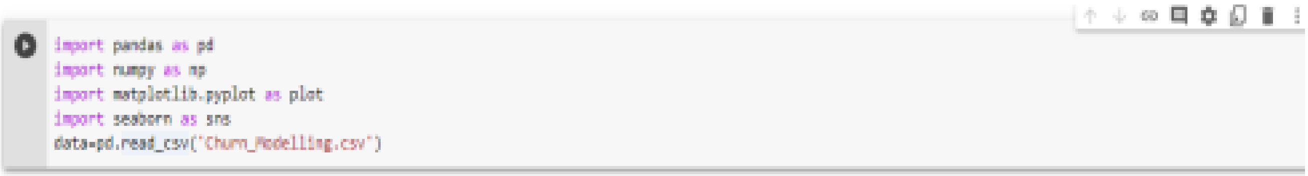
Data Visualization and Pre-processing ASSIGNMENT 2

DATE	**26 SEPTEMBER 2022.**
TEAM ID	**PNT2022TMID38677**
PROJECT NAME	**Fertilizers Recommendation System for Disease Prediction.**
NAME	**K.Saraswathi**

1.Download the dataset

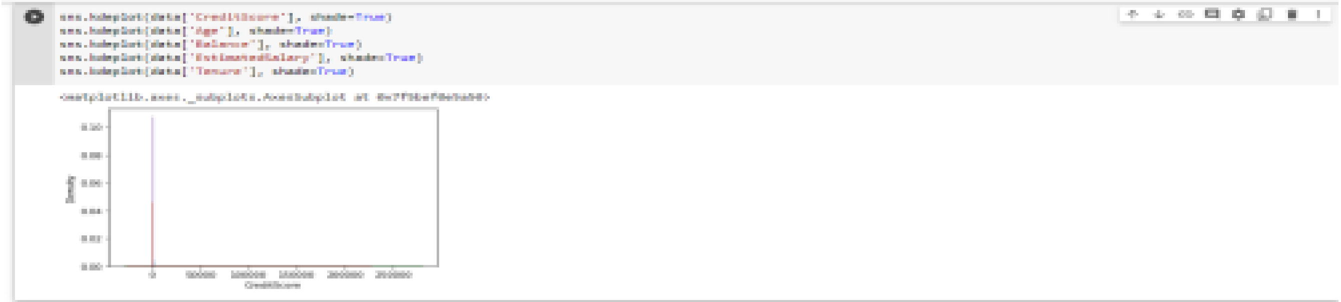
2.Load the dataset



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plot
import seaborn as sns
data=pd.read_csv('Churn_Modelling.csv')
```

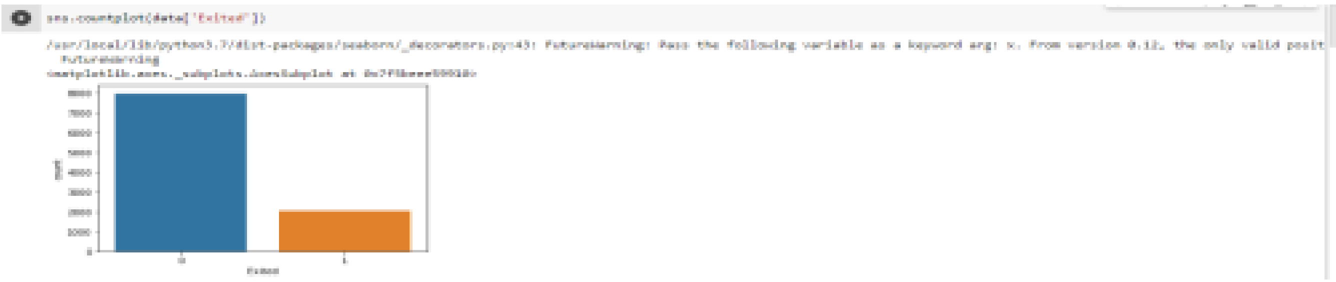
3. perform below visualization

- Univarient
- Bi-varient
- Multi-varient



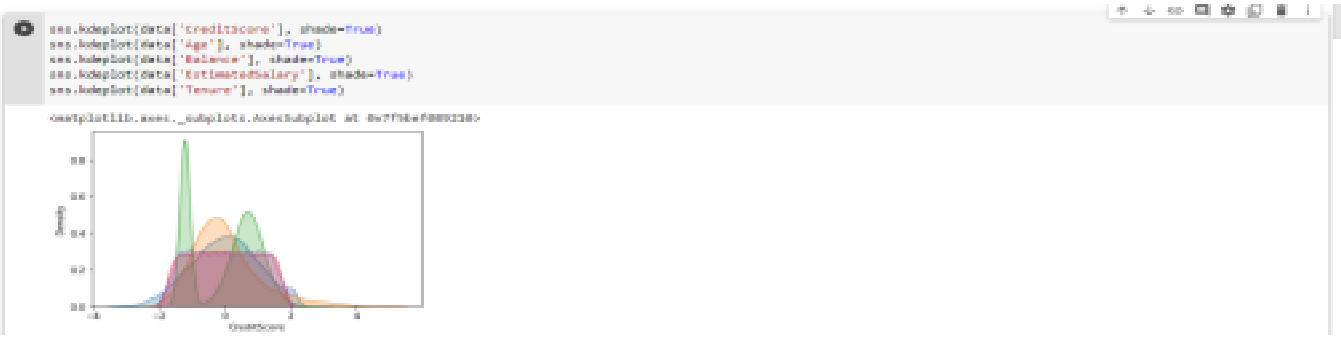
```
sns.kdeplot(data['CreditScore'], shade=True)
sns.kdeplot(data['Age'], shade=True)
sns.kdeplot(data['Balance'], shade=True)
sns.kdeplot(data['EstimatedSalary'], shade=True)
sns.kdeplot(data['Tenure'], shade=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at [illegible]>
```



