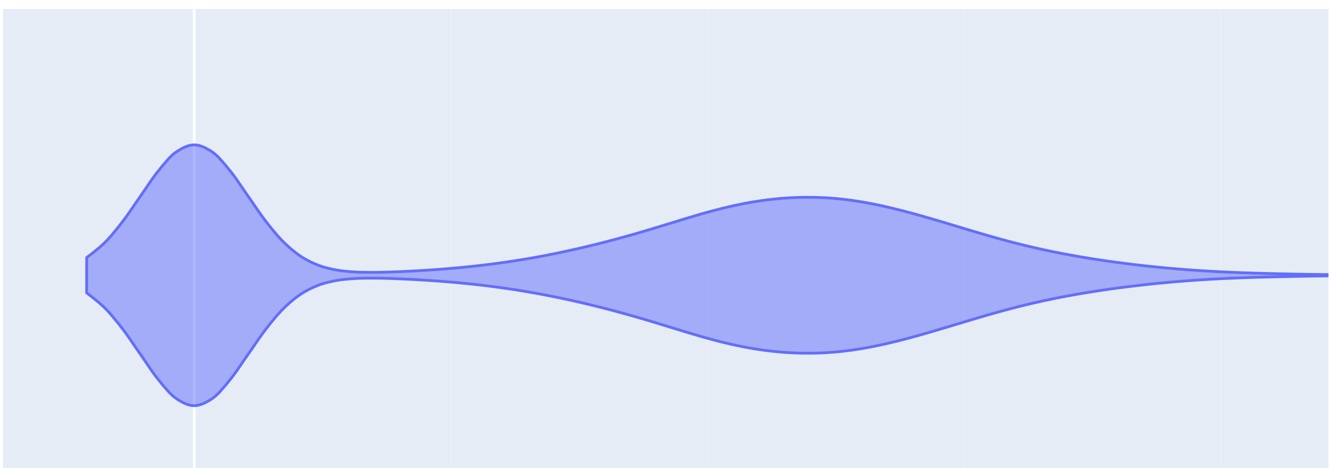


In [24]:
```
px.box(data_frame=df,x='Age')
```



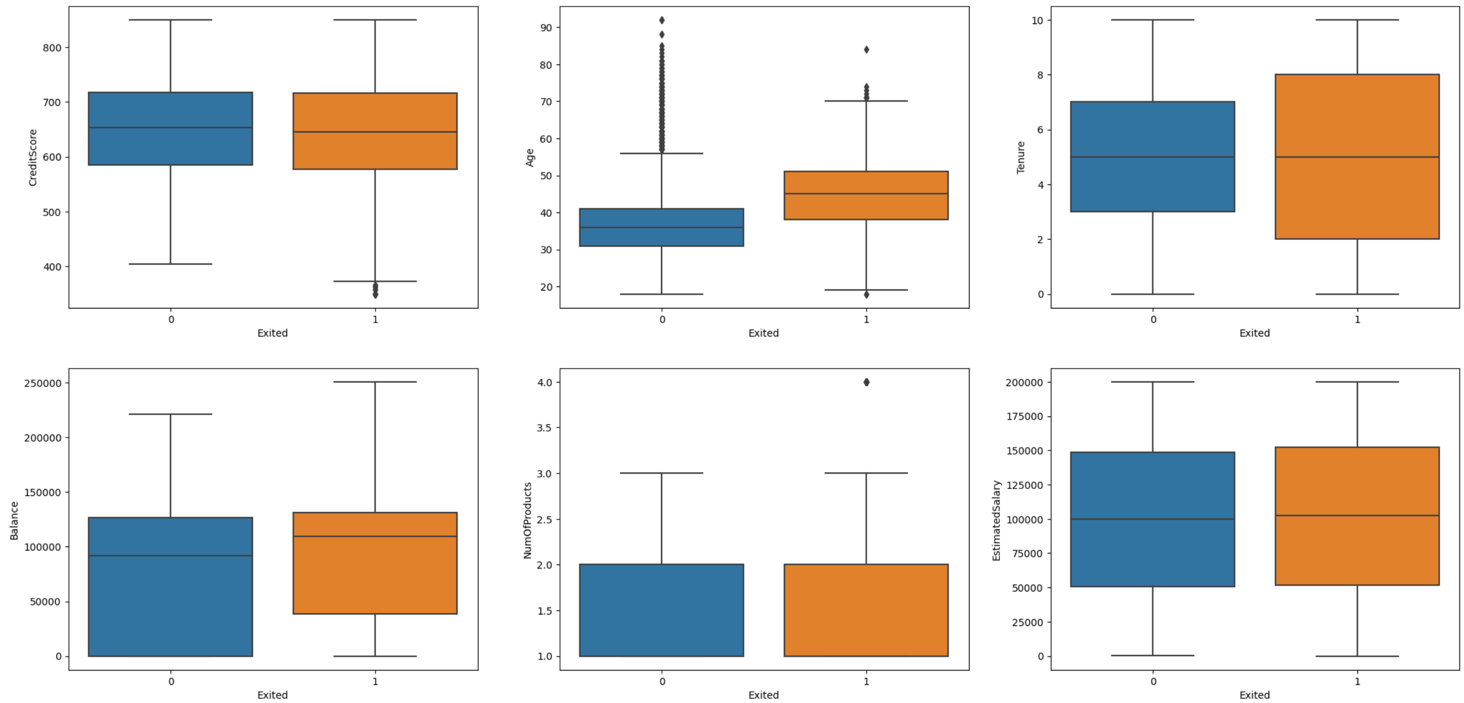
In [25]:

```
px.violin(data_frame=df,x='Balance') # after I load the graphy, it is not showing the pic
```



