

Assignment-2

Real-Time Communication System Powered by AI for Specially Abled

Student Name	VIJAYABARATH D
Student Roll no	621319104063
Maximum Marks	2 Marks

Data Visualization and Pre - processing:

1.Load the dataset

+ Code + Text

Connect Editing

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

#load dataset
df = pd.read_csv(r"/content/churn_Modelling.csv")

[ ] df.head(10)
```

	RowNumber	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	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	1
6	7	15592531	Bartlett	822	France	Male	50	7	0.00	2	1	1	10062.80	0
7	8	15656148	Oblinna	376	Germany	Female	29	4	115046.74	4	1	0	119346.88	1
8	9	15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.50	0
9	10	15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.73	0

+ Code + Text

Connect Editing

```
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  
 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

[ ] #visualizations
#univariate Analysis
import seaborn as sns

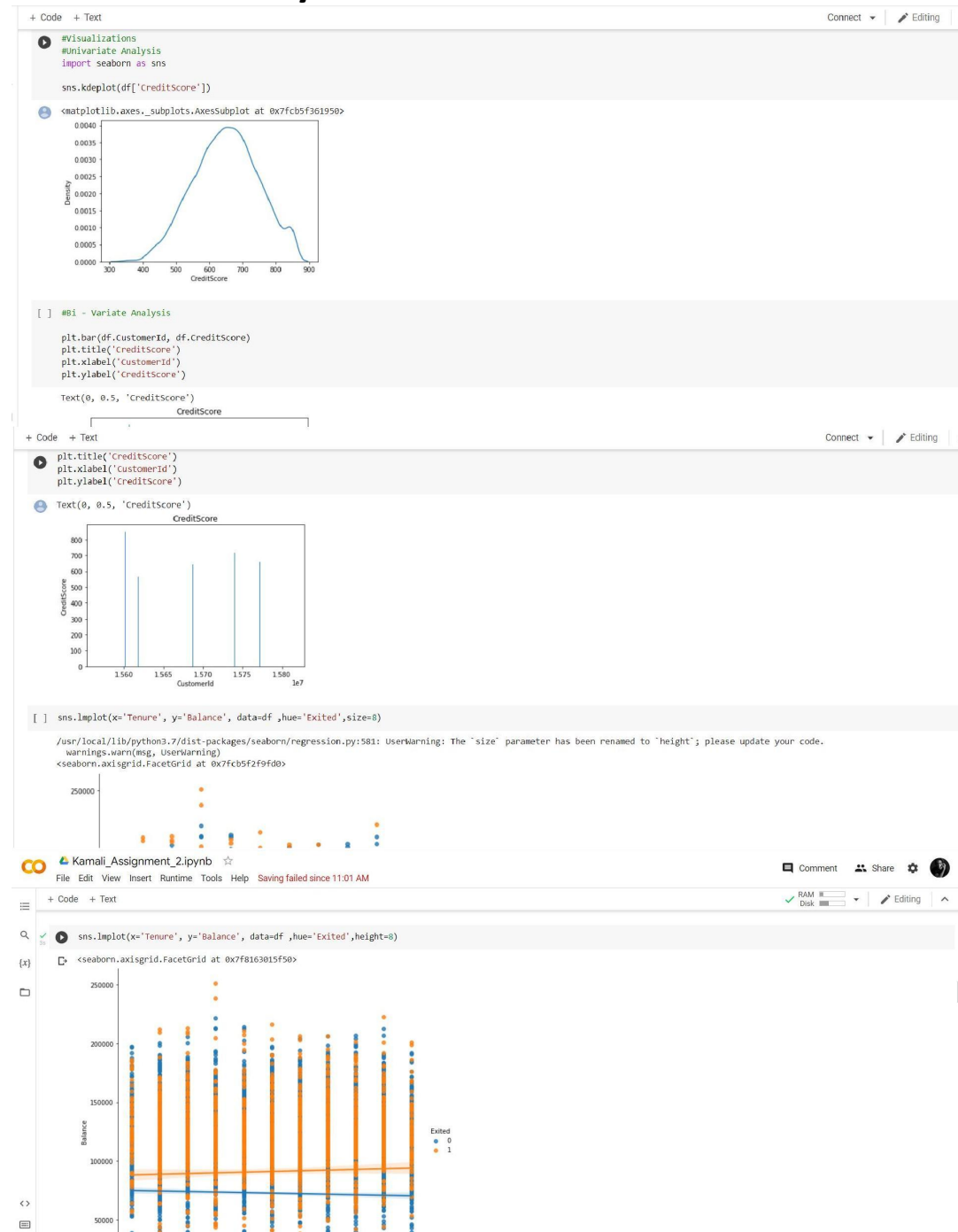
sns.kdeplot(df['CreditScore'])

