

## ASSIGNMENT 2

### DATA VISUALIZATION AND PRE-PROCESSING

Assignment Date	24 September 2022
Student Name	Sandhiya Ramalaxmi M
Student Roll Number	2019105568
Maximum Marks	2 Marks

1)The dataset is downloaded.

2)The file dataset is loaded.

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

: d=pd.read_csv("Churn_Modelling.csv")

: d.columns

: Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
        'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
        'IsActiveMember', 'EstimatedSalary', 'Exited'],
        dtype='object')
```

3)Performing Visualization:

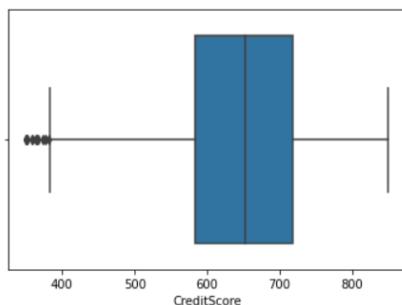
#### **a) Univariate Analysis:**

Univariate analysis

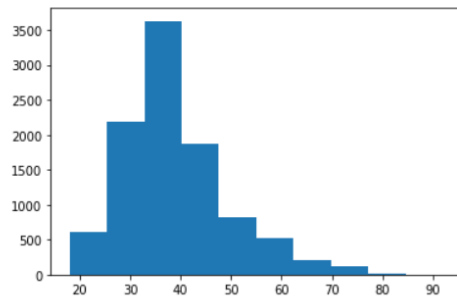
```
: sns.boxplot(d['CreditScore'])

/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 0x7f6e5b962ed0>
```



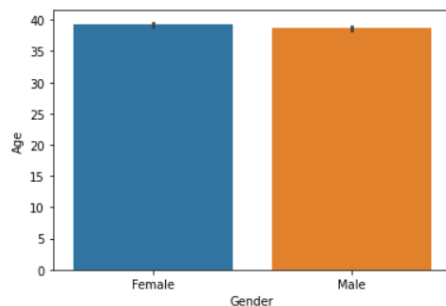
```
: plt.hist(d['Age'])  
plt.show()
```



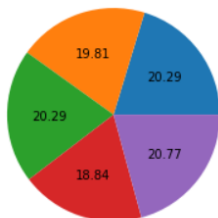
```
sns.barplot(d['Gender'], d['Age'])
```

/usr/local/lib/python3.7/dist-packages/seaborn/\_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. 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 0x7f6e5b836990>



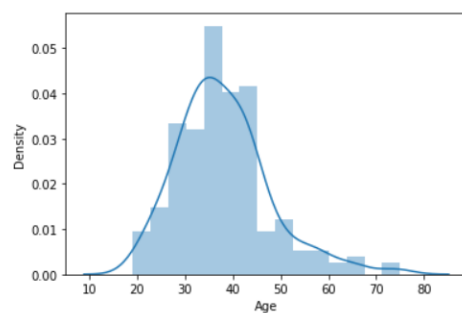
```
pie_chart=plt.pie(d['Age'].head(), autopct="%0.2f")  
plt.show(pie_chart)
```



```
sns.distplot(data['Age'].head(200))
```

/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. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
warnings.warn(msg, FutureWarning)

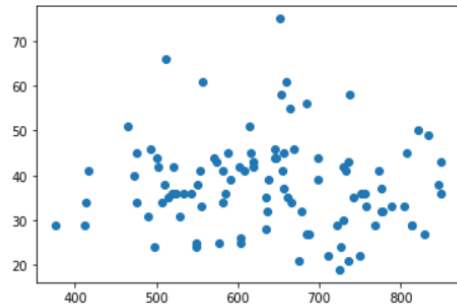
<matplotlib.axes.\_subplots.AxesSubplot at 0x7f6e5b6cd4d0>



## **b) Bi-variate Analysis:**

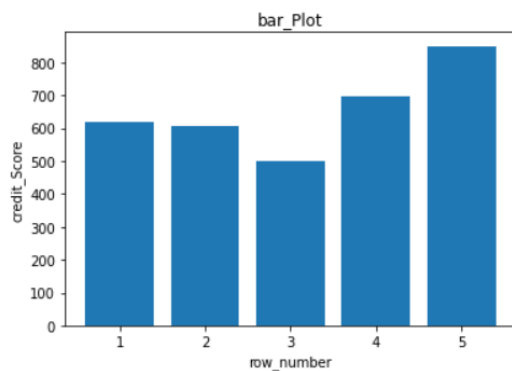
### BIVARIATE ANALYSIS