In [26]:

```
# Boxplots of numerical variables with respect to Exited

_,ax1 = plt.subplots(2,3, figsize=[25,12])
cur_row = 0
cur_col = 0
for col in num_cols:
    sns.boxplot(x='Exited', y=col, data=df, ax=ax1[cur_row][cur_col])
    cur_col += 1
    if cur_col == 3:
        cur_row += 1
        cur_col = 0
```



Using IsolationForest to Estimate Outliers

In [27]:
```
def IQR_outlier(df,x):
    q1  =  df[x].quantile(.25)
    q3  =  df[x].quantile(.75)
    iqr = q3-q1
    df['CreditScore']  = np.where(df[[x]]<(q1-1.5*iqr),-1,
                         np.where(df[[x]]>(q3+1.5*iqr),-1,1))
    return df
```

In [28]:
```
from sklearn.ensemble import IsolationForest
```

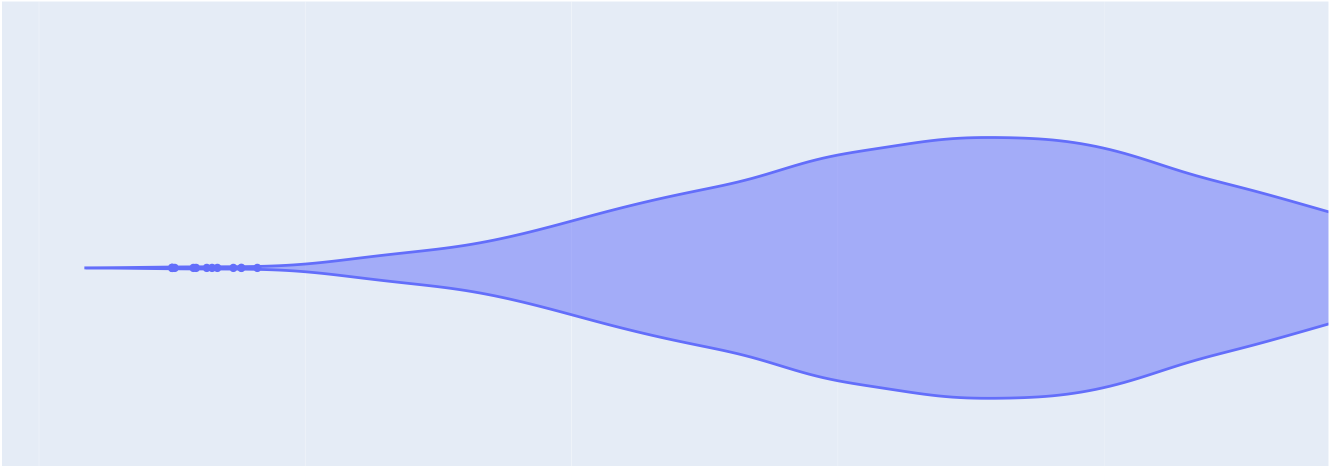
In [29]:
```
IsolationForest().fit(df[['CreditScore']]).predict(df[['CreditScore']])
```

