



3	4	157013540	Boni	6990	France	maFe	39	1	0.00	2	0			
					le									
			Mit		Fe		125							
4	5	157378881	chell	8500	Spain	male	43	2	510.82	1	1			
...	...	...	...	...	...	...	..	...	...	...	...			
99	9996	1560	Obi		Fran	Ma	3	5	0.00	2	1	0	96270.	0
95		6229	jiak u	771	ce	le	9						64	

	Row Num ber	Cust omer Id	Sur na me	Cred itSco re	Geo grap hy	Ge nd er	A g e	Te nu re	Bal anc e	NumO fProdu cts	Has CrC ard	IsActiv eMem ber	Estima tedSal ary	Ex ite d
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabattini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 14 columns

In [3]:

```
df.head()
```

Out[3]:

	Row Num ber	Cust omer Id	Sur na me	Cred itSco re	Geo grap hy	Ge nd er	A nd g	Te nu re	Bal anc e	NumO fProdu cts	Has CrC ard	IsActiv eMem ber	Estima tedSala ite ry	Ex
0		1	46021563	1		Harvegra		619		France	male	Fe 24	2	0.00
			1	1	1		101348.88			1				
1		2	15647311	0		Hill	608	Spai n		male	41	1	07.86	1
			0	1		112542.58		0						

```

2          15619304  Onio  502  France  maleFe  24  8  660.15980  3
          1  0  113931.57  1

3

3          15701354  Boni  France  lemaFe  39  1  0.00  2  0  0  93826.63
          0

4          699

      Row  Cust  Sur  Cred  Geo Ge  A Te Bal NumO Has IsActiv Estima Ex g nu anc fProdu
Num omer na itSco grap nd ber Id me re hy er  CrC eMemb tedSala ite e re e cts ard er ry d
4      5 78881573  chelMitl  850  Spain  leFe  3  510.12582  79084.10  0
          ma  4  2  1  1  1

```

In [4]: df.shape

(10000, 14)

Out[4]:

## 3.Univariate,Bivariate & MultiVariate Analysis

### Univariate Analysis

```

df_france=df.loc[df['Geography']=='France']
df_spain=df.loc[df['Geography']=='Spain']
df_germany=df.loc[df['Geography']=='Germany']

```

In [9]:

In [17]:

```

plt.plot(df_france['Balance'],np.zeros_like(df_france['Balance']),'o')
plt.plot(df_spain['Balance'],np.zeros_like(df_spain['Balance']),'o')
plt.plot(df_germany['Balance'],np.zeros_like(df_germany['Balance']),'o')
plt.xlabel('Age') plt.show()

```

### Bivariate Analysis

In [18]:

```

sns.FacetGrid(df,hue="Geography",size=5).map(plt.scatter,"Age","Balance")
.a dd_legend(); plt.show()

/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:337: UserWarning
: The `size` parameter has been renamed to `height`; please update your code.

```

```
warnings.warn(msg, UserWarning)
```

Multivariate Analysis

```
In [24]: sns.pairplot(df,hue="Gender",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)
```

Out[24]:

```
<seaborn.axisgrid.PairGrid at 0x7f9a9f3029d0>
```

4.Descriptive Statistics

```
df.head()
```

In [29]:

Out[29]:

	Row Num	Customer Number	Subscription	Credit Score	Geography	Gender	Age	Telemarketing	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Expenditure
0	1	156346021	1	602	Hargrave	female	6191	1	France	male	42	2	0.00	1
1	2	15647311.58	Hill0	608	Spain	female	4	838	1	1	07.86	1	0	112542
2	3	930415611	0	1561	Onion	female	5021	1	France	male	42	8	660.15980	3
3	4	157013540		1354	Boni	female	6990	0	France	male	39	1	0.00	20

```

4      5      15737888      chelMiti      850      Spain      maFele 43      2      510.12582
      1      1      1      79084.10      0

```

In [30]: `df.mean()` # Get the mean of each column

```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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.

```

```

"""Entry point for launching an IPython kernel.

```

Out[30]:

```

RowNumber      5.000500e+03
CustomerId      1.569094e+07
CreditScore     6.505288e+02
Age             3.892180e+01
Tenure          5.012800e+00
Balance         7.648589e+04
NumOfProducts  1.530200e+00
HasCrCard       7.055000e-01
IsActiveMember  5.151000e-01
EstimatedSalary1.000902e+05
Exited          2.037000e-01

```

```

dtype: float64
In [31]:
df.mean(axis=1) # Get the mean of each row

```

```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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.

```

```

"""Entry point for launching an IPython kernel.

```

Out[31]:

```

0      1.430602e+06
1      1.440392e+06
2      1.444860e+06

```

```

3      1.435993e+06
4      1.449399e+06

```

...

```

9995    1.428483e+06
9996    1.430866e+06
9997    1.421579e+06
9998    1.441922e+06
9999    1.437044e+06

```

```

Length: 10000, dtype: float64

```

`df.median()` # Get the median of each column In [32]:

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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. """Entry point for launching an IPython kernel.
```

```
RowNumber      5.000500e+03 1.569074e+07
CustomerId      6.520000e+02 3.700000e+01
CreditScore    5.000000e+00
Age            9.719854e+04
Tenure         1.000000e+00 1.000000e+00
Balance        1.000000e+00
NumOfProducts 1.001939e+05
HasCrCard      0.000000e+00
IsActiveMember
EstimatedSalary
Exited dtype:
float64
```

Out[32]:

In [39]:

```
norm_data = pd.DataFrame(np.random.normal(size=100000))
norm_data.plot(kind="density",
                figsize=(10,10)); plt.vlines(norm_data.mean(), # Plot black line at
mean
```

```
        ymin=0, ymax=0.4,
        linewidth=5.0);
plt.vlines(norm_data.median(), # Plot red line at median
        ymin=0,          ymax=0.4,          linewidth=2.0,
        color="red");
```

In [36]:

```
skewed_data = pd.DataFrame(np.random.exponential(size=100000))

skewed_data.plot(kind="density",
                  figsize=(10,10), xlim=(-1,5));
```

