

ASSIGNMENT - 2

Data Visualization and Pre - processing

| | |
|---------------------|-------------------|
| Assignment Date | 27 September 2022 |
| Student Name | R. RAJALAKSHMI |
| Student Roll Number | 420619104028 |
| Maximum marks | 2 Mark |

1.Download the dataset : Dataset

<https://drive.google.com/file/d/160K6XcuYDyRBPGj-JsqThkyFoJhCvOWy/view?usp=sharing>

2.Load the dataset.

```
In [2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

In [3]: df = pd.read_csv(r"E:\SB\Dataset\Churn_Modelling.csv")

In [4]: df

Out[4]:
```

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary |
|------|-----------|------------|-----------|-------------|-----------|--------|-----|--------|-----------|---------------|-----------|----------------|-----------------|
| 0 | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 | 1 | 10134 |
| 1 | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 | 1 | 11254 |
| 2 | 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 | 0 | 11393 |
| 3 | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 | 0 | 9382 |
| 4 | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 | 1 | 7908 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 9995 | 9996 | 15606229 | Obijaku | 771 | France | Male | 39 | 5 | 0.00 | 2 | 1 | 0 | 9627 |
| 9996 | 9997 | 15569892 | Johnstone | 516 | France | Male | 35 | 10 | 57369.61 | 1 | 1 | 1 | 10169 |
| 9997 | 9998 | 15584532 | Liu | 709 | France | Female | 36 | 7 | 0.00 | 1 | 0 | 1 | 4208 |
| 9998 | 9999 | 15682355 | Sabbatini | 772 | Germany | Male | 42 | 3 | 75075.31 | 2 | 1 | 0 | 9288 |
| 9999 | 10000 | 15628319 | Walker | 792 | France | Female | 28 | 4 | 130142.79 | 1 | 1 | 0 | 3819 |

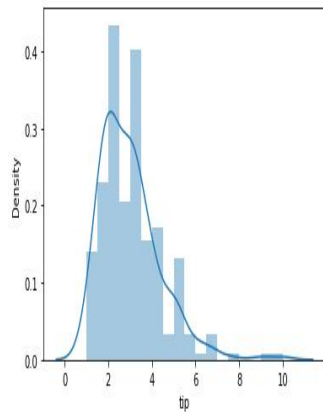
10000 rows x 14 columns

3. Perform Below Visualizations:

* Univariate Analysis

```
sns.distplot(tips['tip'])
```

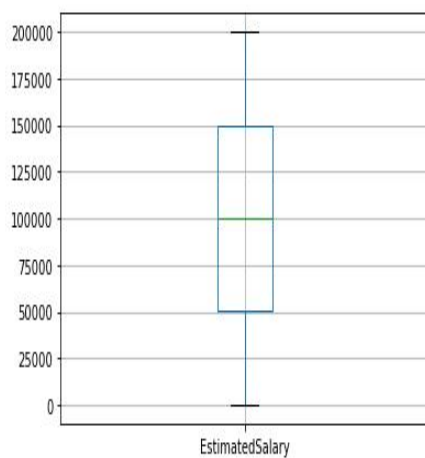
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version.  
warnings.warn(msg, FutureWarning)  
<matplotlib.axes._subplots.AxesSubplot at 0x7f8cd4ef5310>
```



* Bi - Variate Analysis

```
In [11]: df.boxplot(column="EstimatedSalary")
```

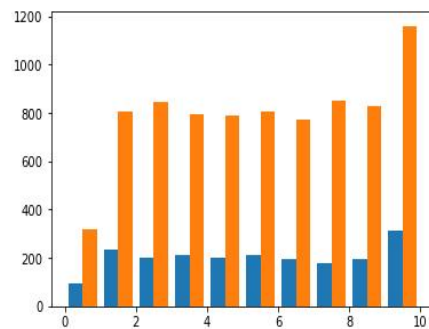
```
Out[11]: <AxesSubplot:>
```



* Multi - Variate Analysis

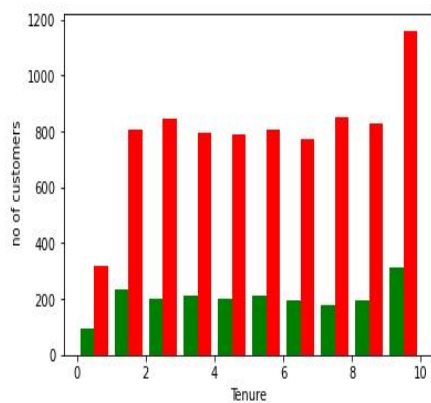
```
In [20]: plt.hist([chrun_yes,chrun_no])
```

```
Out[20]: (array([[ 95., 232., 201., 213., 203., 209., 196., 177., 197.,  
                 314.],  
          [ 318., 803., 847., 796., 786., 803., 771., 851., 828.,  
          1160.]]),  
 array([ 0., 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.]),  
 <a list of 2 BarContainer objects>)
```



