

**Assignment -2**  
Python Programming

Assignment Date	23 september 2022
Student Name	Shiny .R
Student Roll Number	311419205036
Maximum Marks	2 Marks

# Data Visualization and Pre-processing

## Question-1:

### 1. Load the dataset

#### Solution:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
sns.set_style('darkgrid')
sns.set(font_scale=1.3)
```

In [2]:

```
df=pd.read_excel("/content/Churn_Modelling.xlsx")
```

```
In [1]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
sns.set_style('darkgrid')
sns.set(font_scale=1.3)
```

```
In [2]: df=pd.read_excel("/content/Churn_Modelling.xlsx")
```

## Question-2:

### 2. Perform Below Visualizations.

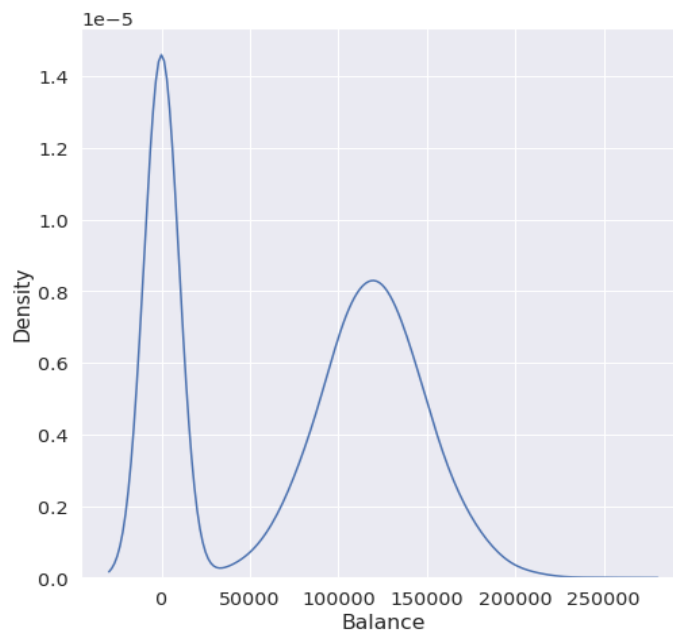
- Univariate Analysis
- Bi - Variate Analysis
- Multi - Variate Analysis

#### Solution:

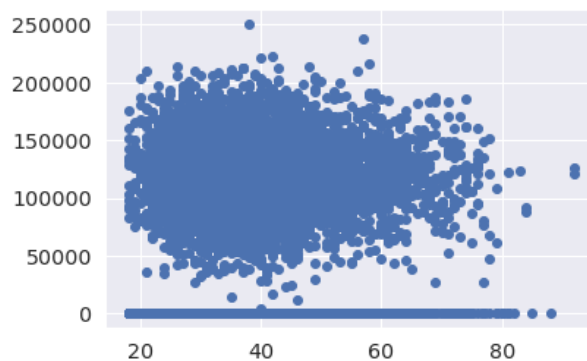
```
.
#Perform Univariate Analysis
plt.figure(figsize=(8,8))
sns.kdeplot(x=df['Balance'])
```

```
In [7]: #Perform Univariate Analysis
plt.figure(figsize=(8,8))
sns.kdeplot(x=df['Balance'])
```

Out[7]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7fc3f3579c50>



```
#Perform Bivariate Analysis
plt.scatter(df.Age,df.Balance)
```



```
#Perform Bivariate Analysis
df.corr()
```

Out[9]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
CreditScore	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
Age	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
Tenure	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
Balance	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
NumOfProducts	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
HasCrCard	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
IsActiveMember	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
EstimatedSalary	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097
Exited	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.000000

```
#Perform Bivariate Analysis
```

```
import statsmodels.api as sm

#define response variable
y = df['CreditScore']

#define explanatory variable
x = df[['EstimatedSalary']]

#add constant to predictor variables
x = sm.add_constant(x)

#fit linear regression model
model = sm.OLS(y, x).fit()