```
sns.countplot(data['Exited'])
```

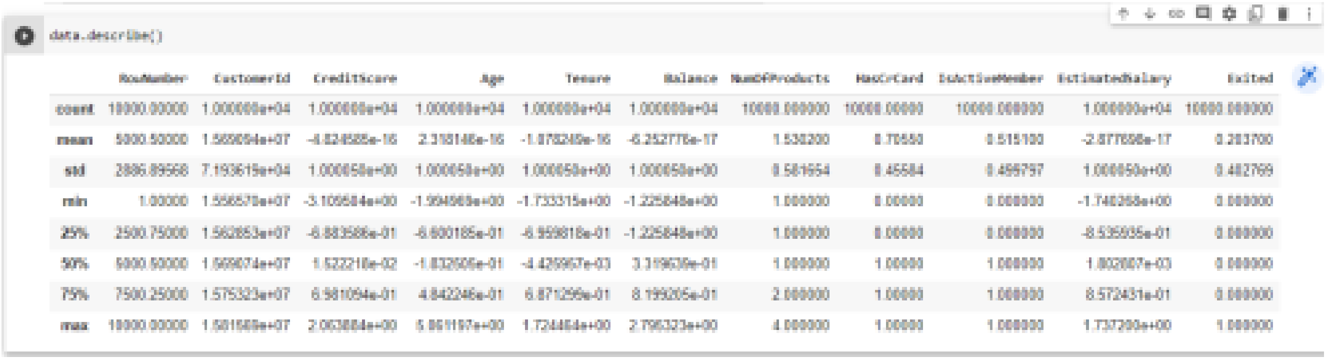
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid posit
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at [illegible]>
```



```
sns.kdeplot(data['CreditScore'], shade=True)
sns.kdeplot(data['Age'], shade=True)
sns.kdeplot(data['Balance'], shade=True)
sns.kdeplot(data['EstimatedSalary'], shade=True)
sns.kdeplot(data['Tenure'], shade=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f56ef089210>
```

