Assignment_2

November 17, 2022

Assignment Date	24 September 2022
Student Name	Shwetha L S
Student Roll Number	2019504586
Maximum Marks	2 Marks

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

0.0.1 Download and Load dataset

[3]:
```
df = pd.read_csv("https://drive.google.com/uc?
  ↪id=1_HcM0K8wt4b7FMLkc1V1dv0y6I_9ULzy")
df.head()
```

[3]:

	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	

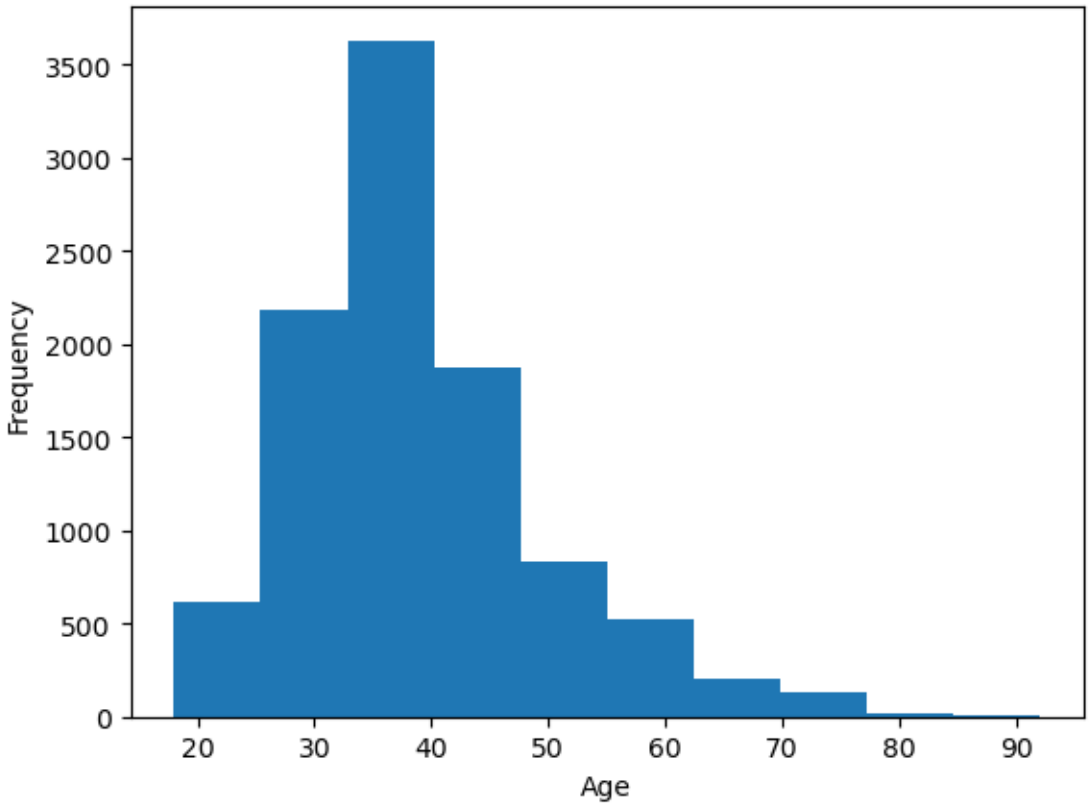
	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1

```
3          93826.63       0
4          79084.10       0
```