```
plt.vlines(skewed_data.mean(), # Plot black line at mean
        ymin=0, ymax=0.8,
        linewidth=5.0);
```

```
plt.vlines(skewed_data.median(), # Plot red line at median
        ymin=0,          ymax=0.8,          linewidth=2.0,
        color="red");
```

```
In [40]: norm_data = np.random.normal(size=50) outliers = np.random.normal(15,
size=3)
```

```
combined_data = pd.DataFrame(np.concatenate((norm_data, outliers), axis=0))
combined_data.plot(kind="density",
                  figsize=(10,10), xlim=(-5,20));
```

```
plt.vlines(combined_data.mean(),                                # Plot black line at mean
           ymin=0,
           ymax=0.2, linewidth=5.0);
```

```
plt.vlines(combined_data.median(),                             # Plot red line at median
           ymin=0, ymax=0.2,
           linewidth=2.0,
           color="red");
```

```
df.mode()
```

In [42]:

Out[42]:

	Row Number	Customer Id	Sur name	Cred grap re	Geo nd hy	Ge itSco er	A Te Bal g nu anc e re e	NumO fProducts	Has CrCard	IsActiv eMember	Estima tedSalary	Ex ite ry
0												
		15565701	Smith		France	7.30	2.00.0			1.0	1.0	24924.920.0
	1			850.0				1.0				
1		15565706	NaN		NaN	NaN	NaN	NaN	NaN	NaN		NaN
	2			NaN		NaN	NaN	NaN			NaN	
2		15565714	NaN		NaN	NaN	NaN	NaN	NaN	NaN		NaN
	3			NaN				NaN			NaN	
3		15565779	NaN		NaN	NaN	NaN	NaN	NaN	NaN		NaN
	4			NaN				NaN			NaN	
4		15565796	NaN		NaN	NaN	NaN	NaN	NaN	NaN		NaN
	5			NaN		NaN	NaN	NaN			NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...
9996		15815628	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN		NaN
9995						NaN	NaN	NaN			NaN	
9996		15815645	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN		NaN
9996						NaN	NaN	NaN			NaN	



[illegible]

10000 rows × 14 columns

## Measures of Spread

In [43]:

```
max(df["Age"]) - min(df["Age"])
```

Out[43]:

74

In [45]:

```
five_num = [df["Age"].quantile(0),  
            df["Age"].quantile(0.25),  
            df["Age"].quantile(0.50),  
            df["Age"].quantile(0.75),  
            df["Age"].quantile(1)]
```

five\_num

```
[18.0, 32.0, 37.0, 44.0, 92.0]
```

Out[45]:

```
df["Age"].describe()
```

In [46]:

```
count 10000.000000 mean
```

```
      38.921800 std
```

```
      10.487806 min
```

```
      18.000000 25%
```

```
      32.000000
```

```
50%  37.000000 75%
```

```
      44.000000 max
```

```
      92.000000
```

```
Name: Age, dtype: float64
```

Out[46]:

```
df["Age"].quantile(0.75) - df["Age"].quantile(0.25)
```

12.0

Out[47]:

In [49]:

```
df.boxplot(column="Age",  
           return_type='axes',  
           figsize=(8,8))  
plt.text(x=0.74, y=22.25, s="3rd Quartile")  
plt.text(x=0.8, y=18.75, s="Median")  
plt.text(x=0.75, y=15.5, s="1st Quartile")  
plt.text(x=0.9, y=10, s="Min")  
plt.text(x=0.9, y=33.5, s="Max")  
plt.text(x=0.7, y=19.5, s="IQR", rotation=90, size=25);
```

In [50]:

```
df["Age"].var()
```

Out[50]:

```
109.99408416841683
```

In [51]:

```
df["Age"].std()
```

```
10.487806451704609
```

Out[51]:

```
abs_median_devs = abs(df["Age"] - df["Age"].median())
abs_median_devs.median() * 1.4826
```

In [52]:

```
8.8956
```

Out[52]:

## Skewness and Kurtosis

In [53]:

```
df["Age"].skew() # Check skewness
```

Out[53]:

```
1.0113202630234552
```

In [54]:

```
df["Age"].kurt() # Check kurtosis
```

Out[54]:

```
1.3953470615086956
```

In [55]:

```
norm_data = np.random.normal(size=100000)
skewed_data = np.concatenate((np.random.normal(size=35000)+2,
                               np.random.exponential(size=65000)), axis=0)
uniform_data = np.random.uniform(0,2, size=100000)
peaked_data = np.concatenate((np.random.exponential(size=50000),
                               np.random.exponential(size=50000)*(-
                               1)), axis=0)
```

```
data_df = pd.DataFrame({"norm":norm_data,
                        "skewed":skewed_data,
                        "uniform":uniform_data,
                        "peaked":peaked_data})
```

In [56]:

```
data_df.plot(kind="density",
             figsize=(10,10), xlim=(-5,5));
```

```
data_df.skew()
```

In [57]:

```
norm      -0.007037
skewed     1.002549
uniform -0.004434 peaked
      0.018058 dtype:
float64 data_df.kurt()
```

Out[57]:

```
norm      -0.009914      skewed
1.314497
```

In [58]:

Out[58]:

```
uniform -1.201740
peaked    2.971592
```

```
dtype: float64
```

## 5.Handle the Missing

### values

In [83]:

```
df=pd.read_csv('/content/Churn_Modelling.csv')
df.head()
```

In [84]:

Out[84]:

	Row Estima	Cust Ex	Sur	Cred	Geo	Ge	A	Te	Bal	NumO	Has	IsActiv
	Num tedSala	omer ite	na	itSco	grap	nd	g	nu	anc	fProdu	CrC	eMemb
	ber ry	Id d	me	re	hy	er	e	re	e	cts	ard	er
Har												
0	1	15634602			grave	619		France	maleFe	42	2	
	0.00	1	1	1	101348.88			1				
		1564			Spai	Fe	4		838			112542
1	2	7311	Hill	608	n	male	1	1	07.86	1	0	
	1	.58	0									
		1561	Onio	502	Fran	maFe	4	8	660.159	3	1	0
		113931	1									
2	3	9304	ce	le	2	80	.57					
				699	France	Fe	3	1	0.00	2	0	0
				93826.	0							

<b>3</b>	4	15701354	Boni	male	9	63
----------	---	----------	------	------	---	----

1573	chelMit	850	Spai	maFe	4	2	510.125	1	1	1
79084.	0									

4 5 7888 1 n le 3 82 10

In [86]:

```
df.isnull()
```

Out[86]:

Estima	Row Ex	Cust	Sur	Cred	Geo	Ge	A	Te	Bal	NumO	Has	IsActiv	
	Num tedSala	omer ite	na	itSco	grap	nd	ge	nu	anc	fProdu	CrC	eMem	
	ber ry	Id d	me	re	hy	er		re	e	cts	ard	ber	
0			False			False	F	False	False			False	
	False	False False	e False	False False	False lse	se		al		se	se	False	
						se							
	Row A	Cust	Sur	Cred	Geo	Ge	Te	Bal	NumO	Has	IsActiv	CrC	
	Num omer	na itSco	grap nd	ge ber	Id me	re hy	er	re	e	cts	ard	ber	ry d
1	False	False False	False lse	False	False	False	False	seal	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False
			False			False	F	False	False				False

3	False	False False	e lse	False	False	se	seal	se	se	False	False	False		
4	False	False False	Fals e lseFa	False	False	False	sealF	False	False	False	False	False		
...	...	...	...	...	...	...	...	...	...	...	...	...		
9 <sup>9</sup>	False	False	Fals	False	F	Fal	Fal	False	False	False	False	Fa	Fal	al
9			e				se	se	se	se				lse
5														
9 <sup>9</sup>	False	False	Fals	False	False	False	alF	False	False	False	False	False	False	lseFa
9			e				se							
6														
99	False	Fals	False	Fal	F	Fal	Fal	False	False	False	False	Fa		
9		False	e	False	al	se	se	se	se					lse
7														
9 <sup>9</sup>	False	False				Fal	F	Fal	Fal					Fa
9			Fals	False	False	se	seal	se	se	False	False	False	False	lse
8			e											
9 <sup>9</sup>	False	False				Fals	False	False	Fal	F	Fal	Fal	False	
99			e			se	se	se	se	False	Fa	al		lse

10000 rows × 14 columns

```
In [89]: sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[89]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a987d8290>
```

In [93]:

```
sns.set_style('whitegrid')
sns.countplot(x='Geography',data=df)
```

Out[93]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a92a88850>
```

```
sns.set_style('whitegrid')
sns.countplot(x='Geography',hue='Gender',data=df,palette='RdBu_r')
```

In [94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a92ec10d0>
```

Out[94]:

```
sns.set_style('whitegrid')
sns.countplot(x='Geography',hue='Gender',data=df,palette='rainbow')
```

In [96]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a92afac50>
```

```
sns.distplot(df['Age'].dropna(),kde=False,color='darkred',bins=40)
```

Out[96]:

In [97]:

```
/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)
```

Out[97]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a98787590>
```

In [98]:

```
df['Age'].hist(bins=30,color='darkred',alpha=0.3)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a92d64c10>
```

Out[98]:

```
sns.countplot(x='NumOfProducts',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a9306f790>
```

In [100]:

```
df['Age'].hist(color='green',bins=40,figsize=(8,4))
```

Out[100]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a90f52d90>
```

## Cufflinks for plots

In [101]:

Out[101]:

```
import cufflinks as cf
cf.go_offline()
```

In [102]:

```
df['Age'].plot(kind='hist',bins=30,color='green')
```

In [ ]:

# Data Cleaning

```
plt.figure(figsize=(12, 7))
sns.boxplot(x='Gender',y='Age',data=df,palette='winter')
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a90f59450>
```

In [107]:

Out[107]:

```
def impute_age(cols):
    Age = cols[0]
    Pclass = cols[1]

    if pd.isnull(Age):

        if Pclass == 1:
            return 37

        elif Pclass == 2: return
            29

        else:
            return 24
    else:
        return Age
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

In [307]:

Out[112]:

In [112]:

Out[122]:

In [122]:

Out[114]:

In [114]:

Out[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a8aa699d0>
```

```
df.drop('Gender',axis=1,inplace=True)
```

Out[114]:

```
df.head()
```

RowN	Custo	Sur	Credi	Geog	Age	Tenure	Balance	ProductNumOfs	Estimat	Ex
HasCrCard	e	IsActive	Member	umbe	merI	nam	tScor	raph r d	edSalar	ite y
e	e	y							d	d



0	1	15634602	grave	619	France	42	2	0.00	1	1	1
		101348.88	1								

RowN	Custo	Sur	Credi	Geog	Age	Tenure	Balance	Product	NumOfs	HasCrCard	IsActive	Member
EstimatedSalary	Exited	umbe	merI	nam	tScor	raph	r	d	e	e	y	
1	2	15647311		Hill	608	Spain	41	1	83807.86	1	0	
		1	112542.58		0							
15619		Oni	Franc	4	1596	113931.2	3	304	o	502	e	
2	8	60.80	3	1	0	57	1					
3	4	15701354		Boni	699	France	39	1	0.00	2	0	0
		93826.63		0								
4	5	15737888		Mitchell	850	Spain	43	2	125510.82	1	1	
		1	79084.10		0							

## Converting Categorical Features

In [116]:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999 Data
columns (total 13 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  Age            10000 non-null int64
6  Tenure         10000 non-null int64
7  Balance        10000 non-null float64  8 NumOfProducts    10000 non-null
   int64            9 HasCrCard         10000 non-null int64
10 IsActiveMember 10000 non-null int64
```

```

11 EstimatedSalary 10000 non-null float64
12 Exited 10000 non-null int64 dtypes: float64(2), int64(9), object(2)
memory usage: 1015.8+ KB

```