<matplotlib.axes._subplots.AxesSubplot at 0x7fcb5f361950>

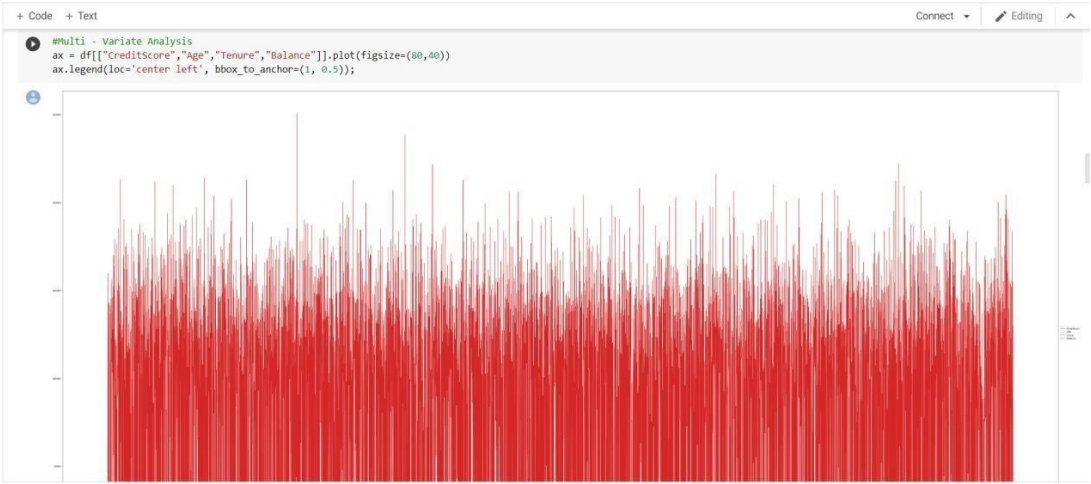
0.0040 [
```

Univariate Analysis

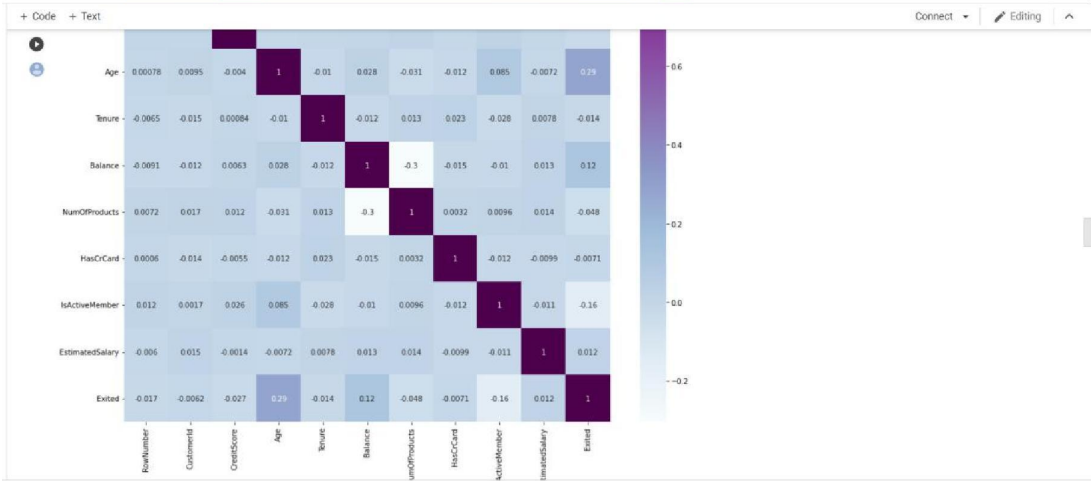
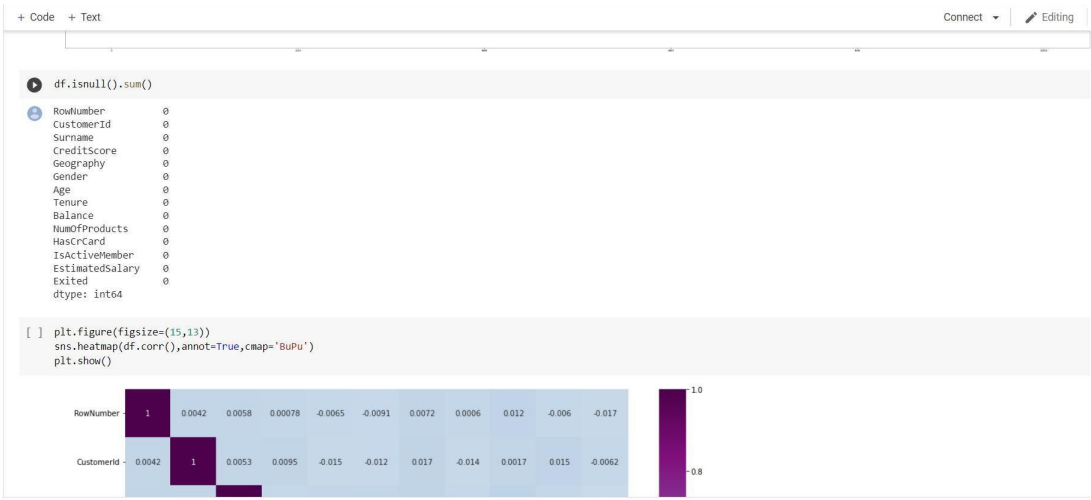
Bi - Variate Analysis



Multi - Variate Analysis



Handle the Missing values.



Perform descriptive statistics on the dataset.

+ Code+ Text

ConnectEditing

