Customer Churn Prediction

In this project, we will predict whether a customer will leave the bank or not based on many factors

Following Factors are:

1. Credit score
2. Location of the Customer
3. Gender
4. Age
5. Tenure
6. Account Balance
7. Number of Bank Products Customer Uses
8. Has Credit Card
9. Is Active Member
10. Estimated Salary

In [1]:

```
from IPython.display import Image
Image(url='https://miro.medium.com/max/844/1*MyKDLRda6yHGR_8kgVvckg.png')
```

Out[1]:

In [2]:

```
# Importing the essential Libraries
import pandas as pd
import numpy as np
```

In [3]:

```
# Reading the Dataset
df = pd.read_csv('Churn_Modelling.csv')
```

In [4]:

```
df.head()
```

Out[4]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age
0	1	15634602	Hargrave	619	France	Female	42
1	2	15647311	Hill	608	Spain	Female	41
2	3	15619304	Onio	502	France	Female	42
3	4	15701354	Boni	699	France	Female	39
4	5	15737888	Mitchell	850	Spain	Female	43

In [5]:

```
df.shape
```

Out[5]:

```
(10000, 14)
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Ge
ography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts',
'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [7]:

```
df.dtypes
```

Out[7]:

```
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
Exited               int64
dtype: object
```

In [8]:

```
# Printing Unique Values of the categorical variables
print(df['Geography'].unique())
print(df['Gender'].unique())
print(df['NumOfProducts'].unique())
print(df['HasCrCard'].unique())
print(df['IsActiveMember'].unique())
```

```
['France' 'Spain' 'Germany']
['Female' 'Male']
[1 3 2 4]
[1 0]
[1 0]
```

In [9]:

```
# Checking if there are null values or not
df.isnull().sum()
```

Out[9]:

```
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
Exited             0
dtype: int64
```

In [10]:

```
df.describe()
```

Out[10]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	1
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	2

In [11]:

```
df.head()
```

Out[11]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age
0	1	15634602	Hargrave	619	France	Female	42
1	2	15647311	Hill	608	Spain	Female	41
2	3	15619304	Onio	502	France	Female	42
3	4	15701354	Boni	699	France	Female	39
4	5	15737888	Mitchell	850	Spain	Female	43

In [12]:

```
# Including only Potential Predictors as independent varibles
final_dataset = df[['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Ba
```

In [13]:

```
final_dataset.head()
```

Out[13]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts
0	619	France	Female	42	2	0.00	1
1	608	Spain	Female	41	1	83807.86	1
2	502	France	Female	42	8	159660.80	3
3	699	France	Female	39	1	0.00	2
4	850	Spain	Female	43	2	125510.82	1

In [14]:

```
# Converting the categorical variables into numerical and avoiding Dummy Varib
final_dataset = pd.get_dummies(final_dataset, drop_first=True)
```

In [15]:

```
final_dataset.head()
```

Out[15]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveM
0	619	42	2	0.00	1	1	
1	608	41	1	83807.86	1	0	
2	502	42	8	159660.80	3	1	
3	699	39	1	0.00	2	0	
4	850	43	2	125510.82	1	1	

In [16]:

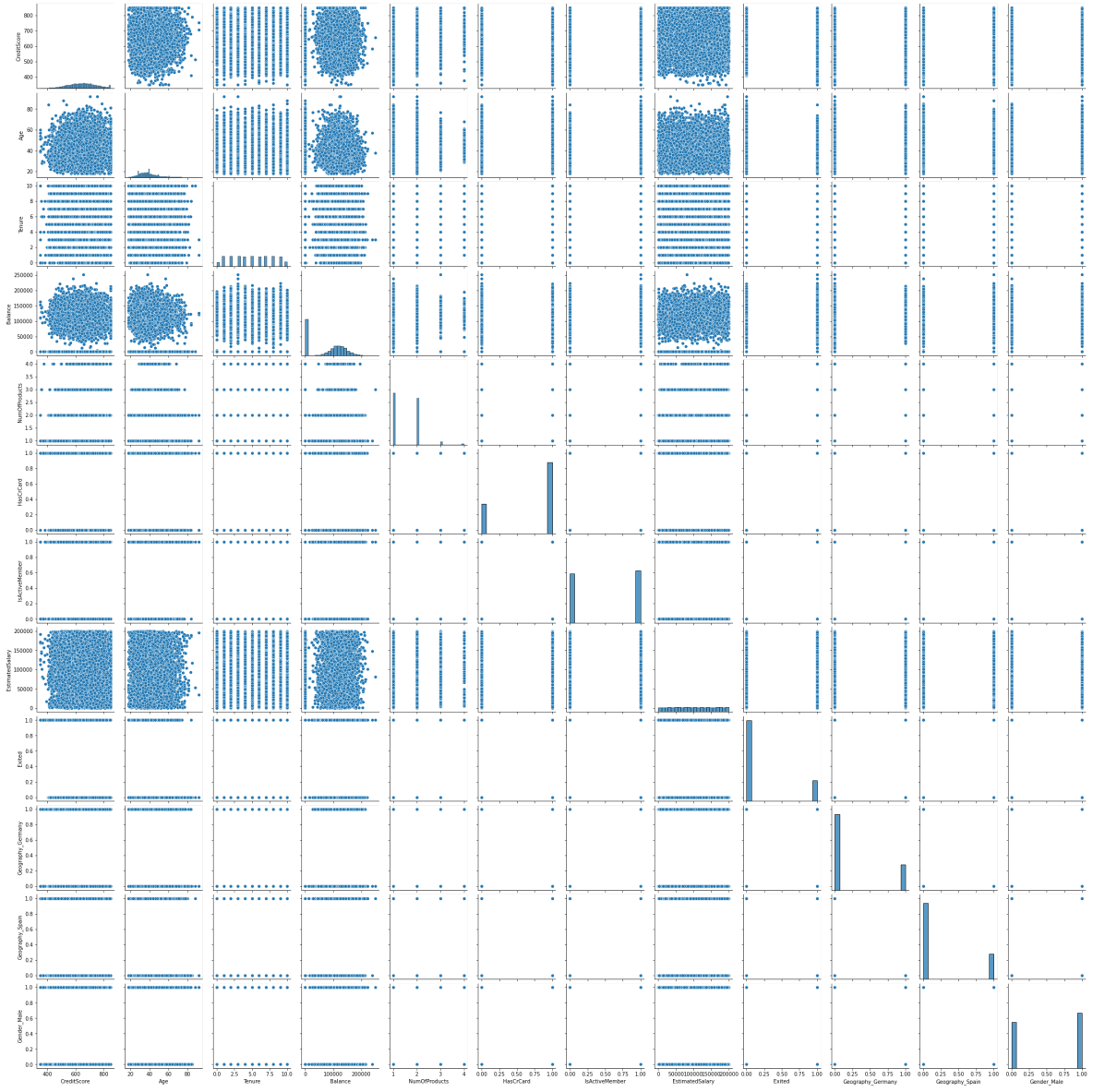
```
import seaborn as sns
```

In [17]:

```
sns.pairplot(final_dataset)
```

Out[17]:

```
<seaborn.axisgrid.PairGrid at 0x1e497be0>
```



In [18]:

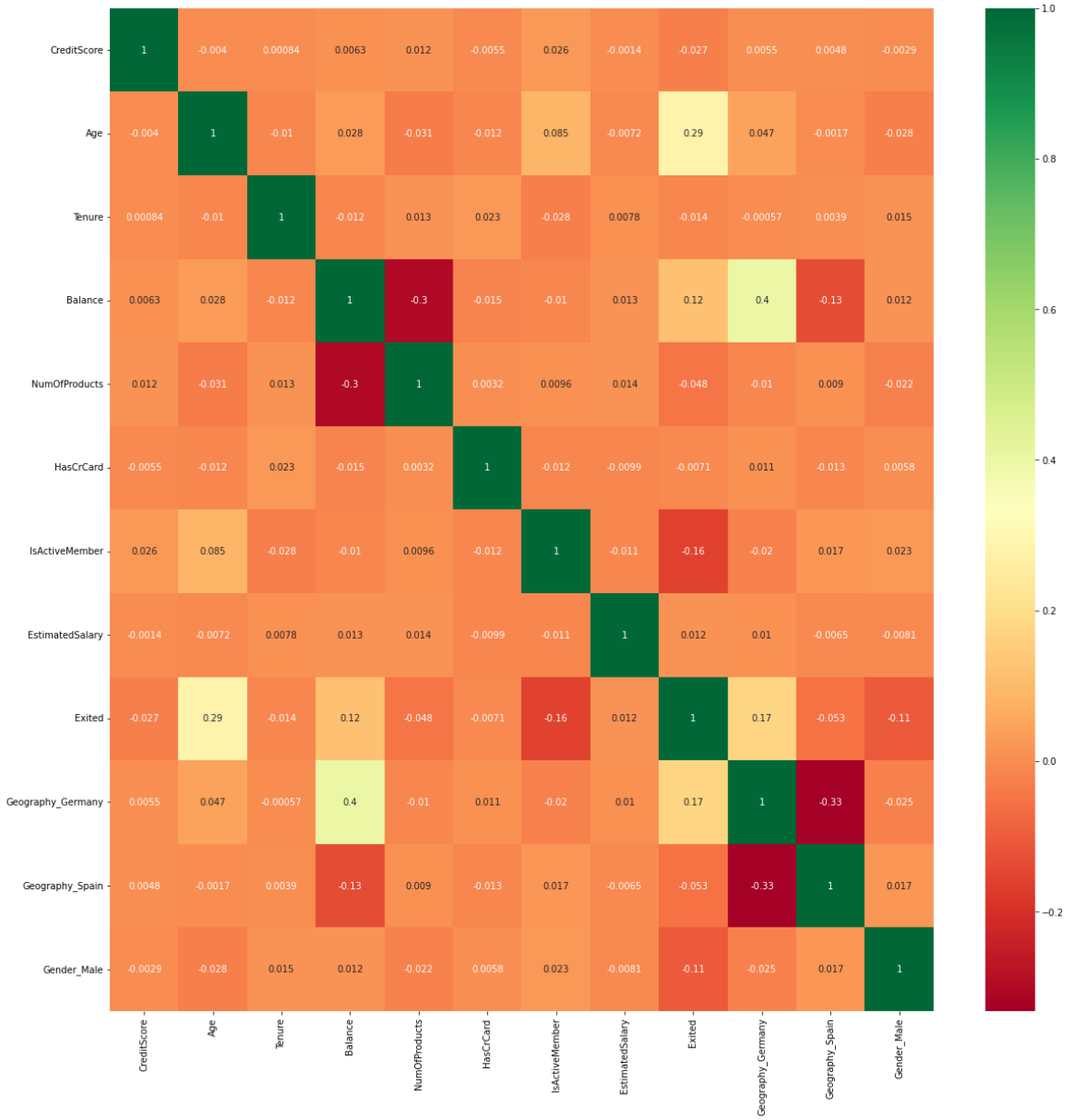
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [19]:

```
# Plotting The Correlations between all the features
corrmat = final_dataset.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
sns.heatmap(final_dataset[top_corr_features].corr(), annot=True, cmap='RdYlGn'
```

Out[19]:

<AxesSubplot:>



From the heatmap , we find that teh Age, Balance and the Geography of the Customer are Most important features

In [20]:

```
final_dataset.head()
```

Out[20]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveM
0	619	42	2	0.00	1	1	
1	608	41	1	83807.86	1	0	
2	502	42	8	159660.80	3	1	
3	699	39	1	0.00	2	0	
4	850	43	2	125510.82	1	1	

In [21]:

```
# Splitting the Dataset into Dependent and Independent Variables
X = final_dataset.iloc[:, [0,1,2,3,4,5,6,7,9,10,11]]
y = final_dataset.iloc[:, 8].values
```

In [22]:

```
X.head()
```

Out[22]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveM
0	619	42	2	0.00	1	1	
1	608	41	1	83807.86	1	0	
2	502	42	8	159660.80	3	1	
3	699	39	1	0.00	2	0	
4	850	43	2	125510.82	1	1	

In [23]:

```
y
```

Out[23]:

```
array([1, 0, 1, ..., 1, 1, 0], dtype=int64)
```

In [24]:

```
# Splitting the dataset into Training and Testing Data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_
```

In [25]:

```
# Standardizing the Dataset
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [26]:

```
print(X_train)
```

```
[[ 0.35649971 -0.6557859   0.34567966 ... -0.57946723 -0.576388
02
   0.91324755]
 [-0.20389777  0.29493847 -0.3483691  ...  1.72572313 -0.576388
02
   0.91324755]
 [-0.96147213 -1.41636539 -0.69539349 ... -0.57946723  1.734942
38
   0.91324755]
 ...
 [ 0.86500853 -0.08535128 -1.38944225 ... -0.57946723 -0.576388
02
  -1.09499335]
 [ 0.15932282  0.3900109   1.03972843 ... -0.57946723 -0.576388
02
   0.91324755]
 [ 0.47065475  1.15059039 -1.38944225 ...  1.72572313 -0.576388
02
   0.91324755]]
```

In [27]:

```
## Feature Importance
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor()
model.fit(X,y)
```

Out[27]:

```
ExtraTreesRegressor()
```

In [28]:

```
print(model.feature_importances_)
```

```
[0.11928469 0.23785024 0.10399456 0.13005991 0.1414219  0.03120
769
 0.04501133 0.11796912 0.02857235 0.02124091 0.0233873 ]
```

Random Forest Classifier

In [29]:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train,y_train)
```

Out[29]:

```
RandomForestClassifier()
```

In [30]:

```
y_pred = rf.predict(X_test)
```

In [31]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
cm = confusion_matrix(y_test,y_pred)
print(cm)
print(accuracy_score(y_test,y_pred))
```

```
[[1546   61]
 [ 207  186]]
0.866
```

In [32]:

```
# pickling the Model
import pickle
file = open('Customer_Churn_Prediction.pkl', 'wb')
pickle.dump(rf, file)
```