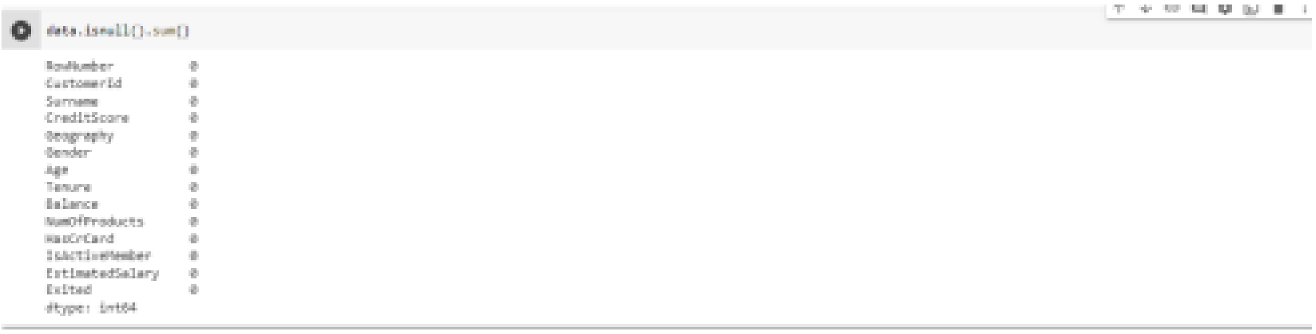
4.Perform the descriptive statistics on the datase



```
data.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	10000.000000	10000.00000	10000.000000	1.000000e+04	10000.000000
mean	5000.50000	1.569094e+07	-4.824585e-16	2.318146e-16	-1.078240e-16	-6.252776e-17	1.530200	0.70550	0.515100	-2.877698e-17	0.203700
std	2886.89568	7.193619e+04	1.000050e+00	1.000050e+00	1.000050e+00	1.000050e+00	0.581654	0.45584	0.499797	1.000050e+00	0.402769
min	1.00000	1.556570e+07	-3.109504e+00	-1.994969e+00	-1.733315e+00	-1.225848e+00	1.000000	0.00000	0.000000	-1.740268e+00	0.000000
25%	2500.75000	1.562853e+07	-6.883586e-01	-6.600185e-01	-6.959818e-01	-1.225848e+00	1.000000	0.00000	0.000000	-8.535935e-01	0.000000
50%	5000.50000	1.569074e+07	1.522218e-02	-1.832505e-01	-4.425957e-03	3.319639e-01	1.000000	1.00000	1.000000	1.802807e-03	0.000000
75%	7500.25000	1.575323e+07	6.981094e-01	4.842246e-01	6.871299e-01	8.199205e-01	2.000000	1.00000	1.000000	8.572431e-01	0.000000
max	10000.00000	1.581569e+07	2.063884e+00	5.061197e+00	1.724464e+00	2.795323e+00	4.000000	1.00000	1.000000	1.737200e+00	1.000000

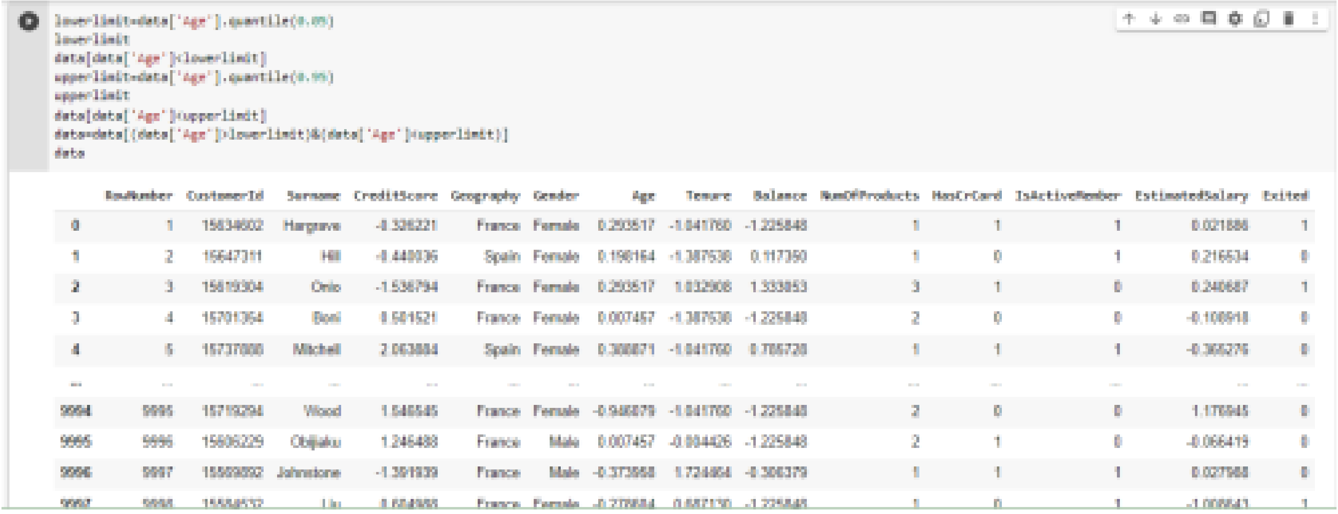
5. Handle the missing values