0.0.2 Uni-variate Analysis

[4]:
```
plt.hist(df["Age"])
plt.xlabel("Age")
plt.ylabel("Frequency")
```

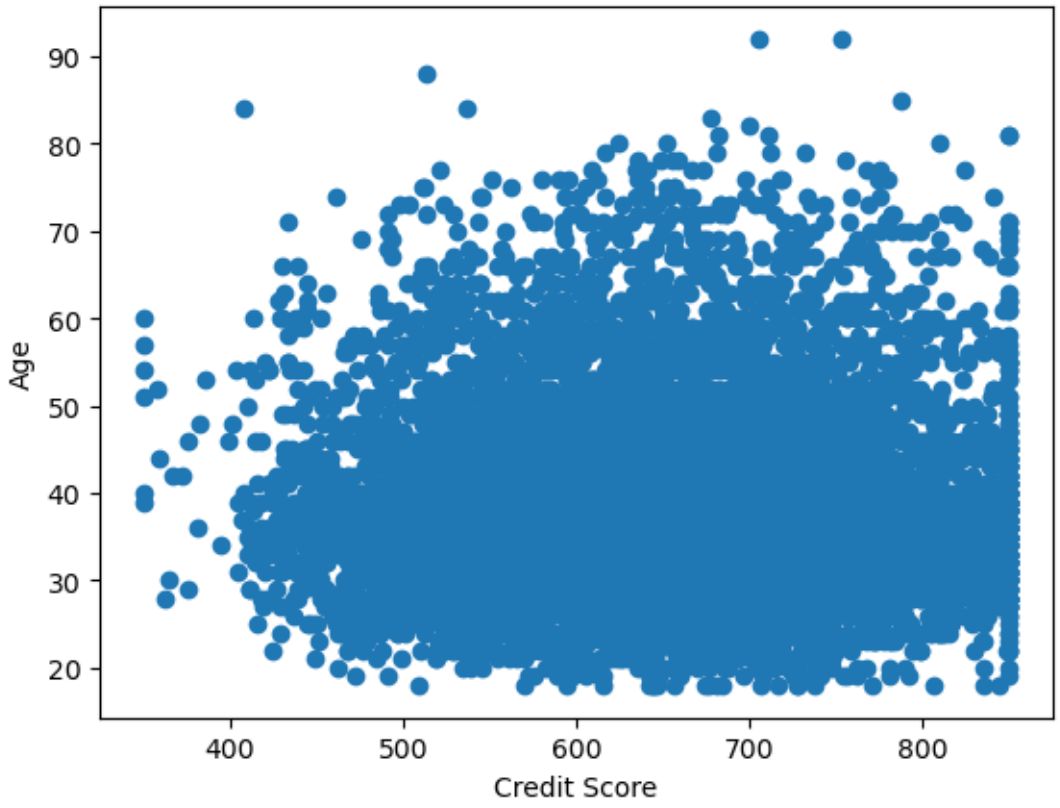
[4]:
```
Text(0, 0.5, 'Frequency')
```



0.0.3 Bi-variate Analysis

[10]:
```
plt.scatter(df["CreditScore"], df["Age"])
plt.xlabel("Credit Score")
plt.ylabel("Age")
```

