Assignment -2

Data virtualization and pre-processing

Assignment Date	26 September 2022
Student Name	T.KIRUTHIKA
Student Roll Number	923119106002
Maximum Marks	2 Marks

Q1. Download the dataset:

Ans: The dataset is downloaded from the question paper.

```
# Importing the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Q2. Load the dataset

Ans:

```
# 2 Reading the dataset
df = pd.read_csv('/content/Churn_Modelling.csv')
```

Q3. Perform Below Visualizations.

Ans:

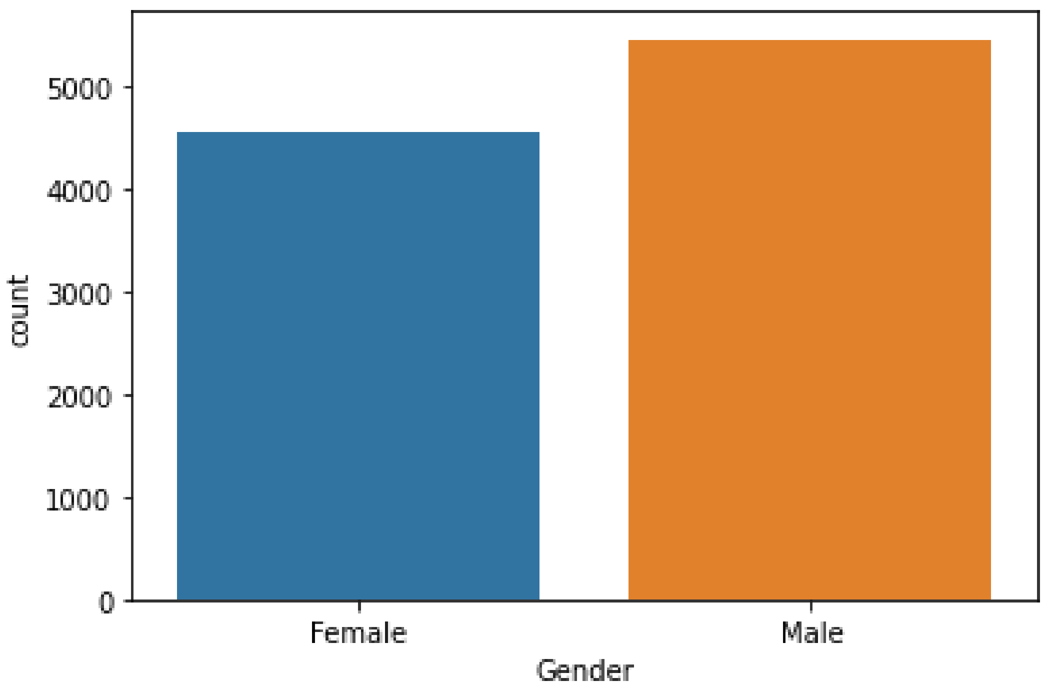
```
# 3 Visualize the data
df.head()
```

```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
0          1    15634602  Hargrave          619    France  Female   42
1          2    15647311      Hill          608     Spain  Female   41
2          3    15619304      Onio          502    France  Female   42
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              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
```

