ASSIGNMENT- 2

1. Download the dataset:

```
from google.colab import files
uploaded = files.upload()
```

2. Load the dataset:

```
import pandas as pd
df=pd.read_csv("Churn_Modelling.csv")
df
```

3. Perform Below Visualizations:

a) Univariate Analysis:

```
import matplotlib.pyplot as plt
plt.hist(df['CreditScore'], edgecolor = 'black')
plt.title("Credit Score")
```

b) Bi-Variate Analysis:

```
plt.scatter(df.CreditScore, df.Age)
plt.title("CreditScore vs Age")
plt.xlabel('Credit Score')
plt.ylabel('Age')
```

c) Multi-Variate Analysis:

```
import seaborn as sns
sns.barplot(x='Gender', y='Age', data=df, palette='bright', hue='Geography');
```

4. Perform descriptive statistics on the dataset:

```
import numpy as np
df.describe(include=['object'])
df.describe(include=['number'])
```

5. Handle the Missing values:

```
df.isnull()
df.isnull().sum()
```

6. Find the outliers and replace the outliers:

Find Outliers:

```
df1 = df.copy()
df1.drop(columns=['RowNumber','CustomerId','Surname','Geography','Gender','Tenure','NumOfProducts','HasCrCard','IsActiveMember', 'Balance', 'EstimatedSalary', 'Exited'])
```

```
Q1,Q3 = np.quantile(df1.CreditScore, [0.25,0.75])
IQR = Q3 - Q1
ul = Q3 + 1.5 * IQR # upper limit
ll = Q1 - 1.5 * IQR # lower limit
outliers = df1.CreditScore[(df1.CreditScore > ul) | (df1.CreditScore < ll)]
np.array(outliers)
```

Replacing Outliers with median:

```
median = np.quantile(df1.CreditScore, 0.50) # median = 652
df1['CreditScore'] = np.where(df1.CreditScore < ll, 652 , df1['CreditScore'])
df1.describe()
```

7. Check for Categorical columns and perform encoding:

```
geography = pd.get_dummies(df1.Geography, drop_first = False)
geography
df1 = pd.concat([df1, geography], axis=1)
df1.drop('Geography', axis = 1, inplace = True)
df1.head()
```

8. Split the data into dependent and independent variables:

```
df1.drop(['RowNumber','CustomerId','Surname','Gender'], axis = 1, inplace = True)
 # x - Independent , y - Dependent
 x = df1.drop('Exited', axis = 1)
 y = df1['Exited']
 x.head()
 y.head()
```

9. Scale the independent variables:

```
 from sklearn.preprocessing import StandardScaler

 ss = StandardScaler()
 x = ss.fit_transform(x)
 x
```

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 = 0.2, random_state = 0 )
x_train.shape
x_test.shape
y_train.shape
```

```
y_test.shape
```