```
In [118]: pd.get_dummies(df['Geography'], drop_first=True).head()
```

Out[118]:

```

      Germany  Spain
0          0    0
      Germany  Spain
1          0    1
2          0    0
3          0    0
4          0    1

```

```
In [124]: df.info
```

Out[124]:

```

<bound method DataFrame.info of RowNumber CustomerId Surname Cre ditScore
Geography Age Tenure \
0          1 15634602 Hargrave  619 France 42      2
1          2 15647311   Hill   608   Spain 41      1 2    3 15619304  Onio
          502 France 42      8
3 4 15701354 Boni 699 France 39 1 4 5 15737888 Mitchell 850 Spain 43 2
... ..
9995          9996 15606229 Obijiaku 771 France 39      5
9996          9997 15569892 Johnstone      516 France 35      10
9997          9998 15584532   Liu   709 France 36      7
9998          9999 15682355 Sabbatini      772 Germany 42      3
9999          10000 15628319   Walker      792 France 28      4

      Balance NumOfProducts HasCrCard IsActiveMember EstimatedSalary
\
0          0.00 1      1      1      101348.88
1          83807.86 1      0      1      112542.58
2          159660.80 3 1 0 113931.57 3 0.00 2 0 0 93826.63 4 125510.82 1 1
          1 79084.10 ... ..
9996          57369.61 1      1      1      101699.77
9997          0.00 1      0      1      42085.58
9998          75075.31 2      1      0      92888.52
9999          130142.79 1      1      0      38190.78

      Exited
0          1
1          0

```

```

2          1
3          0
4          0
...      ...
9995       0
9996       0
9997       1
9998       1
9999       0

```

```
[10000 rows x 13 columns]>
```

In [125]:

```

sex = pd.get_dummies(df['Age'],drop_first=True)
embark =
pd.get_dummies(df['Balance'],drop_first=True)

```

In [127]:

```
df.drop(['Age','HasCrCard','Surname','CustomerId'],axis=1,inplace=True)
```

In [129]:

```
df.head()
```

Out[129]:

	RowNum	CreditSc	Geogra	Tenu	Balanc	NumOfProd	IsActiveMe	EstimatedSa	Exit
	ber	ore	phy	re	e	ucts	mber	lary	ed
0	1	619	France	2	0.00	1	1	101348.88	1
1	2	608	Spain	1	83807.86	1	1	112542.58	0
					159660.				
2	3	502	France	8	80	3	0	113931.57	1
3	4	699	France	1	0.00	2	0	93826.63	0
4	5	850	Spain	2	125510.82	1	1	79084.10	0

In [130]:

```
train = pd.concat([df,sex,embark],axis=1)
```

```
train.head()
```

In [131]:

Out[131]:

2 2 2 2 2 2 2 2 2 2

Ro

w Cr G T N B Nu IsA Est E 1 1 1 1 1 1 2 22 38  
50

ed eo n m e u itS co al na mOoduffPr ctiveMem imaSalted itxe 1 . 26 26 72 13 34  
61 1 7.26 7.83 8.89

gr

ap u ce cts ber ary d 9 ' 9 9 7 4 4 0 5 6 5 0

be re hy r

.  
r e 2. 6. 8. 6. 6. 9. 3  
9 3 2 2 9 8 2.  
7 2 6 8 8 3 6 9

Fr 0. 101 .  
an 2 0 1 1 348 1 0 . 0 0 0 0 0 0 0 0 0  
61 ce 0 .88 .  
0 1 9

Sp 8 112 .  
ai 1 3 1 1 542 0 0 . 0 0 0 0 0 0 0 0 0

1 620  
8

Ro 1 1 1<sup>2</sup> 1<sup>2</sup> 41  
16 22 222 238 250 2  
2 2 2

w Cr G T B Nu IsA Est E  
N ed eo al mOfPr ctiveM imated itx ' 629 692 27<sup>7</sup> 13<sup>4</sup> 6.43 9.10 15 7.62 7.83 8.89 u itS gr<sup>e</sup> a  
m co ap n odu em Sal e 1<sub>9</sub> . 2. 6. 8.  
6. 9 8 6 5 0 be re hy n ce cts ber  
ary d 9 3 2 2 3 r u .

r 2. e

7 2 6 8 8 3 6 9

7.  
8  
6

113 .  
3 0 931 1 0 . 0 0 0 0 0 0 0 0 0 0  
1 .57 .  
5

Fr 9

2 50 6  
3 an 8 938 .  
2 ce 6 2 0 26. 0 0 . 0 0 0 0 0 0 0 0 0  
63 .  
0.  
8  
0

790 .  
1 1 84.10 0 0 .. 0 0 0 0 0 0 0 0 0 0 69 Fr 0.  
3 4 9 an 1 0 ce 0

1  
2  
Sp 5  
4 5 85 ai 2 5  
0 n 1  
0.

8

2

5 rows  $\times$  6459 columns

## 6. Find the outliers and replace the outliers

In [147]:

```
dataset= [11,10,12,14,12,15,14,13,15,102,12,14,17,19,107,  
10,13,12,14,12,108,12,11,14,13,15,10,15,12,10,14,13,15,10]
```

### Detecting outlier using Z score

#### Using Z score

In [148]:

```
outliers=[] def detect_outliers(data):  
threshold=3 mean =  
    np.mean(data)
```

```
std =np.std(data)
```

```
for i in data:
```

```
    z_score=(i - mean)/std
```

```
    if np.abs(z_score) > threshold:
```

```
        outliers.append(y)
```

```
return outliers
```

In [151]:

```
outlier_pt=detect_outliers(dataset)
```

In [152]:

```
outlier_pt
```

Out[152]:

```
[0      101348.88

1      112542.58
2      113931.57    3  93826.63
4      79084.10
...
9995  96270.64    9996
101699.77        9997
42085.58