```
plt.scatter(d['CreditScore'].head(100),d['Age'].head(100))  
plt.show()
```



```
plt.bar(data['RowNumber'].head(),data['CreditScore'].head(), )  
  
plt.title('bar_Plot')  
plt.xlabel('row_number')  
plt.ylabel('credit_Score')
```

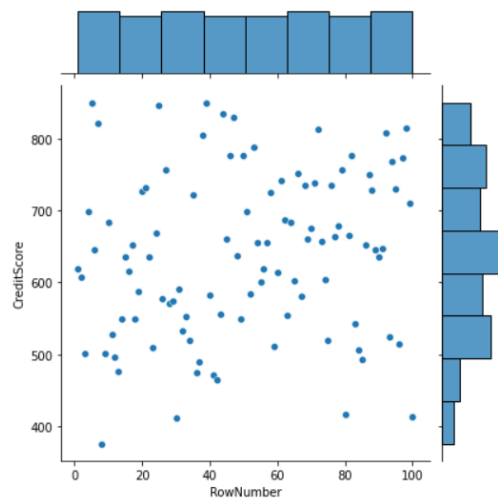
Text(0, 0.5, 'credit\_Score')



```
: sns.jointplot(d['RowNumber'].head(100),d['CreditScore'].head(100), )
```

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

```
: <seaborn.axisgrid.JointGrid at 0x7f6e5b534850>
```



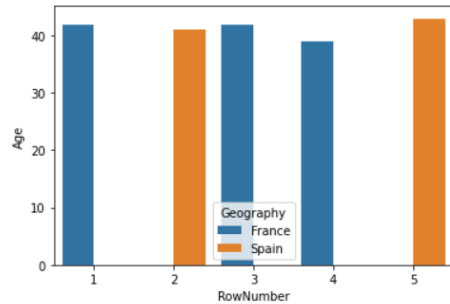
### c) Multi-variate Analysis:

#### MULTIVARIATE ANALYSIS