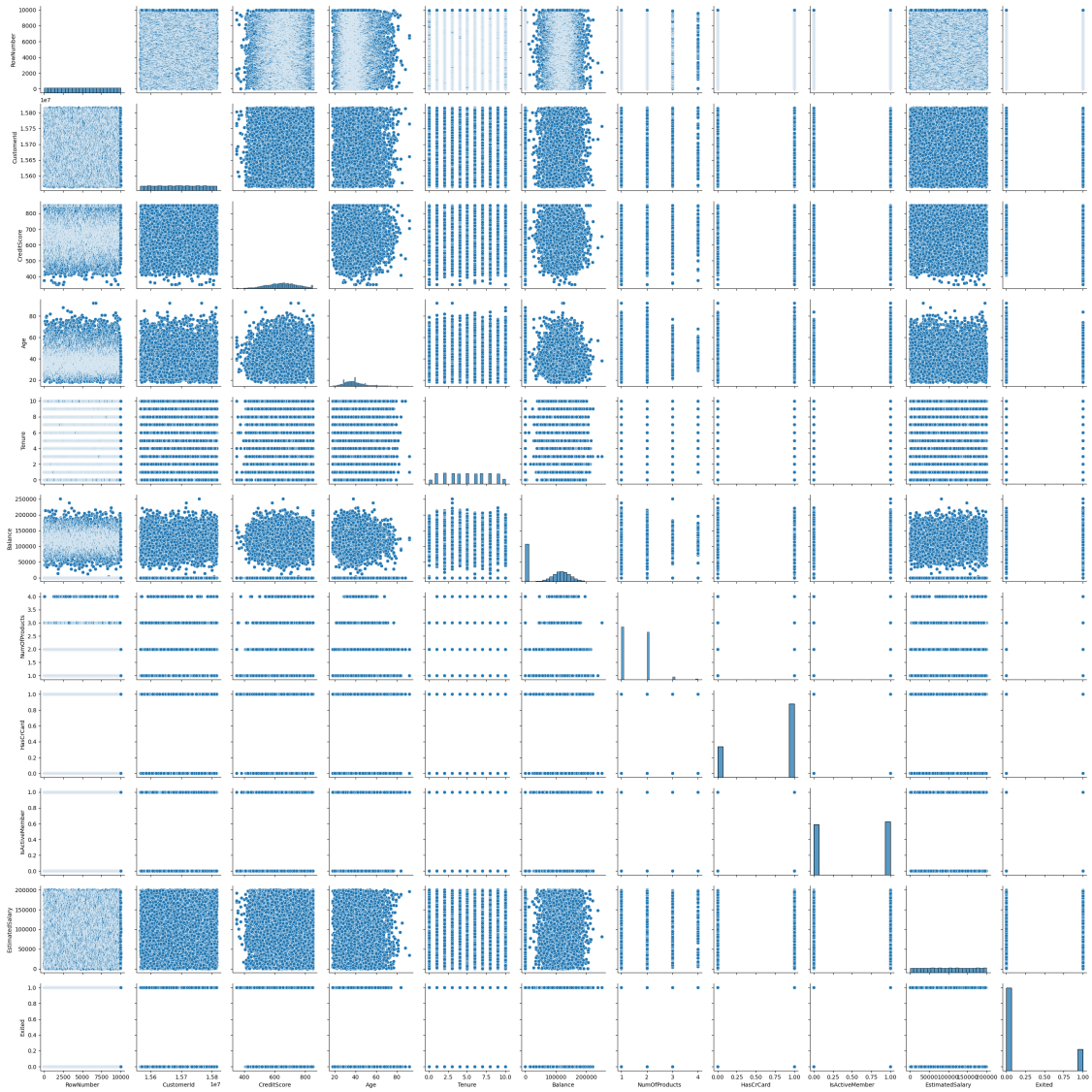
[10]:
```
Text(0, 0.5, 'Age')
```



0.0.4 Multi-variate Analysis

[11]:
```
sns.pairplot(df)
```

[11]:
```
<seaborn.axisgrid.PairGrid at 0x7f72c8580250>
```



0.0.5 Descriptive Statistics

[12]:
```
df.describe()
```

[12]:

	RowNumber	CustomerId	CreditScore	Age	Tenure
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000

```
              Balance  NumOfProducts    HasCrCard  IsActiveMember  \
count    10000.000000   10000.000000  10000.00000    10000.000000
mean     76485.889288       1.530200      0.70550        0.515100
std      62397.405202       0.581654      0.45584        0.499797
min          0.000000       1.000000      0.00000        0.000000
25%          0.000000       1.000000      0.00000        0.000000
50%      97198.540000       1.000000      1.00000        1.000000
75%     127644.240000       2.000000      1.00000        1.000000
max     250898.090000       4.000000      1.00000        1.000000

       EstimatedSalary        Exited
count     10000.000000  10000.000000
mean     100090.239881      0.203700
std       57510.492818      0.402769
min          11.580000      0.000000
25%       51002.110000      0.000000
50%      100193.915000      0.000000
75%      149388.247500      0.000000
max      199992.480000      1.000000
```

0.0.6 Handle missing values

[13]:
```
print("No of missing values:")
df.isnull().sum()
```

```
No of missing values:
```

[13]:
```
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
```

0.0.7 Split Data into independent and dependent variables

[14]:
```
x = df.iloc[:,3:13].values
y = df.iloc[:,13:14].values
print("x shape:", x.shape)
print("y shape:", y.shape)
```

```
x shape: (10000, 10)
y shape: (10000, 1)
```

0.0.8 Categorical columns encoding

[15]:
```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

ct = ColumnTransformer([("oh", OneHotEncoder(), [1,2])],␣
 ↪remainder="passthrough")
x = ct.fit_transform(x)
print("x shape:", x.shape)
```

```
x shape: (10000, 13)
```

0.0.9 Split the data into training and tesing

[16]:
```
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=1)
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: (8000, 13)
y_train shape: (8000, 1)
x_test shape: (2000, 13)
y_test shape: (2000, 1)
```

0.0.10 Scale the independent variables

[18]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```