```
In [25]: plt.hist([chrun_yes,chrun_no],color=["green","red"])  
plt.xlabel("Tenure")  
plt.ylabel(" no of customers")
```

```
Out[25]: Text(0, 0.5, ' no of customers')
```



4. Perform descriptive statistics on the dataset

```
In [11]: df.describe()
```

```
Out[11]:
```

| | RowNumber | CustomerId | CreditScore | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary |
|-------|-------------|--------------|--------------|--------------|--------------|---------------|---------------|--------------|----------------|-----------------|
| count | 10000.00000 | 1.000000e+04 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 |
| mean | 5000.50000 | 1.569094e+07 | 650.528800 | 38.921800 | 5.012800 | 76485.889288 | 1.530200 | 0.70550 | 0.515100 | 100090.239881 |
| std | 2886.89568 | 7.193619e+04 | 96.653299 | 10.487806 | 2.892174 | 62397.405202 | 0.581654 | 0.45584 | 0.499797 | 57510.492818 |
| min | 1.00000 | 1.556570e+07 | 350.000000 | 18.000000 | 0.000000 | 0.000000 | 1.000000 | 0.00000 | 0.000000 | 11.580000 |
| 25% | 2500.75000 | 1.562853e+07 | 584.000000 | 32.000000 | 3.000000 | 0.000000 | 1.000000 | 0.00000 | 0.000000 | 51002.110000 |
| 50% | 5000.50000 | 1.569074e+07 | 652.000000 | 37.000000 | 5.000000 | 97198.540000 | 1.000000 | 1.00000 | 1.000000 | 100193.915000 |
| 75% | 7500.25000 | 1.575323e+07 | 718.000000 | 44.000000 | 7.000000 | 127644.240000 | 2.000000 | 1.00000 | 1.000000 | 149388.247500 |
| max | 10000.00000 | 1.581569e+07 | 850.000000 | 92.000000 | 10.000000 | 250898.090000 | 4.000000 | 1.00000 | 1.000000 | 199992.480000 |

```
In [13]: df["RowNumber"].mean()
```

```
Out[13]: 5000.5
```

```
In [14]: df["EstimatedSalary"].median()
```

```
Out[14]: 100193.915
```

```
In [15]: df["Exited"].mode()
```

```
Out[15]: 0    0
          Name: Exited, dtype: int64
```

5. Handle the Missing values.

```
In [13]: df.isnull().any()
```

```
Out[13]: RowNumber      False
         CustomerId     False
         Surname        False
         CreditScore     False
         Geography      False
         Gender         False
         Age            False
         Tenure         False
         Balance        False
         NumOfProducts  False
         HasCrCard      False
         IsActiveMember False
         EstimatedSalary False
         Exited         False
         dtype: bool
```

```
In [14]: df.isnull().sum()
```

```
Out[14]: 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

```
In [16]: df.skew()
```

C:\Users\Rajii\AppData\Local\Temp\ipykernel_3520\1665899112.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.
df.skew()

```
Out[16]: RowNumber      0.000000
         CustomerId     0.001149
         CreditScore    -0.071607
         Age            1.011320
         Tenure         0.010991
         Balance       -0.141109
         NumOfProducts  0.745568
         HasCrCard     -0.901812
         IsActiveMember -0.060437
         EstimatedSalary 0.002085
         Exited        1.471611
         dtype: float64
```

7. Check for Categorical columns and perform encoding.

```
In [22]: df.dtypes
Out[22]: RowNumber      int64
CustomerId    int64
Surname       object
CreditScore   int64
Geography     object
Gender        object
Age           int64
Tenure        int64
Balance       float64
NumOfProducts int64
HasCrCard     int64
IsActiveMember int64
EstimatedSalary float64
Exited        int64
dtype: object

In [23]: df["Geography"].unique()
Out[23]: array(['France', 'Spain', 'Germany'], dtype=object)

In [24]: df["Gender"].unique()
Out[24]: array(['Female', 'Male'], dtype=object)
```

```
In [20]: from sklearn.compose import ColumnTransformer

In [21]: from sklearn.preprocessing import OneHotEncoder

In [22]: ct = ColumnTransformer([("oh", OneHotEncoder(), [1,2])], remainder="passthrough")

In [24]: x = ct.fit_transform(x)

In [25]: x.shape
Out[25]: (10000, 15)
```

8. Split the data into dependent and independent variables

```
In [17]: # dependent and independent variables

In [18]: x = df.iloc[:,3:13].values
         y = df.iloc[:,13:14].values

In [19]: x.shape
Out[19]: (10000, 10)

In [20]: y.shape
Out[20]: (10000, 1)
```

9. Scale the independent variables

```
In [16]: y = df.iloc[:,13:14].values
```

```
In [19]: y.shape
```

```
Out[19]: (10000, 1)
```

10. Split the data into training and testing

```
In [27]: from sklearn.model_selection import train_test_split
```

```
In [28]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

```
In [29]: x_test.shape
```

```
Out[29]: (2000, 15)
```

```
In [30]: x_train.shape
```

```
Out[30]: (8000, 15)
```