```python
# 3a Univariate Analysis of gender
sns.countplot(df['Gender'])
```

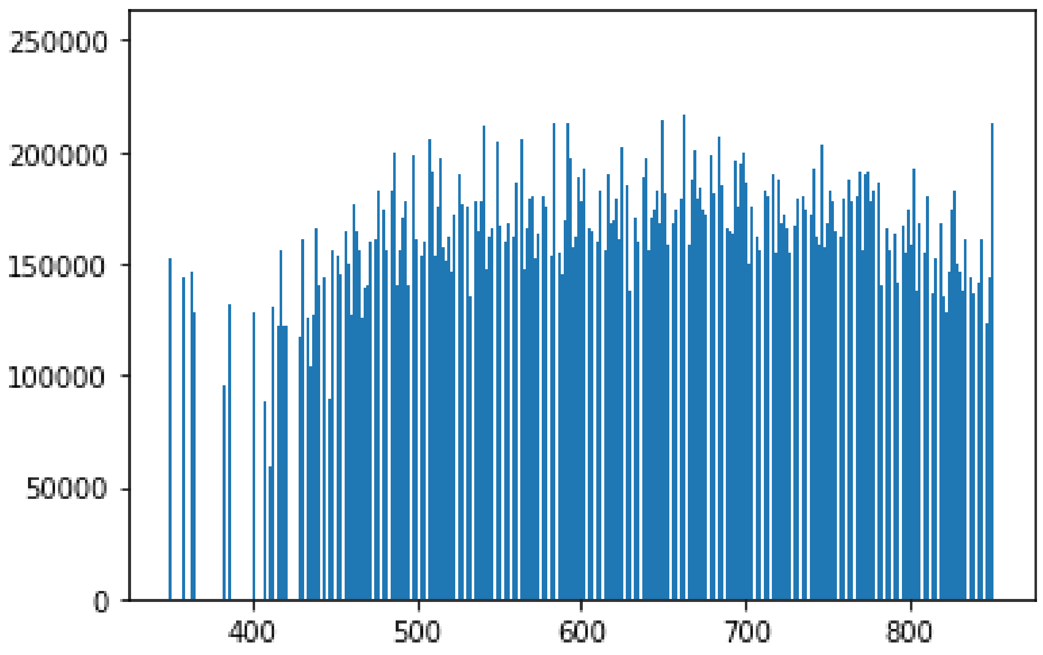
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f77baab4a90>
```



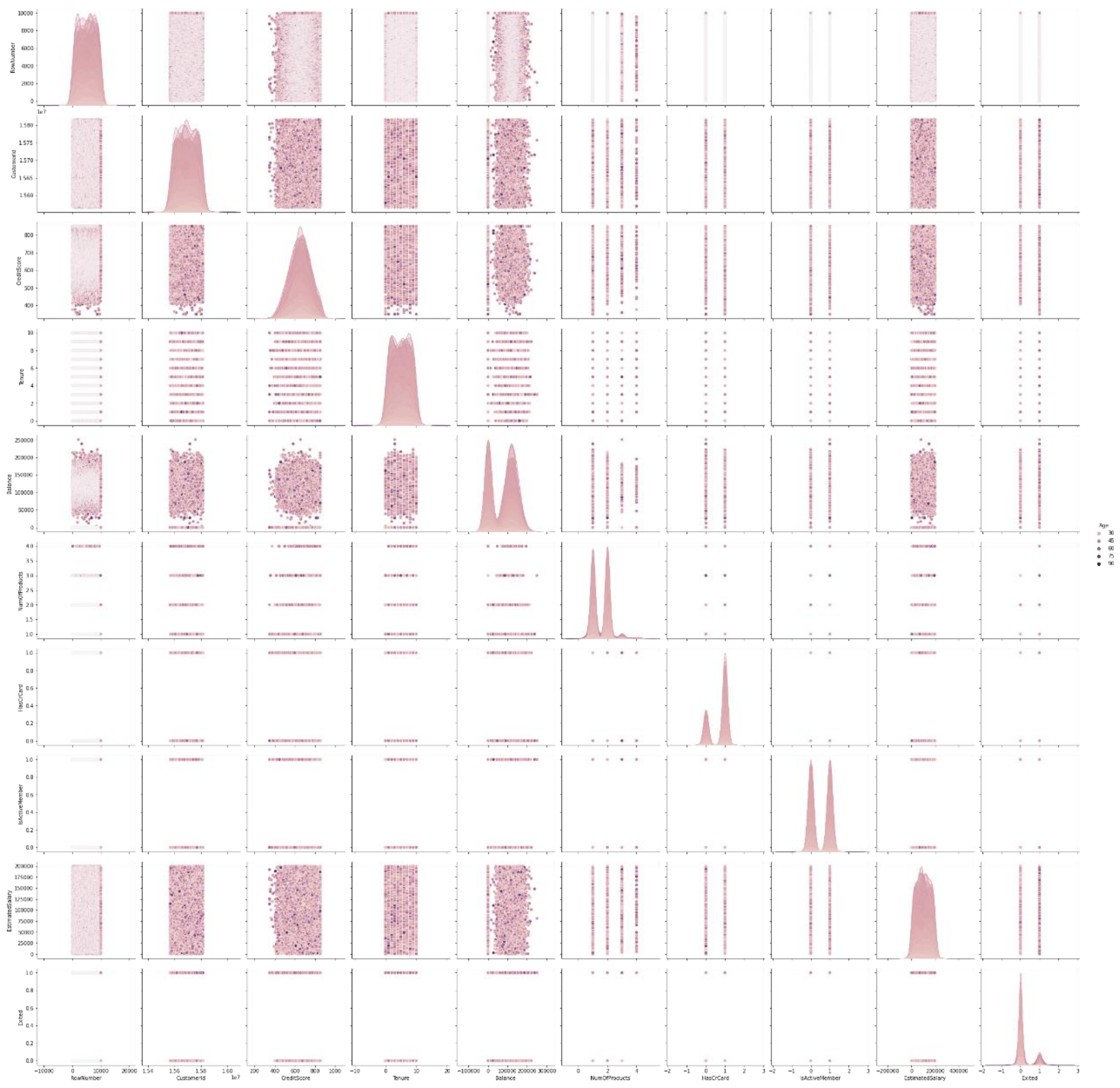
```python
# 3b Bi - Variate Analysis of credit score and balance
plt.bar(df['CreditScore'],df['Balance'])
```

```
<BarContainer object of 10000 artists>
```



```
# 3c multivaiate

sns.pairplot(df,hue='Age',size= 3)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:2076:
UserWarning: The `size` parameter has been renamed to `height`; please
update your code.
  warnings.warn(msg, UserWarning)

<seaborn.axisgrid.PairGrid at 0x7f77b39f5f90>
```