#view model summary
print(model.summary())
```

```
=====
                        OLS Regression Results
=====
```

Dep. Variable:	CreditScore	R-squared:	0.000
Model:	OLS	Adj. R-squared:	-0.000
Method:	Least Squares	F-statistic:	0.01916
Date:	Thu, 29 Sep 2022	Prob (F-statistic):	0.890
Time:	14:58:55	Log-Likelihood:	-59900.
No. Observations:	10000	AIC:	1.198e+05
Df Residuals:	9998	BIC:	1.198e+05
Df Model:	1		
Covariance Type:	nonrobust		

```
=====
```

	coef	std err	t	P> t	[0.025	0.975]
const	650.7617	1.940	335.407	0.000	646.958	654.565
EstimatedSalary	-2.326e-06	1.68e-05	-0.138	0.890	-3.53e-05	3.06e-05

```
=====
```

Omnibus:	132.939	Durbin-Watson:	2.014
Prob(Omnibus):	0.000	Jarque-Bera (JB):	84.242
Skew:	-0.072	Prob(JB):	5.10e-19
Kurtosis:	2.574	Cond. No.	2.32e+05

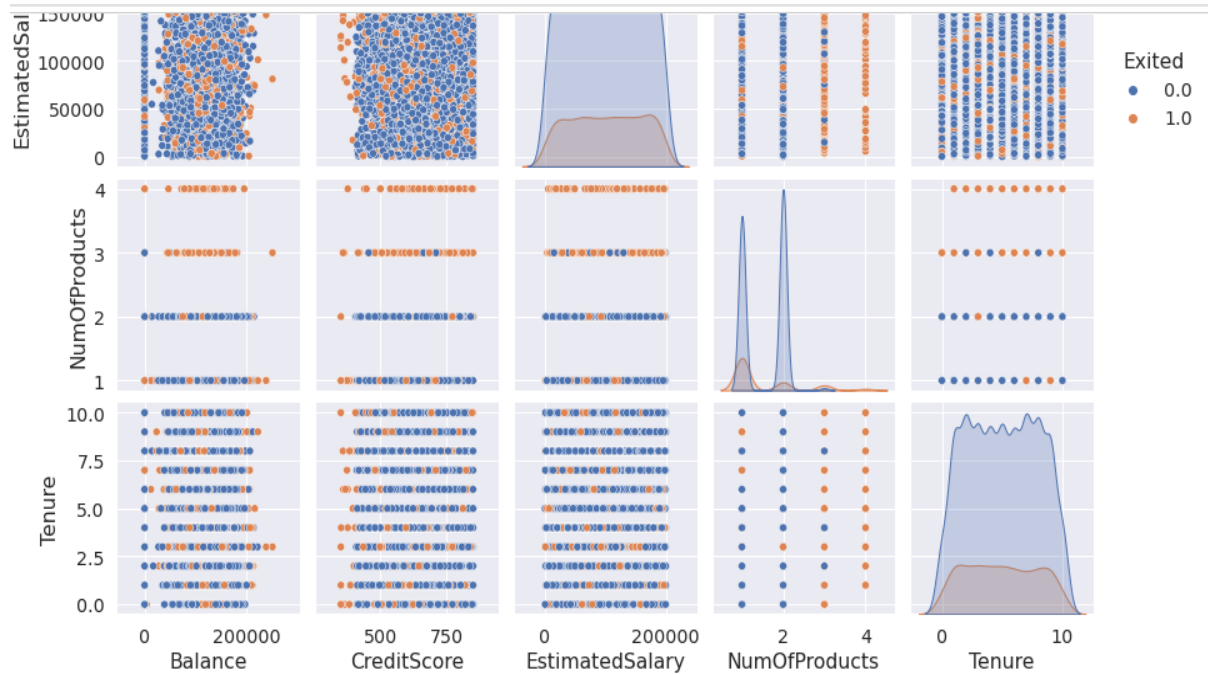
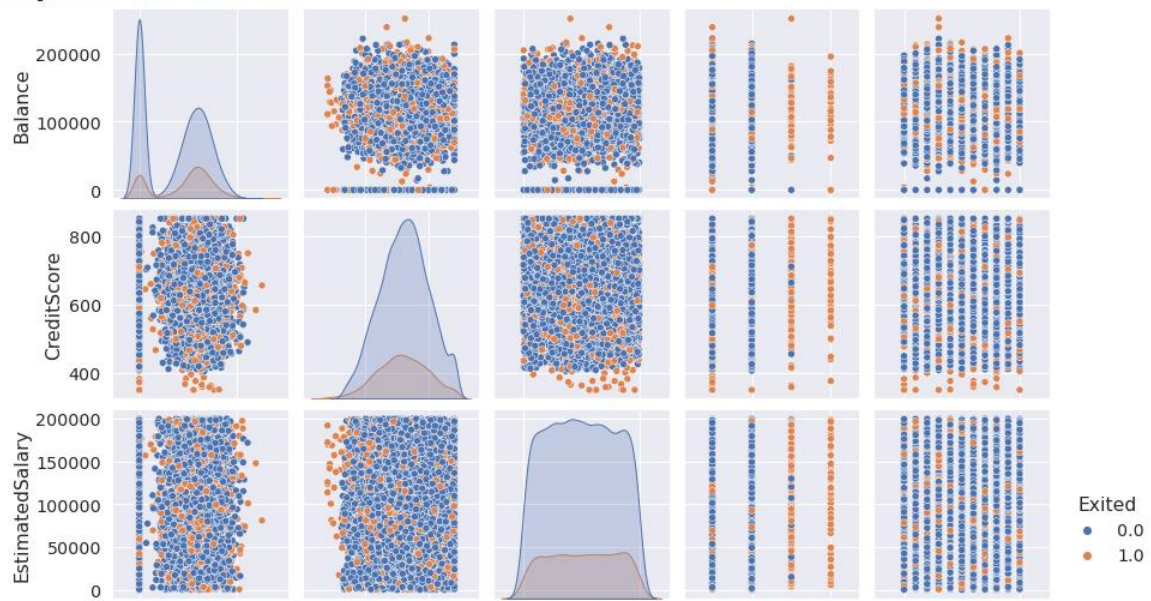
```
=====
```

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 2.32e+05. This might indicate that there are strong multicollinearity or other numerical problems.

