

Data Visualization and Pre-processing

Import libraries

InÂ [1]:

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

Load dataset

InÂ [2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

InÂ [3]:

```
data = pd.read_csv('drive/My Drive/Churn_Modelling.csv')
```

```
data.head()
```

Out[3]:

	Row Number	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	157013	Bonini	699	France	Female	39	1	0.00	2	0	0	93826.63	0

	Row Number	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
		54				ale								
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

InÂ [4]:

```
data.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                10000 non-null  object
3   CreditScore            10000 non-null  int64
4   Geography              10000 non-null  object
5   Gender                 10000 non-null  object
6   Age                   10000 non-null  int64
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(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Visualisations

1. Univariate Analysis

InÂ [5]:

```
data['Gender'].value_counts()
```

Out[5]:

```
Male      5457
Female    4543
```

Name: Gender, dtype: int64

InÂ [7]:

```
# Plotting the features of the dataset to see the correlation  
between them
```

```
plt.hist(x = data.Gender, bins = 3, color = 'pink')  
plt.title('comparison of male and female')  
plt.xlabel('Gender')  
plt.ylabel('population')  
plt.show()
```

InÂ [6]:

```
data['Age'].value_counts()
```

Out[6]:

```
37      478  
38      477  
35      474  
36      456  
34      447  
...  
92         2  
82         1  
88         1  
85         1  
83         1
```

Name: Age, Length: 70, dtype: int64

InÂ [8]:

```
# comparison of age in the dataset
```

```
plt.hist(x = data.Age, bins = 10, color = 'orange')  
plt.title('comparison of Age')  
plt.xlabel('Age')  
plt.ylabel('population')  
plt.show()
```

InÂ [9]:

```
data['Geography'].value_counts()
```

Out[9]:

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

InÂ [10]:

```
# comparison of geography
```

```
plt.hist(x = data.Geography, bins = 5, color = 'green')
plt.title('comparison of Geography')
plt.xlabel('Geography')
plt.ylabel('population')
plt.show()
```

InÂ [11]:

```
data['HasCrCard'].value_counts()
```

Out[11]:

```
1    7055
0    2945
Name: HasCrCard, dtype: int64
```

InÂ [12]:

```
# comparision of how many customers hold the credit card

plt.hist(x = data.HasCrCard, bins = 3, color = 'red')
plt.title('how many people have or not have the credit card')
plt.xlabel('customers holding credit card')
plt.ylabel('population')
plt.show()
```

InÂ [13]:

```
data['IsActiveMember'].value_counts()
```

Out[13]:

```
1    5151
0    4849
Name: IsActiveMember, dtype: int64
```

InÂ [14]:

```
# How many active member does the bank have ?

plt.hist(x = data.IsActiveMember, bins = 3, color = 'brown')
plt.title('Active Members')
plt.xlabel('Customers')
plt.ylabel('population')
plt.show()
```

2. Bi - Variate Analysis

InÂ [15]:

```
# comparison between Geography and Gender

Gender = pd.crosstab(data['Gender'],data['Geography'])
```

```
Gender.div(Gender.sum(1).astype(float), axis=0).plot(kind="bar",
stacked=True, figsize=(6, 6))
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6a93dbbfd0>
```

In [16]:

```
# comparison between geography and card holders
```

```
HasCrCard = pd.crosstab(data['HasCrCard'], data['Geography'])
HasCrCard.div(HasCrCard.sum(1).astype(float), axis =
0).plot(kind = 'bar',
stacked =
True,figsize = (6, 6))
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6a93ced590>
```

In [17]:

```
# comparison of active member in differnt geographies
```

```
IsActiveMember = pd.crosstab(data['IsActiveMember'],
data['Geography'])
IsActiveMember.div(IsActiveMember.sum(1).astype(float), axis =
0).plot(kind = 'bar',
stacked = True,
figsize= (6, 6))
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6a93c7c950>
```

In [18]:

```
# comparing ages in different geographies
```

```
Age = pd.crosstab(data['Age'], data['Geography'])
Age.div(Age.sum(1).astype(float), axis = 0).plot(kind = 'bar',
stacked = True,
figsize = (15,15))
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6a93bfea10>
```

In [19]:

```
# calculating total balance in france, germany and spain
```

```
total_france = data.Balance[data.Geography == 'France'].sum()
total_germany = data.Balance[data.Geography == 'Germany'].sum()
total_spain = data.Balance[data.Geography == 'Spain'].sum()
```

```
print("Total Balance in France :",total_france)
print("Total Balance in Germany :",total_germany)
print("Total Balance in Spain :",total_spain)

Total Balance in France : 311332479.49
Total Balance in Germany : 300402861.38
Total Balance in Spain : 153123552.01
```

InÂ [20]:

```
# plotting a pie chart

labels = 'France', 'Germany', 'Spain'
colors = ['cyan', 'magenta', 'orange']
sizes = [311, 300, 153]
explode = [ 0.01, 0.01, 0.01]

plt.pie(sizes, colors = colors, labels = labels, explode =
explode, shadow = True)

plt.axis('equal')
plt.show()
```

3. Multi - Variate Analysis

InÂ [21]:

```
sns.pairplot(data=data, hue='Exited')
```

Out[21]:

```
<seaborn.axisgrid.PairGrid at 0x7f6a93ddd510>
```

Descriptive statistics

InÂ [23]:

```
#Statistical analysis
data.describe()
```

Out[23]:

	Row Num ber	Cust omer Id	Cred itSco re	Age	Tenu re	Balan ce	Num OfPro ducts	Has CrC ard	IsActi veMe mber	Estim atedS alary	Exite d
co	1000	1.000	1000	1000	1000	10000	10000.	1000	10000.	10000.	1000
u	0.00	000e	0.000	0.000	0.000	.0000	00000	0.00	00000	00000	0.000
nt	000	+04	000	000	000	00	0	000	0	0	000

	Row Num ber	Cust omer Id	Cred itSco re	Age	Tenu re	Balan ce	Num OfPro ducts	Has CrC ard	IsActi veMe mber	Estim atedS alary	Exite d
m ea n	5000 .500 00	1.569 094e +07	650.5 2880 0	38.92 1800	5.012 800	76485 .8892 88	1.5302 00	0.70 550	0.5151 00	10009 0.2398 81	0.203 700
st d	2886 .895 68	7.193 619e +04	96.65 3299	10.48 7806	2.892 174	62397 .4052 02	0.5816 54	0.45 584	0.4997 97	57510. 49281 8	0.402 769
m in	1.00 000	1.556 570e +07	350.0 0000 0	18.00 0000	0.000 000	0.000 000	1.0000 00	0.00 000	0.0000 00	11.580 000	0.000 000
2 5 %	2500 .750 00	1.562 853e +07	584.0 0000 0	32.00 0000	3.000 000	0.000 000	1.0000 00	0.00 000	0.0000 00	51002. 11000 0	0.000 000
5 0 %	5000 .500 00	1.569 074e +07	652.0 0000 0	37.00 0000	5.000 000	97198 .5400 00	1.0000 00	1.00 000	1.0000 00	10019 3.9150 00	0.000 000
7 5 %	7500 .250 00	1.575 323e +07	718.0 0000 0	44.00 0000	7.000 000	12764 4.240 000	2.0000 00	1.00 000	1.0000 00	14938 8.2475 00	0.000 000
m a x	1000 0.00 000	1.581 569e +07	850.0 0000 0	92.00 0000	10.00 0000	25089 8.090 000	4.0000 00	1.00 000	1.0000 00	19999 2.4800 00	1.000 000

Handle the Missing values

```
#Missing Values
data.isnull().sum()
```

InÂ [24]:

Out[24]:

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
No missing values are found.
```

Find the outliers and replace the outliers

```
sns.boxplot(data = data, x = 'CreditScore')
```

InÂ [25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6a8ecfe7d0>
```

Out[25]:

```
sns.boxplot(data = data, x = 'Age')
```

InÂ [26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6a8e9a3f50>
```

Out[26]:

```
sns.boxplot(data = data, x = 'Balance')
```

InÂ [27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6a8d0e0e90>
```

Out[27]:

```
sns.boxplot(data = data, x = 'EstimatedSalary')
```

InÂ [28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6a8d0e0c50>
```

Out[28]:

```
for i in data:
    if data[i].dtype=='int64' or data[i].dtypes=='float64':
        q1=data[i].quantile(0.25)
        q3=data[i].quantile(0.75)
```

InÂ [29]:


```

iqr=q3-q1
upper=q3+1.5*iqr
lower=q1-1.5*iqr
data[i]=np.where(data[i] >upper, upper, data[i])
data[i]=np.where(data[i] <lower, lower, data[i])

```

InÂ [30]:

```
data.describe()
```

Out[30]:

	Row Num ber	Cust omer Id	Cred itSco re	Age	Tenu re	Balan ce	NumO fProd ucts	Has CrC ard	IsActi veMe mber	Estim atedSa lary	Ex ite d
co u nt	1000 0.000 00	1.000 000e +04	1000 0.000 000	1000 0.000 000	1000 0.000 000	10000 .0000 00	10000. 00000 0	1000 0.00 000	10000. 00000 0	10000. 00000 0	10 00 0.0
m ea n	5000. 5000 0	1.569 094e +07	650.5 6130 0	38.66 0800	5.012 800	76485 .8892 88	1.5272 00	0.70 550	0.5151 00	10009 0.2398 81	0.0
st d	2886. 8956 8	7.193 619e +04	96.55 8702	9.746 704	2.892 174	62397 .4052 02	0.5700 81	0.45 584	0.4997 97	57510. 49281 8	0.0
m in	1.000 00	1.556 570e +07	383.0 0000 0	18.00 0000	0.000 000	0.000 000	1.0000 00	0.00 000	0.0000 00	11.580 000	0.0
2 5 %	2500. 7500 0	1.562 853e +07	584.0 0000 0	32.00 0000	3.000 000	0.000 000	1.0000 00	0.00 000	0.0000 00	51002. 11000 0	0.0
5 0 %	5000. 5000 0	1.569 074e +07	652.0 0000 0	37.00 0000	5.000 000	97198 .5400 00	1.0000 00	1.00 000	1.0000 00	10019 3.9150 00	0.0
7 5 %	7500. 2500 0	1.575 323e +07	718.0 0000 0	44.00 0000	7.000 000	12764 4.240 000	2.0000 00	1.00 000	1.0000 00	14938 8.2475 00	0.0