```
[ ] df.drop(['RowNumber', 'CustomerId','Surname'],axis=1,inplace=True)
```

```
[ ] df.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	689	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
#   Column              Non-Null Count  Dtype
---  --
0   CreditScore          10000 non-null  int64
1   Geography            10000 non-null  object
2   Gender               10000 non-null  object
3   Age                  10000 non-null  int64
4   Tenure               10000 non-null  int64
5   Balance              10000 non-null  float64
6   NumOfProducts        10000 non-null  int64
7   HasCrCard            10000 non-null  int64
8   IsActiveMember       10000 non-null  int64
9   EstimatedSalary      10000 non-null  float64
10  Exited               10000 non-null  int64
dtypes: float64(2), int64(7), object(2)
memory usage: 859.5+ KB
```

+ Code+ Text

ConnectEditing

```
[ ] df["Geography"].unique()
array(['France', 'Spain', 'Germany'], dtype=object)
```

```
[ ] df["Gender"].unique()
array(['Female', 'Male'], dtype=object)
```

```
[ ] geo=pd.get_dummies(df["Geography"],drop_first=False)
```

```
[ ] geo.head()
```

	France	Germany	Spain
0	1	0	0
1	0	0	1
2	1	0	0
3	1	0	0
4	0	0	1

```
[ ] gen=pd.get_dummies(df["Gender"],drop_first=False)
```

+ Code+ Text

ConnectEditing

```
[ ] df=pd.concat([df, geo,gen], axis=1)
```

```
[ ] df
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	France	Germany	Spain	Female	Male
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1	1	0	0	1	0
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0	0	0	1	1	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1	1	0	0	1	0
3	689	France	Female	39	1	0.00	2	0	0	93826.63	0	1	0	0	1	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0	0	0	1	1	0
...
9995	771	France	Male	39	5	0.00	2	1	0	96270.64	0	1	0	0	0	1
9996	516	France	Male	35	10	57369.61	1	1	1	101699.77	0	1	0	0	0	1
9997	709	France	Female	36	7	0.00	1	0	1	42085.58	1	1	0	0	1	0
9998	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1	0	1	0	0	1
9999	792	France	Female	28	4	130142.79	1	1	0	38190.78	0	1	0	0	1	0

10000 rows x 16 columns

```
[ ] df.drop(["Geography","Gender"], axis=1, inplace=True)
```