```
#Perform Multivariate Analysis
plt.figure(figsize=(4,4))
sns.pairplot(data=df[["Balance", "CreditScore", "EstimatedSalary", "NumOfProducts", "Tenure", "Exited"]], hue="Exited")
```

<Figure size 288x288 with 0 Axes>



### Question-3:

3. Perform descriptive statistics on the dataset.

#### Solution:

```
#Perform Descriptive Statistics
df=pd.DataFrame(df)
print(df.sum())
```

```

CreditScore      6505288.0
Geography        FranceSpainFranceFranceSpainSpainFranceGermany...
Gender           FemaleFemaleFemaleFemaleFemaleMaleMaleFemaleMa...
Age              389218.0
Tenure           50128.0
Balance          764858892.88
NumOfProducts    15302.0
HasCrCard        7055.0
IsActiveMember   5151.0
EstimatedSalary  1000902398.81
Exited           2037.0
dtype: object
```

```
#Perform Descriptive Statistics
print("----Sum Value----")
print(df.sum(1))
print("-----")
print("-----Product Value-----")
print(df.prod())
print("-----")
```

```

----Sum Value----
0      102015.88
1      197002.44
2      274149.37
3       94567.63
4      205492.92
...
9995    97088.64
9996   159633.38
9997    42840.58
9998   168784.83
9999   169159.57
Length: 10000, dtype: float64
-----
-----Product Value-----
CreditScore      inf
Age              inf
Tenure           0.0
Balance          0.0
NumOfProducts    inf
HasCrCard        0.0
IsActiveMember   0.0
EstimatedSalary  inf
Exited           0.0
dtype: float64
-----
```