	Row Num ber	Cust omer Id	Cred itSco re	Age	Tenu re	Balan ce	NumO fProd ucts	Has CrC ard	IsActi veMe mber	Estim atedSa lary	Ex ite d
m	1000	1.581	850.0			25089				19999	
a	0.000	569e	0000	62.00	10.00	8.090	3.5000	1.00	1.0000	2.4800	0.0
x	00	+07	0	0000	0000	000	00	000	00	00	

Preprocessing

InÂ [31]:

```
# Removing the unnecassary features from the dataset

data = data.drop(['CustomerId', 'Surname', 'RowNumber'], axis =
1)

print(data.columns)
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure',
'Balance',
      'NumOfProducts', 'HasCrCard', 'IsActiveMember',
'EstimatedSalary',
      'Exited'],
      dtype='object')
```

InÂ [32]:

```
data.shape
```

Out[32]:

```
(10000, 11)
```

Split the data into dependent and independent variables

InÂ [33]:

```
# splitting the dataset into x(independent variables) and
y(dependent variables)

x = data.iloc[:,0:10]
y = data.iloc[:,10]

print(x.shape)
print(y.shape)

print(x.columns)
(10000, 10)
(10000,)
```

```
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure',
      'Balance',
      'NumOfProducts', 'HasCrCard', 'IsActiveMember',
      'EstimatedSalary'],
      dtype='object')
```

Check for Categorical columns and perform encoding

InÂ [34]:

```
# Encoding Categorical variables into numerical variables
# One Hot Encoding
```

```
x = pd.get_dummies(x)
```

```
x.head()
```

Out[34]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Geography_France	Geography_Germany	Geography_Spain	Gender_Female	Gender_Male
0	619.0	42.0	2.0	0.00	1.0	1.0	1.0	101348.88	1	0	0	1	0
1	608.0	41.0	1.0	83807.86	1.0	0.0	1.0	112542.58	0	0	1	1	0
2	502.0	42.0	8.0	159660.80	3.0	1.0	0.0	113931.57	1	0	0	1	0
3	699.0	39.0	1.0	0.00	2.0	0.0	0.0	93826.63	1	0	0	1	0

	Credit Score	Age	Tenure	Balance	NumOfProducts	HassCreditCard	IsActiveMember	EstimatedSalary	Geography_France	Geography_Germany	Geography_Spain	Gender_Female	Gender_Male
4	850	43	2.0	125510.82	1.0	1.0	1.0	79084.10	0	0	1	1	0

Split the data into training and testing

InÂ [35]:

```
# splitting the data into training and testing set

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size = 0.25, random_state = 0)

print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)

(7500, 13)
(7500,)
(2500, 13)
(2500,)
```

Scale the independent variables

InÂ [36]:

```
# Feature Scaling
# Only on Independent Variable to convert them into values
ranging from -1 to +1

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)

x_train = pd.DataFrame(x_train)
x_train.head()
```

												Out[36]:	
	0	1	2	3	4	5	6	7	8	9	10	11	12
0	- 0.73 682 8	- 0.04 228 3	- 0.00 886 0	- 0.67 316 0	- 2.58 323 1	- 1.55 362 4	- 1.03 446 0	- 1.64 081 0	- 1.01 558 8	- 1.76 021 6	- 0.57 468 2	- 1.08 726 1	- 1.08 726 1
1	- 1.02 525 7	- 0.67 449 6	- 0.00 886 0	- 1.20 772 4	- 0.82 257 8	- 0.64 365 7	- 1.03 446 0	- 0.07 927 2	- 0.98 465 1	- 0.56 811 2	- 0.57 468 2	- 1.08 726 1	- 1.08 726 1
2	- 0.80 886 1	- 0.46 970 2	- 1.39 329 3	- 0.35 693 7	- 0.82 257 8	- 0.64 365 7	- 0.96 668 8	- 0.99 684 0	- 1.01 558 8	- 0.56 811 2	- 1.74 009 4	- 1.08 726 1	- 1.08 726 1
3	- 0.39 667 7	- 0.06 011 4	- 0.00 886 0	- 0.00 935 6	- 0.93 807 6	- 0.64 365 7	- 0.96 668 8	- 1.59 174 6	- 1.01 558 8	- 0.56 811 2	- 1.74 009 4	- 0.91 974 3	- 0.91 974 3
4	- 0.46 890 8	- 1.37 344 4	- 0.70 107 7	- 1.20 772 4	- 0.82 257 8	- 0.64 365 7	- 0.96 668 8	- 1.28 330 2	- 0.98 465 1	- 0.56 811 2	- 0.57 468 2	- 0.91 974 3	- 0.91 974 3