Out[29]:
```
array([ 1,  1, -1, ...,  1,  1, -1])
```

fix the outliers - using Isolation Forest

In [30]:
```
def outliers_find(df,x):
    q1  =  df[x].quantile(.25)
    q3  =  df[x].quantile(.75)
    iqr = q3-q1
    df['Isolation Forest'] = IsolationForest().fit(df[[x]]).predict(df[[x]])
    return df
```

In [31]:
```
px.violin(data_frame=df,x='CreditScore')#
```



Add a new columns to dataset

Applying lambda function to find percentage of 'Balance' column

In [32]:
```
df3 = df.assign(Percentage_B = lambda x: (x['Balance'] /100000))
df3.select_dtypes(include=['int64','float64'])
```

Out[32]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Percentage_B
0	1	15634602	619	42.0	2	0.00	1	1	1	101348.88	1	0.000000
1	2	15647311	608	NaN	1	83807.86	1	0	1	112542.58	0	0.838079
2	3	15619304	502	42.0	8	159660.80	3	1	0	113931.57	1	1.596608
3	4	15701354	699	39.0	1	0.00	2	0	0	93826.63	0	0.000000
4	5	15737888	850	43.0	2	125510.82	1	1	1	79084.10	0	1.255108
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	771	39.0	5	0.00	2	1	0	96270.64	0	0.000000
9996	9997	15569892	516	35.0	10	57369.61	1	1	1	101699.77	0	0.573696
9997	9998	15584532	709	36.0	7	0.00	1	0	1	42085.58	1	0.000000
9998	9999	15682355	772	42.0	3	75075.31	2	1	0	92888.52	1	0.750753
9999	10000	15628319	792	28.0	4	130142.79	1	1	0	38190.78	0	1.301428

10000 rows × 12 columns

Adding new column of Percentage to the attributes is important for prediction because its varance is very high. otherwise it will infect the model accurancy.

dummy data

In [33]:
```
print(f'Geography unique',df.Geography.unique())
print(f'Gender unique', df.Gender.unique())
```

```
Geography unique ['France' 'Spain' 'Germany']
Gender unique ['Female' 'Male']
```

In [34]:
```
df4 = df_clean.convert_dtypes()
df4.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  string
 2   Gender           10000 non-null  string
 3   Age              10000 non-null  Float64
 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(3), Int64(6), string(2)
memory usage: 947.4 KB
```

In [35]:
```
all_methods=[]
for x in df4.Geography:
    all_methods.extend(x.split('|'))

method=pd.unique(all_methods)

zero_matrix=np.zeros((len(df4),len(method)))

dummies=pd.DataFrame(zero_matrix,columns=method)

meth=df4.Geography[1]
meth.split('|')

for i, meth in enumerate(df4.Geography):
    indices = dummies.columns.get_indexer(meth.split('|'))
    dummies.iloc[i, indices] = 1

df4_windic = df4.join(dummies.add_prefix('M_'))
df4_windic #.iloc[0:5]
```

Out[35]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	M_France	M_Spain	M_Germany
0	619	France	Female	42.0	2	0.0	1	1	1	101348.88	1	1.0	0.0	0.0
1	608	Spain	Female	38.921592	1	83807.86	1	0	1	112542.58	0	0.0	1.0	0.0
2	502	France	Female	42.0	8	159660.8	3	1	0	113931.57	1	1.0	0.0	0.0
3	699	France	Female	39.0	1	0.0	2	0	0	93826.63	0	1.0	0.0	0.0
4	850	Spain	Female	43.0	2	125510.82	1	1	1	79084.1	0	0.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	771	France	Male	39.0	5	0.0	2	1	0	96270.64	0	1.0	0.0	0.0
9996	516	France	Male	35.0	10	57369.61	1	1	1	101699.77	0	1.0	0.0	0.0
9997	709	France	Female	36.0	7	0.0	1	0	1	42085.58	1	1.0	0.0	0.0
9998	772	Germany	Male	42.0	3	75075.31	2	1	0	92888.52	1	0.0	0.0	1.0
9999	792	France	Female	28.0	4	130142.79	1	1	0	38190.78	0	1.0	0.0	0.0

10000 rows × 14 columns

In [36]:
```
all_methods=[]
for x in df4.Gender:
    all_methods.extend(x.split('|'))

method=pd.unique(all_methods)

zero_matrix=np.zeros((len(df4),len(method)))

dummies=pd.DataFrame(zero_matrix,columns=method)

meth=df4.Gender[1]
meth.split('|')

for i, meth in enumerate(df4.Gender):
    indices = dummies.columns.get_indexer(meth.split('|'))
    dummies.iloc[i, indices] = 1

df4_windic2 = df4_windic.join(dummies.add_prefix('G_'))
df4_windic2 #.iloc[0:5]
```