```
#Perform Descriptive Statistics
print("-----Mean Value-----")
print(df.mean())
print("-----")
```

```

print("-----Median Value-----")
print(df.median())
print("-----")
print("-----Mode Value-----")
print(df.mode())
print("-----")
-----Mean Value-----
CreditScore      650.528800
Age              38.921800
Tenure           5.012800
Balance          76485.889288
NumOfProducts    1.530200
HasCrCard        0.705500
IsActiveMember   0.515100
EstimatedSalary  100090.239881
Exited           0.203700
dtype: float64
-----
-----Median Value-----
CreditScore      652.000
Age              37.000
Tenure           5.000
Balance          97198.540
NumOfProducts    1.000
HasCrCard        1.000
IsActiveMember   1.000
EstimatedSalary  100193.915
Exited           0.000
dtype: float64
-----
-----Mode Value-----
   CreditScore Geography Gender  Age  Tenure  Balance  NumOfProducts
0         850.0    France   Male  37.0     2.0     0.0             1.0

   HasCrCard  IsActiveMember  EstimatedSalary  Exited
0         1.0             1.0         24924.92     0.0
-----

```

#### Question-4:

4.Handle the Missing values

#### Solution:

```

#Handling with missing Values
df.isnull().values;
#Checking values are null

#Handling with missing Values
df.notnull()#Checking values are not null

```

```
Out[16]:
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	True	True	True	True	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True	True	True	True
...	...	...	...	...	...	...	...	...	...	...	...
9995	True	True	True	True	True	True	True	True	True	True	True
9996	True	True	True	True	True	True	True	True	True	True	True
9997	True	True	True	True	True	True	True	True	True	True	True
9998	True	True	True	True	True	True	True	True	True	True	True
9999	True	True	True	True	True	True	True	True	True	True	True

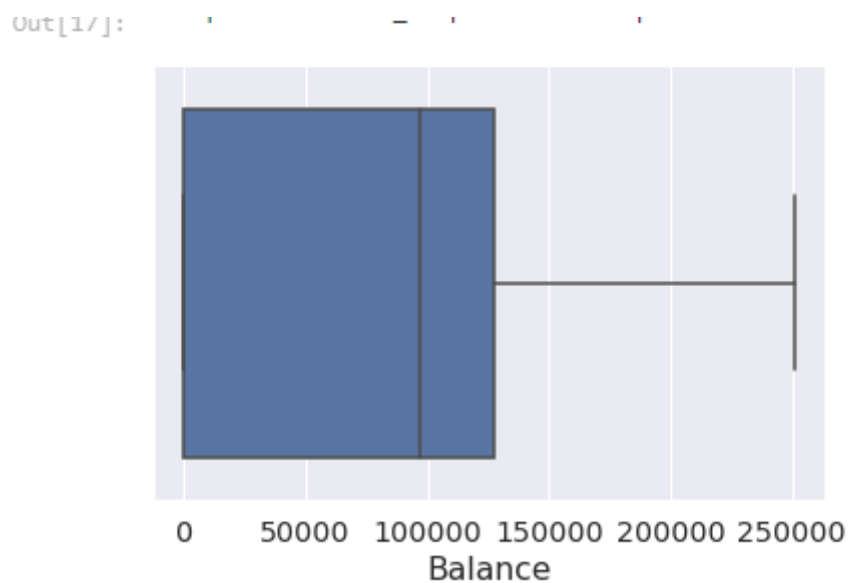
10000 rows × 11 columns

### Question-5:

5. Find the outliers and replace the outliers

#### Solution:

```
#Find outliers & replace the outliers
sns.boxplot(df['Balance'])
```



```
#Find outliers & replace the outliers
print(np.where(df['Balance']>100000))
(array([ 2, 4, 5, ..., 9987, 9993, 9999]),)
```

```
#Find outliers & replace the outliers
from scipy import stats
import numpy as np
```

In [19]:

```

z = np.abs(stats.zscore(df["EstimatedSalary"]))
print(z)