Q4. Perform descriptive statistics on the dataset.

Ans:

```
# 4 Descriptive statistics on the data

df['Balance'].describe()

count     10000.000000
mean      76485.889288
std       62397.405202
min           0.000000
25%           0.000000
50%       97198.540000
75%      127644.240000
max      250898.090000
Name: Balance, dtype: float64

# 4

df['CreditScore'].describe()
```

```
count    10000.000000
mean       650.528800
std         96.653299
min        350.000000
25%        584.000000
50%        652.000000
75%        718.000000
max        850.000000
Name: CreditScore, dtype: float64
```

```
# 4

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

```
850    233
678     63
655     54
705     53
667     53
      ...
404      1
351      1
365      1
417      1
419      1
Name: CreditScore, Length: 460, dtype: int64
```

Q5. Handle the Missing values.

Ans:

```
# 5 Handle the missing values

df.isnull()
```

```
      RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  
Age  \
0         False       False    False        False      False   False  
False
1         False       False    False        False      False   False  
False
2         False       False    False        False      False   False  
False
3         False       False    False        False      False   False  
False
4         False       False    False        False      False   False  
False
...         ...         ...      ...          ...        ...     ...  
...
9995      False       False    False        False      False   False  
False
9996      False       False    False        False      False   False  
False
9997      False       False    False        False      False   False  
False
```

```
9998      False       False     False         False       False    False
False
9999      False       False     False         False       False    False
False

      Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0      False    False          False      False           False
1      False    False          False      False           False
2      False    False          False      False           False
3      False    False          False      False           False
4      False    False          False      False           False
...      ...      ...            ...        ...             ...
9995   False    False          False      False           False
9996   False    False          False      False           False
9997   False    False          False      False           False
9998   False    False          False      False           False
9999   False    False          False      False           False

      EstimatedSalary  Exited
0               False   False
1               False   False
2               False   False
3               False   False
4               False   False
...               ...     ...
9995            False   False
9996            False   False
9997            False   False
9998            False   False
9999            False   False

[10000 rows x 14 columns]
```

Q6. Find the outliers and replace the outliers

Ans:

```
# 6 Find the outliers and replace the outliers

df['CustomerId']>10
```

```
0       True
1       True
2       True
3       True
4       True
        ...
9995    True
9996    True
9997    True
9998    True
9999    True
Name: CustomerId, Length: 10000, dtype: bool
```

```
# 6

df[(df['CustomerId']>7) | (df['CreditScore']<4)]

      RowNumber  CustomerId    Surname  CreditScore Geography  Gender  Age
\
0             1    15634602   Hargrave          619    France  Female   42
1             2    15647311       Hill          608     Spain  Female   41
2             3    15619304       Onio          502    France  Female   42
3             4    15701354       Boni          699    France  Female   39
4             5    15737888   Mitchell          850     Spain  Female   43
...         ...         ...        ...          ...       ...     ...  ...
9995       9996    15606229   Obijiaku          771    France    Male   39
9996       9997    15569892  Johnstone          516    France    Male   35
9997       9998    15584532        Liu          709    France  Female   36
9998       9999    15682355  Sabbatini          772   Germany    Male   42
9999      10000    15628319     Walker          792    France  Female   28

      Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0          2       0.00              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
...      ...        ...            ...        ...             ...
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0

      EstimatedSalary  Exited
0           101348.88       1
1           112542.58       0
2           113931.57       1
3            93826.63       0
4            79084.10       0
...               ...     ...
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
9998         92888.52       1
9999         38190.78       0

[10000 rows x 14 columns]

# 6

df = pd.DataFrame(np.random.randn(100,3))

import numpy as np
from scipy import stats
df[(np.abs(stats.zscore(df))<3).all(axis=1)]
```

```
           0         1         2
0  -0.440795  0.690842 -0.887450
1   0.003317 -0.324481  0.062645
2   0.876527 -0.390660 -0.650967
3  -0.842847 -0.042569  0.569537
4   2.078568 -0.179059  0.041314
..       ...       ...       ...
95  1.209974  0.296284 -0.655162
96 -0.161975  0.190352 -1.799046
97 -0.294117  1.636870 -0.757824
98 -0.757817 -0.307838 -1.983400
99  1.645722 -0.463386  0.045560

[100 rows x 3 columns]
```

Q7. Check for Categorical columns and perform encoding.

Ans:

```
# 7 check catagorical

categorical_feature = [i for i in df.columns if df[i].dtype=='O']
df[categorical_feature].head()
```

```
Empty DataFrame
Columns: []
Index: [0, 1, 2, 3, 4]
```

```
for i in categorical_feature:
  print('{} values found in the feature {}'.format(len(df[i].unique()),i))
```

```
# Importing the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
# 2 Reading the dataset
df = pd.read_csv('/content/Churn_Modelling.csv')
```

```
# 7 perform encoder

from sklearn.preprocessing import LabelEncoder

x = df.drop('Surname', axis=1)
y = df['Surname']

le= LabelEncoder()

y = le.fit_transform(y)

y
```

```
array([1115, 1177, 2040, ..., 1570, 2345, 2751])
```

Q8. Split the data into dependent and independent variables.

Q9. Scale the independent variables

Ans:

```
# 8 and 9 Split the data (independent and dependent)

x = df.iloc[:,0:4].values
y = df.iloc[:,4:5].values

x

array([[1, 15634602, 'Hargrave', 619],
       [2, 15647311, 'Hill', 608],
       [3, 15619304, 'Onio', 502],
       ...,
       [9998, 15584532, 'Liu', 709],
       [9999, 15682355, 'Sabbatini', 772],
       [10000, 15628319, 'Walker', 792]], dtype=object)

y

array([['France'],
       ['Spain'],
       ['France'],
       ...,
       ['France'],
       ['Germany'],
       ['France']], dtype=object)
```

Q10. Split the data into training and testing

Ans:

```
# 10 Split the data (traing anf testing)

from sklearn.model_selection import train_test_split

xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.3,
random_state=0)

xtrain.shape,xtest.shape

((7000, 4), (3000, 4))

ytrain.shape,ytest.shape

((7000, 1), (3000, 1))
```