```
sns.barplot('RowNumber', 'Age', hue='Geography', d=d.head())
```

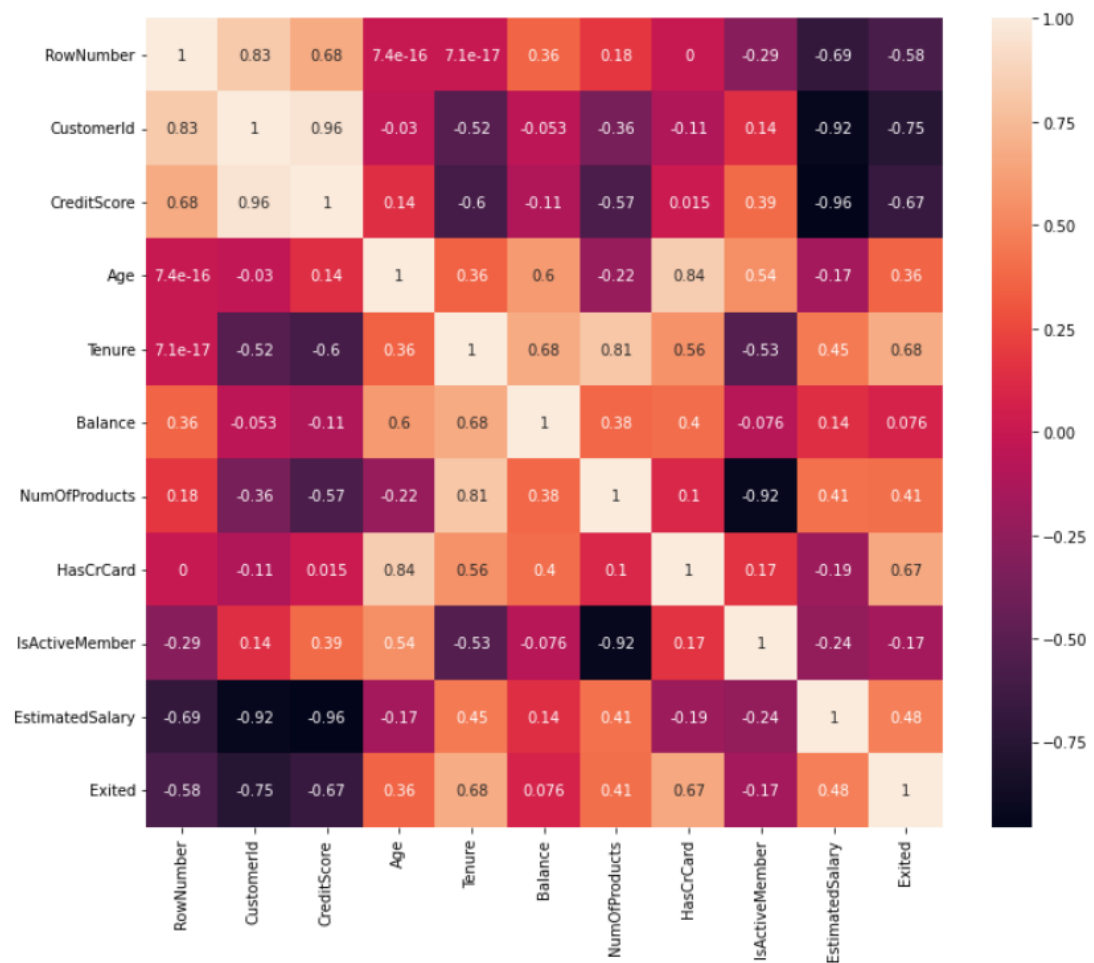
/usr/local/lib/python3.7/dist-packages/seaborn/\_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key word will result in an error or misinterpretation.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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6e58b957d0>
```



```
fig= plt.figure(figsize =(12,10))  
sns.heatmap(d.head().corr(), annot = True)
```

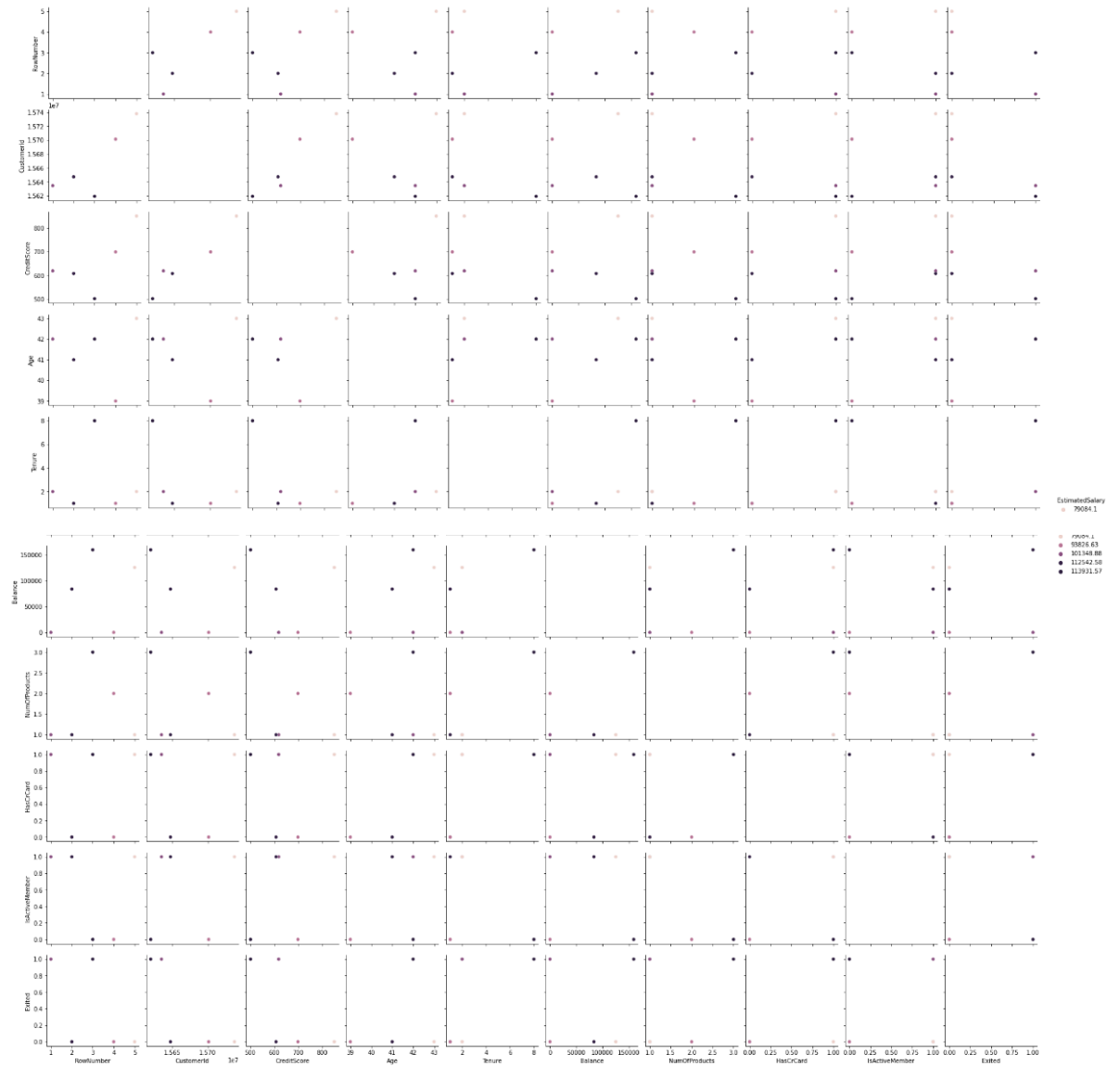
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6e58b1cad0>
```