+ Code
+ Text

Connect
Editing

10000 rows x 16 columns

```
[ ] df.drop(["Geography","gender"], axis=1, inplace=True)
```

```
[ ] df.head()
```

	Creditscore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	France	Germany	Spain	Female	Male
0	619	42	2	0.00	1	1	1	101348.88	1	1	0	0	1	0
1	608	41	1	83807.86	1	0	1	112542.58	0	0	0	1	1	0
2	502	42	8	159660.80	3	1	0	113931.57	1	1	0	0	1	0
3	699	39	1	0.00	2	0	0	93826.63	0	1	0	0	1	0
4	850	43	2	125510.82	1	1	1	79084.10	0	0	0	1	1	0

```
[ ] x=df.drop('Exited',axis=1)
```

```
[ ] x
```

	Creditscore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	France	Germany	Spain	Female	Male
0	619	42	2	0.00	1	1	1	101348.88	1	0	0	1	0
1	608	41	1	83807.86	1	0	1	112542.58	0	0	1	1	0
2	502	42	8	159660.80	3	1	0	113931.57	1	0	0	1	0
3	699	39	1	0.00	2	0	0	93826.63	1	0	0	1	0

Check for Categorical columns and perform encoding. Split the data into dependent and independent variables.

+ Code
+ Text

Connect
Editing

10000 rows x 13 columns

```
[ ] y=df['Exited']
```

```
[ ] y
```

```

0      1
1      0
2      1
3      0
4      0
...
9995    1
9996    0
9997    1
9998    1
9999    0
Name: Exited, Length: 10000, dtype: int64

```

Scale the independent variables

+ Code
+ Text

Connect
Editing

```
[ ] df.shape
```

```
(10000, 14)
```

```
[ ] x.shape
```

```
(10000, 13)
```

```
[ ] y.shape
```

```
(10000,)
```

```
[ ] 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_state=0)
```

```
[ ] x_train.shape
```

```
(8000, 13)
```

```
[ ] x_test.shape
```

```
(2000, 13)
```

Split the data into training and testing

+ Code + Text

Connect ▾

```
(2000, 13)
```

```
[ ] y_test.shape
```

```
(2000,)
```

```
[ ] from sklearn.preprocessing import StandardScaler
```

```
[ ] sc = StandardScaler()
```

```
• x_train = sc.fit_transform(x_train)
```

```
[ ] x_train
```

```
array([[ 0.16958176, -0.46460796,  0.00666099, ...,  1.74309049,
         1.09168714, -1.09168714],
       [-2.30455045,  0.30102557, -1.37744033, ..., -0.57369368,
        -0.91601335,  0.91601335],
       [-1.19119591, -0.94312892, -1.031415 , ..., -0.57369368,
        1.09168714, -1.09168714],
       ...,
       [ 0.9015152 , -0.36890377,  0.00666099, ..., -0.57369368,
        -0.91601335,  0.91601335],
       [-0.62420521, -0.08179119,  1.39076231, ...,  1.74309049,
        1.09168714, -1.09168714],
       [-0.28401079,  0.87525072, -1.37744033, ..., -0.57369368,
        1.09168714, -1.09168714]])
```

```
[ ] x_test = sc.transform(x_test)
```

+ Code + Text

Connect ▾

```
[ ]
```

```
array([[ 1.09168714, -1.09168714],
       [-0.28401079,  0.87525072, -1.37744033, ..., -0.57369368,
        1.09168714, -1.09168714]])
```

```
[ ] x_test = sc.transform(x_test)
```

```
• x_test
```

```
array([[ -0.55204276, -0.36890377,  1.04473698, ..., -0.57369368,
         1.09168714, -1.09168714],
       [-1.31490297,  0.10961719, -1.031415 , ..., -0.57369368,
        1.09168714, -1.09168714],
       [ 0.57162971,  0.30102557,  1.04473698, ...,  1.74309049,
        1.09168714, -1.09168714],
       ...,
       [-0.74791227, -0.27319958, -1.37744033, ...,  1.74309049,
        -0.91601335,  0.91601335],
       [-0.00566991, -0.46460796, -0.33936434, ..., -0.57369368,
        -0.91601335,  0.91601335],
       [-0.79945688, -0.84742473,  1.04473698, ..., -0.57369368,
        -0.91601335,  0.91601335]])
```