```

---

```

0      0.021886
1      0.216534
2      0.240687
3      0.108918
4      0.365276
...
9995    0.066419
9996    0.027988
9997    1.008643
9998    0.125231
9999    1.076370
Name: EstimatedSalary, Length: 10000, dtype: float64

```

### Question-6:

6. Check for Categorical columns and perform encoding

#### Solution:

```

#Check for categorical columns & performs encoding
from sklearn.preprocessing import LabelEncoder

df['Gender'].unique()
df['Gender'].value_counts()

encoding=LabelEncoder()
df["Gender"]=encoding.fit_transform(df.iloc[:,1].values)
df

```

```

]: #Check for categorical columns & performs encoding
   from sklearn.preprocessing import LabelEncoder
   df['Gender'].unique()

```

```

]: array(['Female', 'Male'], dtype=object)

```

```

]: #Check for categorical columns & performs encoding
   df['Gender'].value_counts()

```

```

]: Male      5457
   Female    4543
   Name: Gender, dtype: int64

```

---



```
Out[22]:
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619.0	France	0	42.0	2.0	0.00	1.0	1.0	1.0	101348.88	1.0
1	608.0	Spain	2	41.0	1.0	83807.86	1.0	0.0	1.0	112542.58	0.0
2	502.0	France	0	42.0	8.0	159660.80	3.0	1.0	0.0	113931.57	1.0
3	699.0	France	0	39.0	1.0	0.00	2.0	0.0	0.0	93826.63	0.0
4	850.0	Spain	2	43.0	2.0	125510.82	1.0	1.0	1.0	79084.10	0.0
...	...	...	...	...	...	...	...	...	...	...	...
9995	771.0	France	0	39.0	5.0	0.00	2.0	1.0	0.0	96270.64	0.0
9996	516.0	France	0	35.0	10.0	57369.61	1.0	1.0	1.0	101699.77	0.0
9997	709.0	France	0	36.0	7.0	0.00	1.0	0.0	1.0	42085.58	1.0
9998	772.0	Germany	1	42.0	3.0	75075.31	2.0	1.0	0.0	92888.52	1.0
9999	792.0	France	0	28.0	4.0	130142.79	1.0	1.0	0.0	38190.78	0.0

10000 rows × 11 columns

### Question-7:

7.Split the data into dependent and independent variables.

#### Solution:

```
#Split the data into Dependent & Independent Variables
print("-----Dependent Variables-----")
X=df.iloc[:,1:4]
print(X)
print("-----")
print("-----Independent Variables-----")
Y=df.iloc[:,4]
print(Y)
print("-----")
```

### Question-8:

8. Scale the independent variables

#### Solution:

```
#Split the data into Dependent & Independent Variables
print("-----Dependent Variables-----")
X=df.iloc[:,1:4]
print(X)
print("-----")
print("-----Independent Variables-----")
Y=df.iloc[:,4]
print(Y)
print("-----")
```

### Question-9:

9. Split the data into training and testing

#### Solution:

```
#Split the data into training & testing
from sklearn.model_selection import train_test_split
```

In [34]:

```
#Split the data into training & testing
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=4, random_state=4)
x_train
x_test
y_train
y_test
Out[31]:
```

	const	EstimatedSalary
1603	1.0	23305.85
8713	1.0	41248.80
4561	1.0	143317.42
6600	1.0	174123.16

```
In [32]: #Split the data into training & testing
y_train
```

```
Out[32]: 2558    727.0
7642    811.0
8912    623.0
3319    430.0
6852    600.0
...
456     733.0
6017    487.0
709     686.0
8366    637.0
1146    614.0
Name: CreditScore, Length: 9996, dtype: float64
```

Out[34]:

	const	EstimatedSalary
2558	1.0	137903.54
7642	1.0	121765.00
8912	1.0	109470.34
3319	1.0	2923.61
6852	1.0	7312.25
...	...	...
456	1.0	7666.73
6017	1.0	9085.00
709	1.0	147794.63
8366	1.0	102515.42
1146	1.0	54776.64

9996 rows × 2 columns