```
fig= plt.figure(figsize =(7,5))
sns.pairplot(d.head(), hue='EstimatedSalary')
```

<seaborn.axisgrid.PairGrid at 0x7f6e52858090>

<Figure size 504x360 with 0 Axes>



#### 4) Descriptive statistics:

```
: d.head()
```

#	Number	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
2	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
4	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
5	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

```

: d.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

```

```
d.describe()
```

	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	1.589094e+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	0.000000	0.000000	11.580000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.000000	0.000000	51002.110000
50%	5000.50000	1.589074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.000000	1.000000	100193.915000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.000000	1.000000	149388.247500
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.000000	1.000000	199992.480000

## 5) Handling of missing values

Handling of missing values

```
: d.isna().sum()
```

```

: 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

```

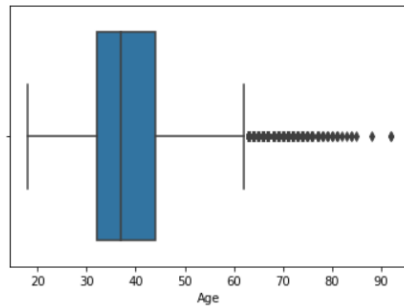
## 6) Checking for outliers and replacing them

Checking for outliers and replacing them

```
sns.boxplot(d['Age'])
```

```
/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 0x7f6e56c45ed0>
```



```
qt= d.quantile(q=[0.25,0.75])  
qt
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.25	2500.75	15628528.25	584.0	32.0	3.0	0.00	1.0	0.0	0.0	51002.1100	0.0
0.75	7500.25	15753233.75	718.0	44.0	7.0	127644.24	2.0	1.0	1.0	149388.2475	0.0

```
irq=qt.loc[0.75]- qt.loc[0.25] # q3 and q1  
irq
```

```
RowNumber      4999.5000  
CustomerId      124705.5000  
CreditScore     134.0000  
Age             12.0000  
Tenure          4.0000  
Balance         127644.2400  
NumOfProducts   1.0000  
HasCrCard       1.0000  
IsActiveMember  1.0000  
EstimatedSalary 98386.1375  
Exited          0.0000  
dtype: float64
```

```
upper= qt.loc[0.75]+(1.5*irq)  
upper
```

```
RowNumber      1.499950e+04  
CustomerId      1.594029e+07  
CreditScore     9.190000e+02  
Age             6.200000e+01  
Tenure          1.300000e+01  
Balance         3.191106e+05  
NumOfProducts   3.500000e+00  
HasCrCard       2.500000e+00  
IsActiveMember  2.500000e+00  
EstimatedSalary 2.969675e+05  
Exited          0.000000e+00  
dtype: float64
```

```
d['Age'].mean()
```

```
38.9218
```

