

Assignment -2

Data Visualization and Pre-processing in ipynb

Assignment Date	21 September 2022
Student Name	Komala.P
Team ID	IBM-29269-1662616381
Maximum Marks	2 Marks

1.Download the dataset

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

2.Load the dataset
```

```
df=pd.read_csv('/content/Churn_Modelling.csv')
df.head()
```

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

```
   Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  \  0  2
0.00    1         1             1           1
1      1   83807.86             1           0             1
2      8  159660.80             3           1             0
3      1         0.00             2           0             0
4      2  125510.82             1           1             1
```

```
   EstimatedSalary  Exited  0
101348.88         1
1      112542.58         0
2      113931.57         1
3       93826.63         0
4       79084.10         0
```

```
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	10000	non-null	object
3	CreditScore	10000	non-null	int64

- a. Geography 10000 non-null object
- b. Gender 10000 non-null object
- c. Age 10000 non-null int64
- d. Tenure 10000 non-null int64
- e. Balance 10000 non-null float64
- f. NumOfProducts 10000 non-null int64
- g. HasCrCard 10000 non-null int64
- h. IsActiveMember 10000 non-null int64
- i. EstimatedSalary 10000 non-null float64
- j. Exited 10000 non-null int64 dtypes: float64(2), int64(9), object(3) memory usage: 1.1+ MB

1. Perform Below Visualisations Univariate

```
Analysis df['Geography'].value_counts()
```

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

```
# comparison of geography
```

```
plt.hist(x = df.Geography, bins = 6, color = 'lightblue')
plt.title('comparison of Geography') plt.xlabel('Geography')
plt.ylabel('population') plt.show()
```

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

1    5151
0    4849
Name: IsActiveMember, dtype: int64

# How many active member does the bank have ?

plt.hist(x = df.IsActiveMember, bins = 5, color = 'pink')
plt.title('Active Members')
plt.xlabel('Customers')
plt.ylabel('population') plt.show()
```

```
df['Gender'].value_counts() Male
5457
Female 4543
Name: Gender, dtype: int64
```

```
# Plotting the features of the dataset to see the correlation between them
plt.hist(x = df.Gender, bins = 4, color = 'lightgreen') plt.title('comparison of
male and female')
plt.xlabel('Gender')
plt.ylabel('population') plt.show()
```

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

```
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

```

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

```

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

```

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

```

1    7055
0    2945
Name: HasCrCard, dtype: int64

```

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

```

plt.hist(x = df.HasCrCard, bins = 5, color = 'blue') plt.title('how
many people have or not have the credit card') plt.xlabel('customers
holding credit card') plt.ylabel('population')
plt.show()

```

Bi - Variate Analysis

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

```

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

```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1a78a13d0>
```

```
# comparison between Geography and Gender
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fala6c48bd0>
```

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

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fala6c36810>
```

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

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

```
# plotting a pie chart
```

```
labels = 'France', 'Germany', 'Spain' colors =  
['green', 'yellow', '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()
```

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

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1a6b0c0d0>
```

Multi - Variate Analysis

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

```
<seaborn.axisgrid.PairGrid at 0x7fa1a1860550>
```

2. Perform descriptive statistics on the dataset

```
df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure
\					
count	10000.000000	1.000000e+04	10000.000000	10000.000000	10000.000000
mean	5000.500000	1.569094e+07	650.528800	38.921800	5.012800
std	2886.895688	7.193619e+04	96.653299	10.487806	2.892174
min	1.000000	1.556570e+07	350.000000	18.000000	0.000000
25%	2500.750000	1.562853e+07	584.000000	32.000000	3.000000
50%	5000.500000	1.569074e+07	652.000000	37.000000	5.000000
75%	7500.250000	1.575323e+07	718.000000	44.000000	7.000000
max	10000.000000	1.581569e+07	850.000000	92.000000	10.000000

	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
count	10000.000000	10000.000000	10000.000000	10000.000000	
mean	76485.889288	1.530200	0.705500	0.515100	
std	62397.405202	0.581654	0.455840	0.499797	
min	0.000000	1.000000	0.000000	0.000000	
25%	0.000000	1.000000	0.000000	0.000000	

50%	97198.540000	1.000000	1.000000	1.000000
75%	127644.240000	2.000000	1.000000	1.000000
max	250898.090000	4.000000	1.000000	1.000000

	EstimatedSalary	Exited
count	10000.000000	10000.000000
mean	100090.239881	0.203700
std	57510.492818	0.402769
min	11.580000	0.000000
25%	51002.110000	0.000000
50%	100193.915000	0.000000
75%	149388.247500	0.000000
max	199992.480000	1.000000

3. Handle the Missing values