```
data.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
```

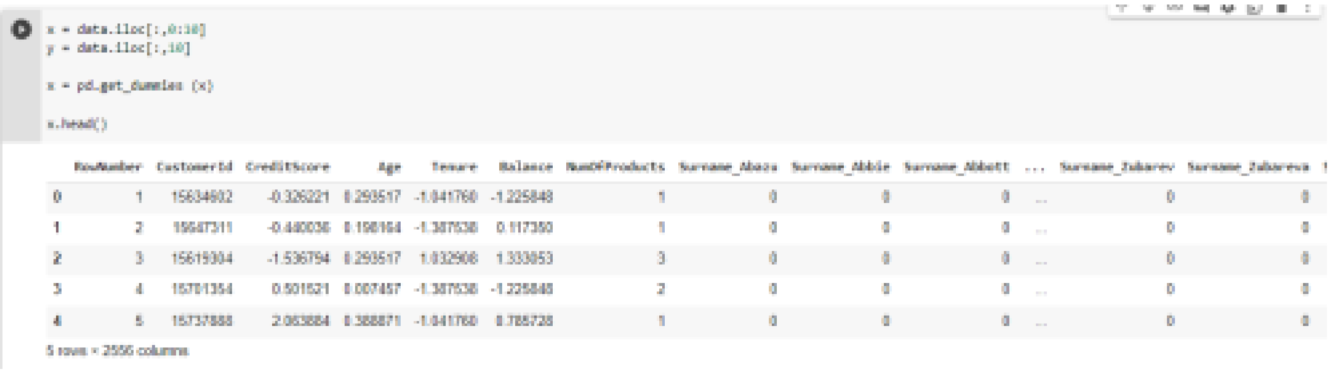
6. Find the outliers and replace the outliers



```
lowerlimit=data['Age'].quantile(0.05)
lowerlimit
data[data['Age']<lowerlimit]
upperlimit=data['Age'].quantile(0.95)
upperlimit
data[data['Age']>upperlimit]
data=data[(data['Age']>lowerlimit)&(data['Age']<upperlimit)]
data
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	-0.326221	France	Female	0.293517	-1.041760	-1.225848	1	1	1	0.021886	1
1	2	15647311	Hill	-0.440036	Spain	Female	0.198164	-1.387538	0.117350	1	0	1	0.216534	0
2	3	15619304	Onio	-1.536794	France	Female	0.293517	1.032908	1.333053	3	1	0	0.240687	1
3	4	15701354	Boni	0.501521	France	Female	0.007457	-1.387538	-1.225848	2	0	0	-0.108918	0
4	5	15737888	Mitchell	2.063884	Spain	Female	0.388871	-1.041760	0.785728	1	1	1	-0.365276	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9994	9995	15719294	Wood	1.246545	France	Female	-0.946079	-1.041760	-1.225848	2	0	0	1.170945	0
9995	9996	15606229	Obijiaku	1.246488	France	Male	0.007457	-0.004426	-1.225848	2	1	0	-0.066419	0
9996	9997	15569892	Johnstone	-1.391939	France	Male	-0.373958	1.724464	-0.306379	1	1	1	0.027988	0
9997	9998	15584532	Liu	0.604988	France	Female	-0.278684	0.687130	-1.225848	1	0	1	-1.008643	1

7.Check the categorical columns and perform encoding

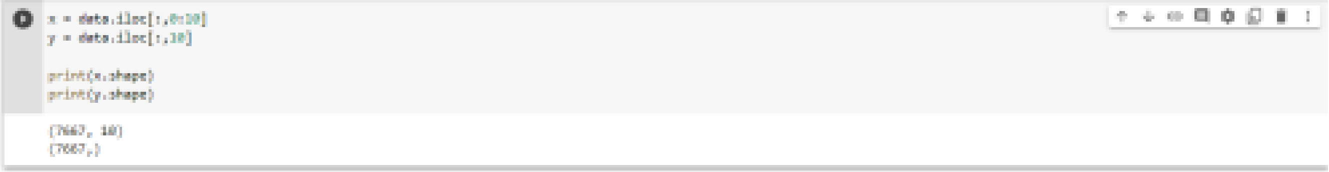


```
x = data.iloc[:,0:10]
y = data.iloc[:,10]

x = pd.get_dummies (x)

x.head()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	Surname_Abaza	Surname_Abbie	Surname_Abbott	...	Surname_Zubarev	Surname_Zubareva
0	1	15634602	-0.326221	0.293517	-1.041760	-1.225848	1	0	0	0	...	0	0
1	2	15647311	-0.440036	0.198164	-1.387538	0.117350	1	0	0	0	...	0	0
2	3	15619304	-1.536794	0.293517	1.032908	1.333053	3	0	0	0	...	0	0
3	4	15701354	0.501521	0.007457	-1.387538	-1.225848	2	0	0	0	...	0	0
4	5	15737888	2.063884	0.388871	-1.041760	0.785728	1	0	0	0	...	0	0

5 rows × 2556 columns

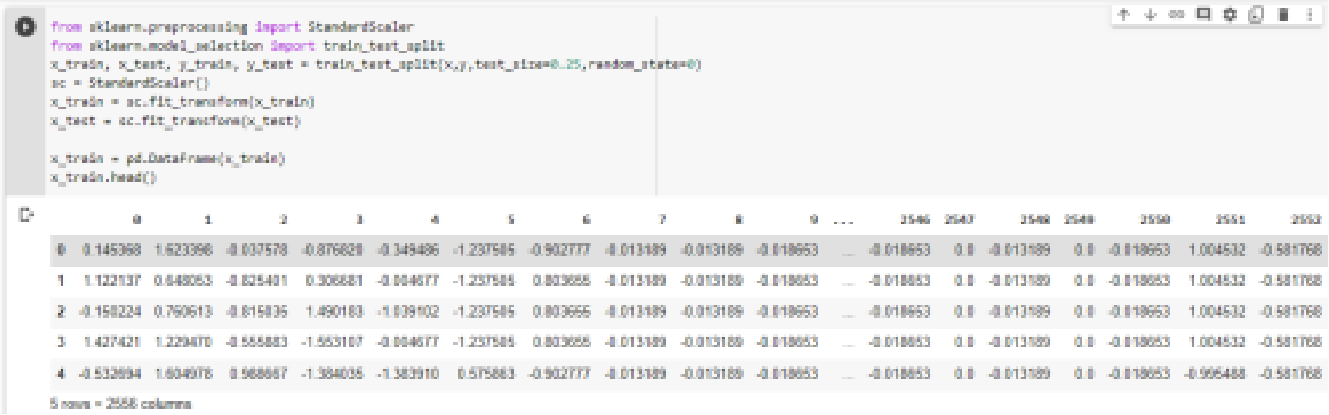
8.Split the dataset into ipdendent and dependent variables.



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

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