## 7) Categorical data and Encoding:

Categorical data and Encoding

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

d['Gender'].replace({'Female':0, 'Male': 1 }, inplace=True)
d['Geography'].replace({'France':0, 'Germany':1, 'Spain':2}, inplace=True)
d.head()

:
  RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age  Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary
0         1      15634602   Hargrave         619         0         0   42         2         0.00             1             1             1         101348.88
1         2      15647311     Hill         608         2         0   41         1      83807.86             1             0             1         112542.58
2         3      15619304     Onio         502         0         0   42         8     159660.80             3             1             0         113931.57
3         4      15701354     Boni         699         0         0   39         1          0.00             2             0             0          93826.63
4         5      15737888   Mitchell         850         2         0   43         2     125510.82             1             1             1          79084.10
```

```
# using dummy values
data_d= pd.get_dummies(d,columns = ['Surname'])
data_d.head()
```

RowNumber	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	...	Surname_Zinachukwudi	Surname_Zito
0	1	15634602	619	0	0	42	2	0.00	1	1 ...	0	0
1	2	15647311	608	2	0	41	1	83807.86	1	0 ...	0	0
2	3	15619304	502	0	0	42	8	159660.80	3	1 ...	0	0
3	4	15701354	699	0	0	39	1	0.00	2	0 ...	0	0
4	5	15737888	850	2	0	43	2	125510.82	1	1 ...	0	0

5 rows × 2945 columns

## 8) Splitting the data into dependent and independent variables

Splitting the data into dependent and independent variables

```
x=data_d.drop(columns= ['EstimatedSalary']).values
y=data_d['EstimatedSalary'].values
print(x)
print(y)

[[1.0000000e+00 1.5634602e+07 6.1900000e+02 ... 0.0000000e+00
 0.0000000e+00 0.0000000e+00]
 [2.0000000e+00 1.5647311e+07 6.0800000e+02 ... 0.0000000e+00
 0.0000000e+00 0.0000000e+00]
 [3.0000000e+00 1.5619304e+07 5.0200000e+02 ... 0.0000000e+00
 0.0000000e+00 0.0000000e+00]
 ...
 [9.9980000e+03 1.5584532e+07 7.0900000e+02 ... 0.0000000e+00
 0.0000000e+00 0.0000000e+00]
 [9.9990000e+03 1.5682355e+07 7.7200000e+02 ... 0.0000000e+00
 0.0000000e+00 0.0000000e+00]
 [1.0000000e+04 1.5628319e+07 7.9200000e+02 ... 0.0000000e+00
 0.0000000e+00 0.0000000e+00]]
[101348.88 112542.58 113931.57 ... 42085.58 92888.52 38190.78]
```



## 9) Scaling the independent variables

Scaling the independent variables

```
: from sklearn.preprocessing import scale

: x = scale(x)
: x
array([[ -1.73187761, -0.78321342, -0.32622142, ..., -0.01000005 ,
        -0.01414355, -0.01414355],
       [ -1.7315312 , -0.60653412, -0.44003595, ..., -0.01000005 ,
        -0.01414355, -0.01414355],
       [ -1.73118479, -0.99588476, -1.53679418, ..., -0.01000005 ,
        -0.01414355, -0.01414355],
       ...,
       [  1.73118479, -1.47928179,  0.60498839, ..., -0.01000005 ,
        -0.01414355, -0.01414355],
       [  1.7315312 , -0.11935577,  1.25683526, ..., -0.01000005 ,
        -0.01414355, -0.01414355],
       [  1.73187761, -0.87055909,  1.46377078, ..., -0.01000005 ,
        -0.01414355, -0.01414355]])
```

## 10) Splitting the data into training and testing

Splitting 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)

: print(x_train.shape, x_test.shape)
(8000, 2944) (2000, 2944)
```