WEEK-9

Logistic Regression

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

insurance=pd.read_csv("insurance_data.csv")
insurance.shape
```

(27, 2)

```
insurance.head()
```

	age	bought_insurance
0	22	0
1	25	0
2	47	1
3	52	0
4	46	1

```
insurance.describe()
```

	age	bought_insurance
count	27.000000	27.000000
mean	39.666667	0.518519
std	15.745573	0.509175
min	18.000000	0.000000
25%	25.000000	0.000000
50%	45.000000	1.000000
75%	54.500000	1.000000
max	62.000000	1.000000

```
insurance.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27 entries, 0 to 26
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   age              27 non-null     int64
 1   bought_insurance  27 non-null     int64
dtypes: int64(2)
memory usage: 560.0 bytes
```

```
insurance[insurance["age"]>35]
insurance[insurance["bought_insurance"]==1]
```

	age	bought_insurance
2	47	1
4	46	1

```
5    56              1
7    60              1
8    62              1
9    61              1
14   49              1
15   55              1
16   25              1
17   58              1
22   40              1
23   45              1
24   50              1
25   54              1
```

```
x=insurance.drop("bought_insurance",axis=1)
x.head()
```

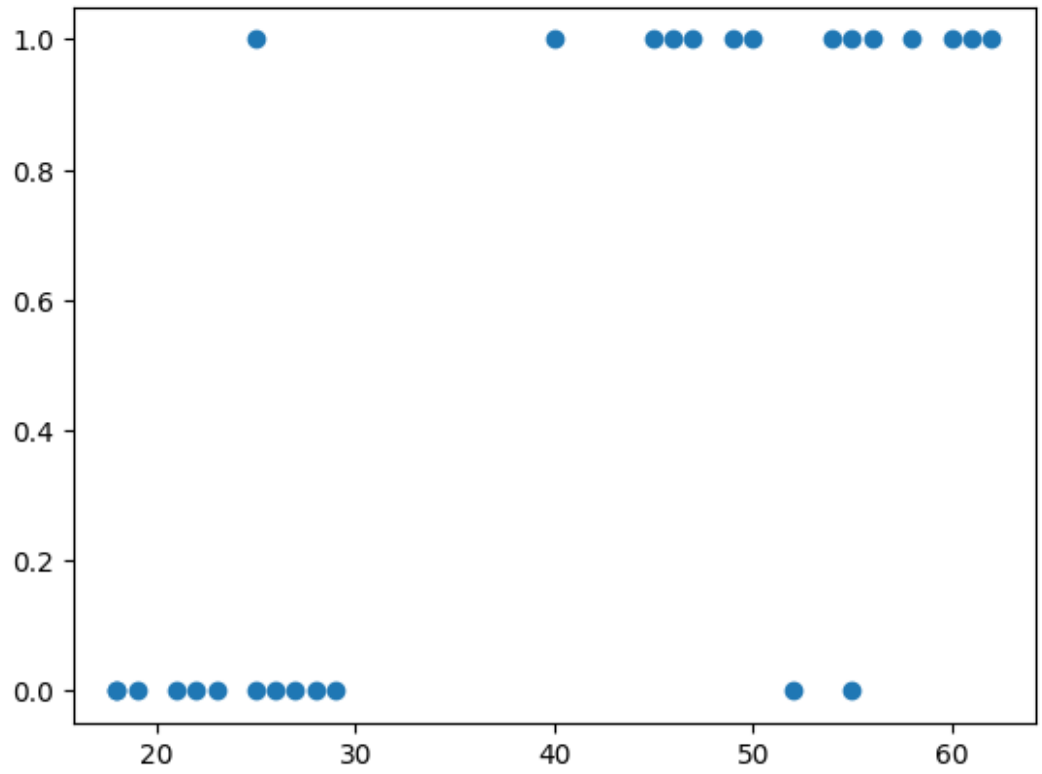
```
   age
0   22
1   25
2   47
3   52
4   46
```

```
y=insurance["bought_insurance"]
y.head()
```

```
0    0
1    0
2    1
3    0
4    1
Name: bought_insurance, dtype: int64
```

```
plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x116daba4cd0>
```




```
sns.regplot(x=insurance['age'],y=insurance['bought_insurance'])
```

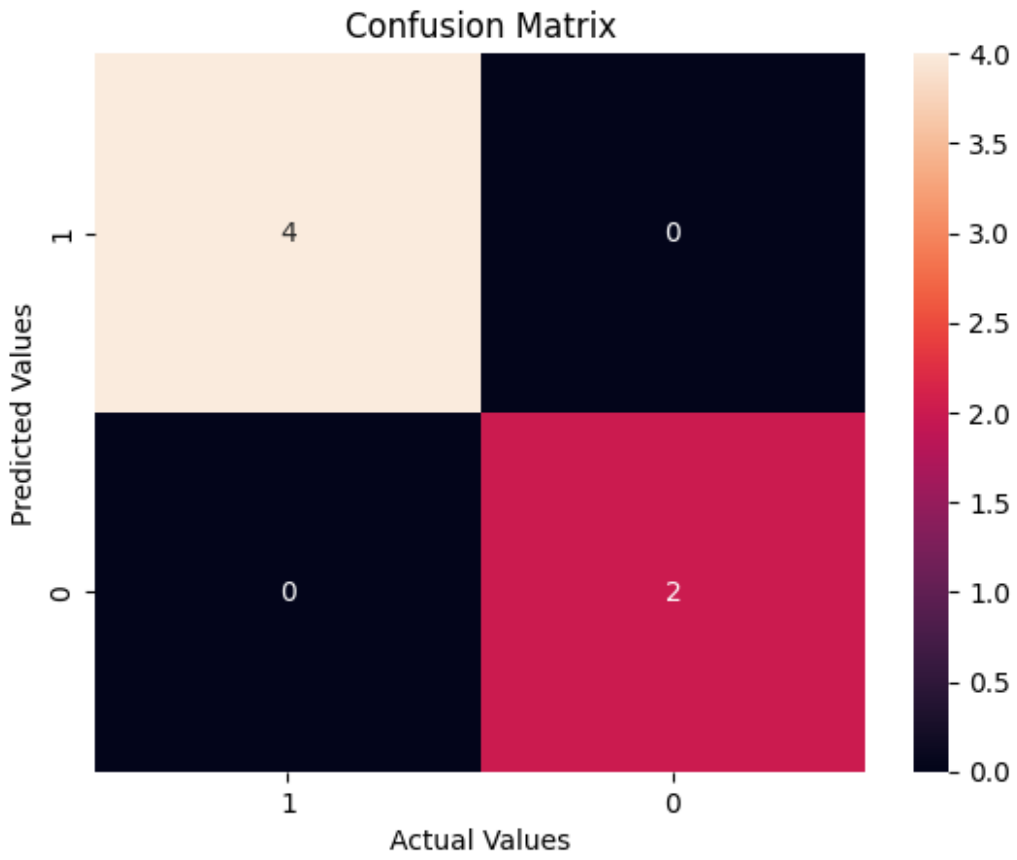
```
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.2,random_state=10)
```

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train,Y_train)
y_predicted=model.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,y_predicted)
```

```
sns.heatmap(cm,
            annot=True,
            xticklabels=["1","0"],
            yticklabels=["1","0"]
           )
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Confusion Matrix")
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```


Confusion Matrix
Predicted Values
1
0
4
0
0
2
1
0
Actual Values
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0