```
(7667, 10)
(7667,)
```

9.Scale the independent variable

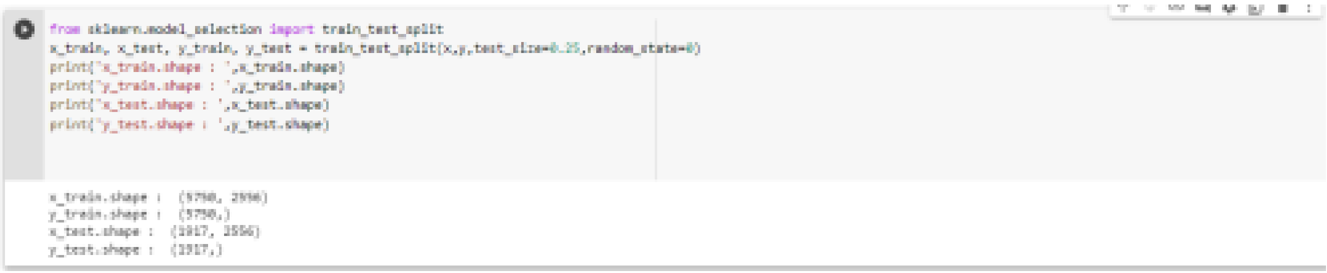


```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25,random_state=0)
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)

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

	0	1	2	3	4	5	6	7	8	9	...	2546	2547	2548	2549	2550	2551	2552
0	0.145368	1.623396	-0.037578	-0.876828	-0.349486	-1.237585	-0.902777	-0.013189	-0.013189	-0.018653	...	-0.018653	0.0	-0.013189	0.0	-0.018653	1.004532	-0.581768
1	1.122137	0.648053	-0.825401	0.306681	-0.004677	-1.237585	0.803655	-0.013189	-0.013189	-0.018653	...	-0.018653	0.0	-0.013189	0.0	-0.018653	1.004532	-0.581768
2	-0.158224	0.760613	-0.815836	1.490183	-1.039102	-1.237585	0.803666	-0.013189	-0.013189	-0.018653	...	-0.018653	0.0	-0.013189	0.0	-0.018653	1.004532	-0.581768
3	1.427421	1.229470	-0.555883	-1.553107	-0.004677	-1.237585	0.803655	-0.013189	-0.013189	-0.018653	...	-0.018653	0.0	-0.013189	0.0	-0.018653	1.004532	-0.581768
4	-0.532694	1.604978	0.988667	-1.384035	-1.383910	0.575863	-0.902777	-0.013189	-0.013189	-0.018653	...	-0.018653	0.0	-0.013189	0.0	-0.018653	-0.995488	-0.581768

5 rows × 2556 columns

10.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 : ',x_train.shape)
print('y_train.shape : ',y_train.shape)
print('x_test.shape : ',x_test.shape)
print('y_test.shape : ',y_test.shape)
```

```
x_train.shape :  (5750, 2556)
y_train.shape :  (5750,)
x_test.shape :  (1917, 2556)
y_test.shape :  (1917,)
```