Out[36]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	M_France	M_Spain	M_Germany
0	619	France	Female	42.0	2	0.0	1	1	1	101348.88	1	1.0	0.0	0.0
1	608	Spain	Female	38.921592	1	83807.86	1	0	1	112542.58	0	0.0	1.0	0.0
2	502	France	Female	42.0	8	159660.8	3	1	0	113931.57	1	1.0	0.0	0.0
3	699	France	Female	39.0	1	0.0	2	0	0	93826.63	0	1.0	0.0	0.0
4	850	Spain	Female	43.0	2	125510.82	1	1	1	79084.1	0	0.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	771	France	Male	39.0	5	0.0	2	1	0	96270.64	0	1.0	0.0	0.0
9996	516	France	Male	35.0	10	57369.61	1	1	1	101699.77	0	1.0	0.0	0.0
9997	709	France	Female	36.0	7	0.0	1	0	1	42085.58	1	1.0	0.0	0.0
9998	772	Germany	Male	42.0	3	75075.31	2	1	0	92888.52	1	0.0	0.0	1.0
9999	792	France	Female	28.0	4	130142.79	1	1	0	38190.78	0	1.0	0.0	0.0

10000 rows × 16 columns

In [37]:
```
df4_windic2.drop(['Geography','Gender'],axis=1,inplace=True)
df4_windic2
```

Out[37]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	M_France	M_Spain	M_Germany	G_Female	G_Male
0	619	42.0	2	0.0	1	1	1	101348.88	1	1.0	0.0	0.0	1.0	0.0
1	608	38.921592	1	83807.86	1	0	1	112542.58	0	0.0	1.0	0.0	1.0	0.0
2	502	42.0	8	159660.8	3	1	0	113931.57	1	1.0	0.0	0.0	1.0	0.0
3	699	39.0	1	0.0	2	0	0	93826.63	0	1.0	0.0	0.0	1.0	0.0
4	850	43.0	2	125510.82	1	1	1	79084.1	0	0.0	1.0	0.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	771	39.0	5	0.0	2	1	0	96270.64	0	1.0	0.0	0.0	0.0	1.0
9996	516	35.0	10	57369.61	1	1	1	101699.77	0	1.0	0.0	0.0	0.0	1.0
9997	709	36.0	7	0.0	1	0	1	42085.58	1	1.0	0.0	0.0	1.0	0.0
9998	772	42.0	3	75075.31	2	1	0	92888.52	1	0.0	0.0	1.0	0.0	1.0
9999	792	28.0	4	130142.79	1	1	0	38190.78	0	1.0	0.0	0.0	1.0	0.0

10000 rows × 14 columns

Modeling

build a Logistic Regression model . the target is Exited is 0/1---> classification

In [49]:
```
from sklearn import model_selection

X = df4_windic2.drop('Exited',axis=1)
y= df['Exited']
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.25, stratify = y, random_state=516)
```

Standardization

In [50]:
```
# standardization
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train[:])
X_train[:] = scaler.transform(X_train[:])
X_test[:] = scaler.transform(X_test[:])
```

In [51]:
```
from sklearn.linear_model import LogisticRegression

logistic_model = LogisticRegression()

logistic_model.fit(X_train, y_train)
logistic_model.predict(X_test)
print("accuracy score is:")
logistic_model.score(X_test, y_test)
```

```
accuracy score is:
```

Out[51]: 0.8172

KNN

In [52]:
```
# 2. KNN
from sklearn.neighbors import KNeighborsClassifier

knn_model = KNeighborsClassifier()

knn_model.fit(X_train, y_train)
knn_model.predict(X_test)
print("accuracy score is:")
knn_model.score(X_test, y_test)

# accuracy score is:
# 0.8272
```

```
accuracy score is:
```

Out[52]: 0.8272

Random Forest

In [53]:
```
#3. Random Forest
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier()

rf_model.fit(X_train, y_train)
rf_model.predict(X_test)
print("accuracy score is:")
rf_model.score(X_test, y_test)
```

```
accuracy score is:
```

Out[53]: 0.866

Conclusion

From the Model prediction: Bank Customer who is not an active member, Credit cores, Estimated Salary are low, but Femail and elder and have high Balance will has higher risk of churn.

In [ ]:
```
-end
```