9998      92888.52

9999      38190.78
Name: EstimatedSalary, Length: 10000, dtype: float64, 0      101348.88
1      112542.58
2      113931.57    3  93826.63    4  79084.10
...
9995  96270.64    9996
101699.77        9997
42085.58

9998      92888.52

9999      38190.78
Name: EstimatedSalary, Length: 10000, dtype: float64, 0      101348.88
1  112542.58    2
113931.57      3
93826.63
4      79084.10
...
9995  96270.64    9996
101699.77        9997
42085.58

9998      92888.52

9999      38190.78
Name: EstimatedSalary, Length: 10000, dtype: float64]
```

In [153]:

```
## Perform all the steps of IQR
```



```
sorted(dataset)
```

Out[153]:

```
[10,  
 10,  
 10,  
 10,  
 10,  
 11,  
 11,  
 12,  
 12,  
 12,  
 12,  
 12,  
 12,  
 12,  
 12,  
 13,  
 13,  
 13,  
 13,  
 13,  
 14,  
 14,  
 14,  
 14,  
 14,  
 14,  
 15,  
 15,  
 15,  
 15,  
 15,  
 17,  
 19,  
 102,  
 107,  
 108]
```

In  
[155]:

```
quantile1, quantile3= np.percentile(dataset, [25,75])  
print(quantile1,quantile3)
```

In  
[156]:

```
12.0 15.0
```

In  
[157]:

```
## Find the IQR
```

```
iqr_value=quantile3-quantile1  
print(iqr_value)
```

```
3.0
```

```
## Find the lower bound value and the higher bound value
```

In  
[159]:

```
lower_bound_val = quantile1 -(1.5 * iqr_value)  
upper_bound_val = quantile3 +(1.5 * iqr_value)  
print(lower_bound_val,upper_bound_val)
```

In  
[160]:

```
7.5 19.5
```

## 7.Check for Categorical columns and perform encoding

```
df=pd.read_csv('/content/Churn_Modelling.csv')
```

In [161]:

```
df.head()
```

In [162]:

Out[162]:

Row Number	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602		Hargrave	Male	41	1	101348.88	4	2	0.00	1	1
1	2	15647311	Hill	Spain	Female	44		838	1	07.86	1	0	1
2	3	15619304	Onion	France	Female	44		159	8	660.80	3	1	0
3	4	13541570		Bonito	Female	69	0	93826.63	93	1	0.00	2	0
4	5	78881573	Mitchell	Spain	Female	43		12582			79084.10		0

In [163]:

```
df_numeric = df[['RowNumber', 'CustomerId', 'CreditScore', 'Age', 'Tenure', 'Balance',
```

```
'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary', 'Exited']]
```

```
df_categorical = df[['Surname', 'Geography', 'Gender']]
```

```
In [164]: df_numeric.head()
```

```
Out[164]:
```

```
RowNu Custo Credit A Ten Balan NumOfPr HasCr IsActiveM Estimated Exi mber merId Score ge ure ce oducts Card
ember Salary ted
```

```
0 1 156346 02 619 42 2 0.00 1 1 1 101348.88 1
```

```
RowNu Custo Credit A Ten Balan NumOfPr HasCr IsActiveM Estimated Exi mber merId Score ge ure ce oducts
Card ember Salary ted
```

```
1 2 156473 11 608 41 1 83807 .86 1 0 1 112542.58 0
```

```
2 3 156193 04 502 42 8 15966 0.80 3 1 0
113931.57 1
```

```
3 4 157013 54 699 39 1 0.00 2 0 0 93826.63 0
```

```
4 5 157378 88 850 43 2 12551 0.82 1 1 1
79084.10 0
```

```
df_categorical.head()
```

```
In [165]:
```

```
Out[165]:
```

```
Surname Geography Gender
```

```
0 Hargrave France Female
```

```
1 Hill Spain Female
```

```
2 Onio France Female
```

```
3 Boni France Female
```

```
4 Mitchell Spain Female
```

```
In [166]:
```

```
In [167]:
```

```
print(df['Surname'].unique())
print(df['Geography'].unique())
print(df['Gender'].unique())
```

```
['Hargrave' 'Hill' 'Onio' ... 'Kashiwagi' 'Aldridge' 'Burbidge']
['France' 'Spain' 'Germany']
['Female' 'Male']
```

In [168]:

```
from sklearn.preprocessing import LabelEncoder marry_encoder
= LabelEncoder()

marry_encoder.fit(df_categorical['Gender'])

LabelEncoder()
```

Out[168]:

In [169]:

```
marry_values = marry_encoder.transform(df_categorical['Gender'])
```

In [170]:

```
print("Before Encoding:", list(df_categorical['Gender'][-10:]))
print("After Encoding:", marry_values[-10:]) print("The inverse from the
encoding result:", marry_encoder.inverse_transform(marry_values[-10:]))
```

```
Before Encoding: ['Male', 'Female', 'Male', 'Male', 'Female', 'Male', 'Male', 'Male', 'Female', 'Male', 'Female']
After Encoding: [1 0 1 1 0 1 1 0 1 0]
The inverse from the encoding result: ['Male' 'Female' 'Male' 'Male' 'Female' 'Male' 'Male' 'Male' 'Female' 'Male' 'Female']
```

In [171]:

```
residence_encoder = LabelEncoder() residence_values =
residence_encoder.fit_transform(df_categorical['Geography'])
```

```
print("Before Encoding:", list(df_categorical['Geography'][:5]))
print("After Encoding:", residence_values[:5]) print("The
inverse from the encoding result:",
residence_encoder.inverse_transform(residence_values[:5]))
```

```
Before Encoding: ['France', 'Spain', 'France', 'France', 'Spain']
After Encoding: [0 2 0 0 2]
The inverse from the encoding result: ['France' 'Spain' 'France' 'France' 'Spain']
```

In [172]: **from** sklearn.preprocessing **import** OneHotEncoder

```
gender_encoder = OneHotEncoder()
```

**from** sklearn.preprocessing **import** OneHotEncoder

In [174]:

```
import numpy as np
```

```
gender_encoder = OneHotEncoder() gender_resaped =
np.array(df_categorical['Gender']).reshape(-1, 1) gender_values =
gender_encoder.fit_transform(gender_resaped)
```

```
print(df_categorical['Gender'][:5])
) print()
print(gender_values.toarray()[:5])
print()
```

```
print(gender_encoder.inverse_transform(gender_values)[:5])

0    Female
1    Female
2    Female
3    Female
4    Female
Name: Gender, dtype: object
```

```
[[1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]]
[['Female']
 ['Female']
 ['Female']
 ['Female']
 ['Female']]
```

In [175]:

```
smoke_encoder = OneHotEncoder()
smoke_resaped = np.array(df_categorical['Surname']).reshape(-1, 1)
smoke_values = smoke_encoder.fit_transform(smoke_resaped)
print(df_categorical['Surname'][:5])
```

```
print()
print(smoke_values.toarray()[:5]) print()
print(smoke_encoder.inverse_transform(smoke_values)[:5])
```

```
0    Hargrave
1    Hill
2    Onio
3    Boni
4    Mitchell
Name: Surname, dtype: object
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]

 [0. 0. 0. ... 0. 0. 0.]]
```

```
[['Hargrave']
```

```
['Hill']
['Onio']
['Boni']
```

```
['Mitchell']]
```

```
work_encoder = OneHotEncoder()
work_resaped = np.array(df_categorical['Geography']).reshape(-1, 1)
work_values = work_encoder.fit_transform(work_resaped)
print(df_categorical['Geography'][:5]) print()
print(work_values.toarray()[:5]) print()
print(work_encoder.inverse_transform(work_values)[:5])
```

In [176]:

```
[1. 0. 0.]
[0. 0. 1.]
[1. 0. 0.]
[1. 0. 0.]
[0. 0. 1.]]
[['France'] ['Spain']]
[['France']]
[['France']]
[['Spain']]
```

```
df_categorical_encoded = pd.get_dummies(df_categorical, drop_first=True)
df_categorical_encoded.head()
```

S S S S S

S S  
Su  
u

n	n	na	na	r	a	na
		a	a	me	n	n
		Z	Z	e	n	n
		ra	gr	n a	u	ub

**e**

---

---

---

**m**

owa,

**e**

---

A

**b****bi**

**e**

0

0000

0000

```
.
00000000000300000000000.
.

.
00000000010400000000000.
.
```

5 rows × 2934 columns

```
In [179]: df_new = pd.concat([df_numeric, df_categorical_encoded], axis=1)
df_new.head()
```

Out[179]:



Row Number	Character	Code	Category	Sub-category	Item	Value	Unit	Notes
1	1	5	6	3	4	6	0	2
0	1	5	6	3	4	6	0	2
1	1	5	6	3	4	6	0	2
2	1	5	6	3	4	6	0	2
3	1	5	6	3	4	6	0	2
4	1	5	6	3	4	6	0	2
5	1	5	6	3	4	6	0	2
6	1	5	6	3	4	6	0	2
7	1	5	6	3	4	6	0	2
8	1	5	6	3	4	6	0	2
9	1	5	6	3	4	6	0	2
10	1	5	6	3	4	6	0	2
11	1	5	6	3	4	6	0	2
12	1	5	6	3	4	6	0	2
13	1	5	6	3	4	6	0	2
14	1	5	6	3	4	6	0	2
15	1	5	6	3	4	6	0	2
16	1	5	6	3	4	6	0	2
17	1	5	6	3	4	6	0	2
18	1	5	6	3	4	6	0	2
19	1	5	6	3	4	6	0	2
20	1	5	6	3	4	6	0	2
21	1	5	6	3	4	6	0	2
22	1	5	6	3	4	6	0	2
23	1	5	6	3	4	6	0	2
24	1	5	6	3	4	6	0	2
25	1	5	6	3	4	6	0	2
26	1	5	6	3	4	6	0	2
27	1	5	6	3	4	6	0	2
28	1	5	6	3	4	6	0	2
29	1	5	6	3	4	6	0	2
30	1	5	6	3	4	6	0	2
31	1	5	6	3	4	6	0	2
32	1	5	6	3	4	6	0	2
33	1	5	6	3	4	6	0	2
34	1	5	6	3	4	6	0	2
35	1	5	6	3	4	6	0	2
36	1	5	6	3	4	6	0	2
37	1	5	6	3	4	6	0	2
38	1	5	6	3	4	6	0	2
39	1	5	6	3	4	6	0	2
40	1	5	6	3	4	6	0	2
41	1	5	6	3	4	6	0	2
42	1	5	6	3	4	6	0	2
43	1	5	6	3	4	6	0	2
44	1	5	6	3	4	6	0	2
45	1	5	6	3	4	6	0	2
46	1	5	6	3	4	6	0	2
47	1	5	6	3	4	6	0	2
48	1	5	6	3	4	6	0	2
49	1	5	6	3	4	6	0	2
50	1	5	6	3	4	6	0	2
51	1	5	6	3	4	6	0	2
52	1	5	6	3	4	6	0	2
53	1	5	6	3	4	6	0	2
54	1	5	6	3	4	6	0	2
55	1	5	6	3	4	6	0	2
56	1	5	6	3	4	6	0	2
57	1	5	6	3	4	6	0	2
58	1	5	6	3	4	6	0	2
59	1	5	6	3	4	6	0	2
60	1	5	6	3	4	6	0	2
61	1	5	6	3	4	6	0	2
62	1	5	6	3	4	6	0	2
63	1	5	6	3	4	6	0	2
64	1	5	6	3	4	6	0	2
65	1	5	6	3	4	6	0	2
66	1	5	6	3	4	6	0	2
67	1	5	6	3	4	6	0	2
68	1	5	6	3	4	6	0	2
69	1	5	6	3	4	6	0	2
70	1	5	6	3	4	6	0	2
71	1	5	6	3	4	6	0	2
72	1	5	6	3	4	6	0	2
73	1	5	6	3	4	6	0	2
74	1	5	6	3	4	6	0	2
75	1	5	6	3	4	6	0	2
76	1	5	6	3	4	6	0	2
77	1	5	6	3	4	6	0	2
78	1	5	6	3	4	6	0	2
79	1	5	6	3	4	6	0	2
80	1	5	6	3	4	6	0	2
81	1	5	6	3	4	6	0	2
82	1	5	6	3	4	6	0	2
83	1	5	6	3	4	6	0	2
84	1	5	6	3	4	6	0	2
85	1	5	6	3	4	6	0	2
86	1	5	6	3	4	6	0	2
87	1	5	6	3	4	6	0	2
88	1	5	6	3	4	6	0	2
89	1	5	6	3	4	6	0	2
90	1	5	6	3	4	6	0	2
91	1	5	6	3	4	6	0	2
92	1	5	6	3	4	6	0	2
93	1	5	6	3	4	6	0	2
94	1	5	6	3	4	6	0	2
95	1	5	6	3	4	6	0	2
96	1	5	6	3	4	6	0	2
97	1	5	6	3	4	6	0	2
98	1	5	6	3	4	6	0	2
99	1	5	6	3	4	6	0	2
100	1	5	6	3	4	6	0	2

5 rows  $\times$  2945 columns

## 8.Split the data into dependent and independent variables.