```
df.isnull().sum() 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
```

4. Find the outliers and replace the outliers

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa19f13e510>
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa19f0c2410>
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa19f03d1d0>
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa19d74fb10>
```

```
sns.boxplot(data = df, x = 'RowNumber')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa19d7c2b90>
```

```
sns.boxplot(data = df, x = 'Tenure')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa19be57c90>
```

5. Check for Categorical columns and perform encoding

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

	RowNumber	CustomerId	CreditScore	Age	Tenure	Surname_Abazu	\
0	1.0	15634602.0	619.0	42.0	2.0	0	
1	2.0	15647311.0	608.0	41.0	1.0	0	
2	3.0	15619304.0	502.0	42.0	8.0	0	
3	4.0	15701354.0	699.0	39.0	1.0	0	
4	5.0	15737888.0	850.0	43.0	2.0	0	
	Surname_Abbie	Surname_Abbott	Surname_Abdullah	Surname_Abdulov	...		
\							
0	0	0	0	0	0	...	

1	0	0	0	0	...
2	0	0	0	0	...
3	0	0	0	0	...
4	0	0	0	0	...

	Surname_Zubarev	Surname_Zubareva	Surname_Zuev	Surname_Zuyev	\
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	0	0	0	0	

	Surname_Zuyeva	Geography_France	Geography_Germany	Geography_Spain	\
0	0	1	0	0	
1	0	0	0	1	
2	0	1	0	0	
3	0	1	0	0	
4	0	0	0	1	

	Gender_Female	Gender_Male
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0

[5 rows x 2942 columns]

6. Split the data into dependent and independent variables

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

```
x = df.iloc[:,0:8]
y = df.iloc[:,8]
```

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

```
print(x.columns)
```

```
(10000, 8)
(10000,)
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
      'Gender', 'Age', 'Tenure'],
      dtype='object')
```

7. Scale the independent variables

```
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()
```

	0	1	2	3	4	5	6		
\									
0	-0.702176	-	-0.736828	0.042283	0.008860	-0.016332	0.0		
		1.343330							
	-0.0231								
1	-1.485722	1.5583	1.025257	-0.674496	0.008860	-0.016332	0.0		
		30							
	-0.0231								
2	-0.524522	-	0.808861	-0.469702	1.393293	-0.016332	0.0		
		0.655156							
	-0.0231								
3	-1.167396	1.2005	0.396677	-0.060114	0.008860	-0.016332	0.0		
		94							
	-0.0231								
4	-1.451159	0.7787	-0.468908	1.373444	0.701077	-0.016332	0.0		
		98							
	-0.0231								
8									
9									
...									
2932									
2933									
2934									
2935									
2936									
2									
\									
0	0.0	0.0	...	-0.011548	0.0	-	-0.011548	-0.016332	-
						0.011548			1.015
1	0.0	0.0	...	-0.011548	0.0	-	-0.011548	-0.016332	0.98
						0.011548			

2	0.0	0.0	...	-0.011548	0.0	-	-0.011548	-0.016332	-
						0.011548			1.015
3	0.0	0.0	...	-0.011548	0.0	-	-0.011548	-0.016332	-
						0.011548			1.015
4	0.0	0.0	...	-0.011548	0.0	-	-0.011548	-0.016332	0.98
						0.011548			

	2938	2939	2940	2941
	0	-	1.087261	-1.087261
1.760216	0.574682			
1 -	-	1.087261	-1.087261	
0.568112	0.574682			
2 -	1.740094	1.087261	-1.087261	
0.568112				
3 -	1.740094	-	0.919743	
0.568112		0.919743		
4 -	-	-	0.919743	
0.568112	0.574682	0.919743		

[5 rows x 2942 columns]

8. Split the data into training and testing

```
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, 2942)
(7500,)
(2500, 2942)
(2500,)
```