```
df=pd.read_csv('/content/Churn_Modelling.csv')
```

In [180]:

```
print(df["Balance"].min()) print(df["Balance"].max())  
print(df["Balance"].mean())
```

In [182]:

```
0.0  
250898.09 76485.889288  
print(df.count(0))
```

In [183]:

```
RowNumber      10000  
CustomerId     10000  
Surname        10000  
CreditScore    10000  
Geography      10000  
Gender Age     10000  
Tenure         10000  
Balance        10000  
NumOfProducts 10000  
HasCrCard      10000  
IsActiveMember 10000  
EstimatedSalary 10000  
Exited dtype:   10000  
int64          10000
```

In [184]:

```
print(df.shape)  
(10000, 14)
```

In [185]:

```
print(df.size)  
140000
```

In [187]:

```
X = df.iloc[:, :-1].values print(X)  
[[1 15634602 'Hargrave' ... 1 1 101348.88]  
 [2 15647311 'Hill' ... 0 1 112542.58]  
 [3 15619304 'Onio' ... 1 0 113931.57] ...  
 [9998 15584532 'Liu' ... 0 1 42085.58]  
 [9999 15682355 'Sabbatini' ... 1 0 92888.52]  
 [10000 15628319 'Walker' ... 1 0 38190.78]]  
Y = df.iloc[:, -1].values print(Y)  
[1 0 1 ... 1 1 0]
```

In [271]:

## 9.Scale the independent variables

In [215]:

```
df = pd.read_csv('/content/Churn_Modelling.csv')

x = df[['Age', 'Tenure']].values y = df['Gender'].values
fig, ax = plt.subplots(ncols=2, figsize=(12, 4))

ax[0].scatter(x[:,0], y) ax[1].scatter(x[:,1], y)
plt.show()
```

In [216]:

```
fig, ax = plt.subplots(figsize=(12, 4))

ax.scatter(x[:,0], y)
ax.scatter(x[:,1], y)

<matplotlib.collections.PathCollection at 0x7f9a8a854ad0>
```

Out[216]:

In [217]:

```
fig, ax = plt.subplots(figsize=(12, 4))

ax.hist(x[:,0]) ax.hist(x[:,1])
```

Out[217]:

```
(array([ 413., 1035., 1048., 1009., 989., 1012., 967., 1028., 1025.,
        1474.]), array([ 0., 1., 2., 3., 4., 5., 6., 7., 8.,
        9., 10.])),
<a list of 10 Patch objects>)
```

In [220]:

```
from sklearn.preprocessing import StandardScaler from
sklearn.preprocessing import MinMaxScaler fig, ax = plt.subplots(figsize=(12,
4))

scaler = StandardScaler() x_std
= scaler.fit_transform(x)
ax.hist(x_std[:,0])
ax.hist(x_std[:,1])
```

Out[220]:

```
(array([ 413., 1035., 1048., 1009., 2001.,      0., 1995.,      0., 1025.,
        1474.]), array([-1.73331549, -1.38753759, -1.04175968, -
        0.69598177, -0.35020386,
        -0.00442596, 0.34135195, 0.68712986, 1.03290776, 1.37868567,
        1.72446358])),
<a list of 10 Patch objects>)
```

In [219]:

```
fig, ax = plt.subplots(figsize=(12, 4))
```

```
scaler = StandardScaler() x_std  
= scaler.fit_transform(x)  
ax.scatter(x_std[:,0], y)  
ax.scatter(x_std[:,1], y)
```

Out[219]:

```
<matplotlib.collections.PathCollection at 0x7f9a8a2fde50>
```

In [221]:

```
fig, ax = plt.subplots(figsize=(12, 4))
```

```
scaler = MinMaxScaler()  
x_minmax = scaler.fit_transform(x)  
ax.hist(x_minmax[:,0])  
ax.hist(x_minmax  
[:,1])
```

Out[221]:

```
(array([ 413., 1035., 1048., 1009., 989., 1012., 967., 1028., 1025.,  
        1474.]), array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7,  
        0.8, 0.9, 1. ]),  
<a list of 10 Patch objects>)
```

In [222]:

```
fig, ax = plt.subplots(figsize=(12, 4))
```

```
scaler = MinMaxScaler()  
x_minmax = scaler.fit_transform(x)  
ax.scatter(x_minmax[:,0], y)  
ax.scatter(x_minmax[:,1], y)
```

Out[222]:

```
<matplotlib.collections.PathCollection at 0x7f9a8a0cae10>
```

In [223]:

```
fig, ax = plt.subplots(figsize=(12, 4))
```

```
scaler = MinMaxScaler() x_minmax =  
scaler.fit_transform(x)  
ax.scatter(x_minmax[:,0], y)
```

Out[223]:

```
<matplotlib.collections.PathCollection at 0x7f9a8a0caf10>
```

In [224]:

```
fig, ax = plt.subplots(figsize=(12, 4))
```

```
scaler = MinMaxScaler() x_minmax =  
scaler.fit_transform(x)  
ax.hist(x_minmax[:,0])
```

Out[224]:

```
(array([ 611., 2179., 3629., 1871., 910., 441., 208., 127., 20.,  
        4.]),
```

```
array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),  
<a list of 10 Patch objects>)
```

In [227]:

```
from sklearn.model_selection import train_test_split from
```

```
sklearn.pipeline import Pipeline
```

```
from sklearn.linear_model import SGDRegressor from sklearn.preprocessing  
import StandardScaler from
```

```
sklearn.preprocessing import MinMaxScaler from sklearn.metrics import  
mean_absolute_error import sklearn.metrics as metrics
```

```
import pandas as pd import
```

```
numpy as np import
```

```
matplotlib.pyplot as plt
```

```
# Import Data
```

```
df = pd.read_csv('/content/Churn_Modelling.csv')  
x = df[['Age', 'Tenure']].values y =  
df['Balance'].values
```

```
# Split into a training and testing set
```

```
X_train, X_test, Y_train, Y_test = train_test_split(x, y)
```

```
# Define the pipeline for scaling and model fitting
```

```
pipeline = Pipeline([  
    ("MinMax Scaling", MinMaxScaler()),  
    ("SGD Regression", SGDRegressor())  
])
```

```
# Scale the data and fit the model
```

```
pipeline.fit(X_train, Y_train)
```

```
# Evaluate the model
```

```
Y_pred = pipeline.predict(X_test)
```

```
print('Mean Absolute Error: ', mean_absolute_error(Y_pred, Y_test))  
print('Score', pipeline.score(X_test, Y_test))
```

```
Mean Absolute Error: 57120.533393590835
```

```
Score 0.0004207814312172653
```

## 10.Split the data into training and testing

```
dataset = pd.read_csv('/content/Churn_Modelling.csv')  
print(dataset)
```

In [267]:

```
 \      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]

In [287]:

```
dataset.drop(["HasCrCard"],axis=1,inplace=True)
```

print(dataset.shape)#no. of rows and columne

In [288]:

```

print(dataset.head(10))
(10000, 7)
      CustomerId CreditScore Age Tenure          Balance IsActiveMember \
0      15634602      619 42          2 0.00          1
1      15647311      608 41          1 83807.86          1
2      15619304      502 42          8 159660.80          0 3 15701354          699 39
      1          0.00 0 4 15737888          850 43          2 125510.82          1
5 15574012          645 44          8 113755.78          0 6 15592531          822
50 7          0.00 1
7 15656148          376 29          4 115046.74          0 8 15792365          501
44 4 142051.07          1
9 15592389          684 27          2 134603.88          1

```

EstimatedSalary



```

0    101348.88 1
    112542.58 2
    113931.57
3         93826.63
4         79084.10
5        149756.71
6         10062.80
7        119346.88
8         74940.50
9         71725.73

```

In [289]:

```

X=dataset.iloc[:, :-1].values
X

```

```

Out[289]: array([[1.5634602e+07, 6.1900000e+02, 4.2000000e+01,
2.0000000e+00,
    0.0000000e+00, 1.0000000e+00],
[1.5647311e+07, 6.0800000e+02, 4.1000000e+01, 1.0000000e+00,
    8.3807860e+04, 1.0000000e+00],
[1.5619304e+07, 5.0200000e+02, 4.2000000e+01, 8.0000000e+00,
    1.5966080e+05, 0.0000000e+00],
...,
[1.5584532e+07, 7.0900000e+02, 3.6000000e+01, 7.0000000e+00,
    0.0000000e+00, 1.0000000e+00],
[1.5682355e+07, 7.7200000e+02, 4.2000000e+01, 3.0000000e+00,
    7.5075310e+04, 0.0000000e+00],
[1.5628319e+07, 7.9200000e+02, 2.8000000e+01, 4.0000000e+00,
    1.3014279e+05, 0.0000000e+00]])

```

```

Y=dataset.iloc[:, -1].values
Y

```

In [290]:

```

array([101348.88, 112542.58, 113931.57, ..., 42085.58, 92888.52,
Out[290]: 38190.78])

```

```

from sklearn.model_selection import train_test_split

```

In [291]:

```

X_train,X_test,Y_train,Y_test = train_test_split( X, Y, test_size = 0.25,
random_state = 0 )

```

In [306]:

```

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train = sc.fit_transform(X_train) X_test
= sc.transform(X_test) print(X_train)

```

```

[[-1.34333028 -0.73550706 0.01526571 0.00886037 0.67316003 -1.03446007]
 [ 1.55832963 1.02442719 -0.65260917 0.00886037 -1.20772417 -1.03446007]
 [-0.65515619 0.80829492 -0.46178778 1.39329338 -0.35693706 0.96668786]
...

```

```
[-1.63542994  0.90092304 -0.36637708  0.00886037  1.36657199 -1.03446007]
[-0.38540456 -0.62229491 -0.08014499  1.39329338 -1.20772417  0.96668786] [-
1.37829524 -0.28265848  0.87396199 -1.37557264  0.51741687 -1.03446007]]
```

In [305]:

```
print(X_test)
[[-1.05852196 -0.55025082 -0.36637708  1.04718513  0.88494297  0.96668786]
 [-0.51554728 -1.31185979  0.11067641 -1.02946438  0.43586703 -1.03446007]
 [-0.8058485  0.57157862  0.3014978  1.04718513  0.31486378  0.96668786]
 ...]
```

```
[ 0.25326371 1.95070838 0.01526571 -1.37557264 0.30819395 -1.03446007]
[-0.17836122 0.29369426 -0.08014499 0.70107688 0.55698791 -1.03446007]
[ 0.40190663 0.870047 -0.74801987 -0.68335613 0.7006957 -1.03446007]]
```