As we know, In Regression Problem where value is Continuous, We can use Linear Regression to predict a Continuous number value such as Home prices, Weather or Stock prices

So Predicted value is continuous

Now, there is a second type, called Classification problem where value is Categorical for example :

1. Identify if an Email is spam or not?
2. Will customer buy life insurance?
3. Which party a person is going to vote for?

    A. Democratic
    B. Republican
    C. Independent

Now, in Classification There is Two Types :

1. Binary Classification : Where you have only two categories which is a YES or NO outcome, for example Will a customer buy life insurance? the answer will be in yes or no
2. Multi Class Classification : Where you have more than two categories such as, Which party a person is going to vote for? with 3 different options as given above

The Task

lets say you work at a life insurance company and you are given a task to predict how likely a potential customer is to buy your insurance

So, as we can see from the data table below, it is a Binary type of classification where the outcome is yes/no or true/false or 1/0

Import all libraries

In [61]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
```

Reading the Data

In [62]:
```
data = pd.read_csv('insurance_data.csv')
data
```

Out[62]:

	age	bought_insurance
0	22	0
1	25	0
2	47	1
3	52	0
4	46	1
5	56	1
6	55	0
7	60	1
8	62	1
9	61	1
10	18	0
11	28	0
12	27	0
13	29	0
14	49	1
15	55	1
16	25	1
17	58	1
18	19	0
19	18	0
20	21	0
21	26	0
22	40	1
23	45	1
24	50	1
25	54	1
26	23	0

As we can understand from the pattern above that younger age people mostly not likely to buy the insurance whereas an older people are more likely to buy the insurance

Using Linear Regression on this sort of problem will not be suitable, watch codebasics Logistic Regression 8th ML tutorial to know why!

Hence, we can use Logistic Regression for this as even tho it has Regression in its name, but it is mostly used for Classification problem

Logistic Regression has like an S line shape using Sigmoid or Logit Function.

```
sigmoid(z) = 1/1+e^-z
```

e = Euler's number = 2.71828

Now, this about this equation for a moment, what we are doing here is, we are dividing by 1 by a number which is slightly greater than 1, and when you have this situation, the outcome will be less than 1, so all you are doing with this Sigmoid function is coming up with a range between 0 and 1.

So if you fit set of numbers to this Sigmoid function, all it will do is convert them to 0 to 1 range and the line you will get is like an S shape line unlike a straight line in Linear Regression

As we know the equation for our Linear Regression like is y = m * x + b

so all you are doing is, you are feeding this line into a Sigmoid function and you convert this line into an S shape with the equation below, we can see that the 'z' is replaced with m*x+b

```
y = 1/1 + e^-(m*x+b)
```

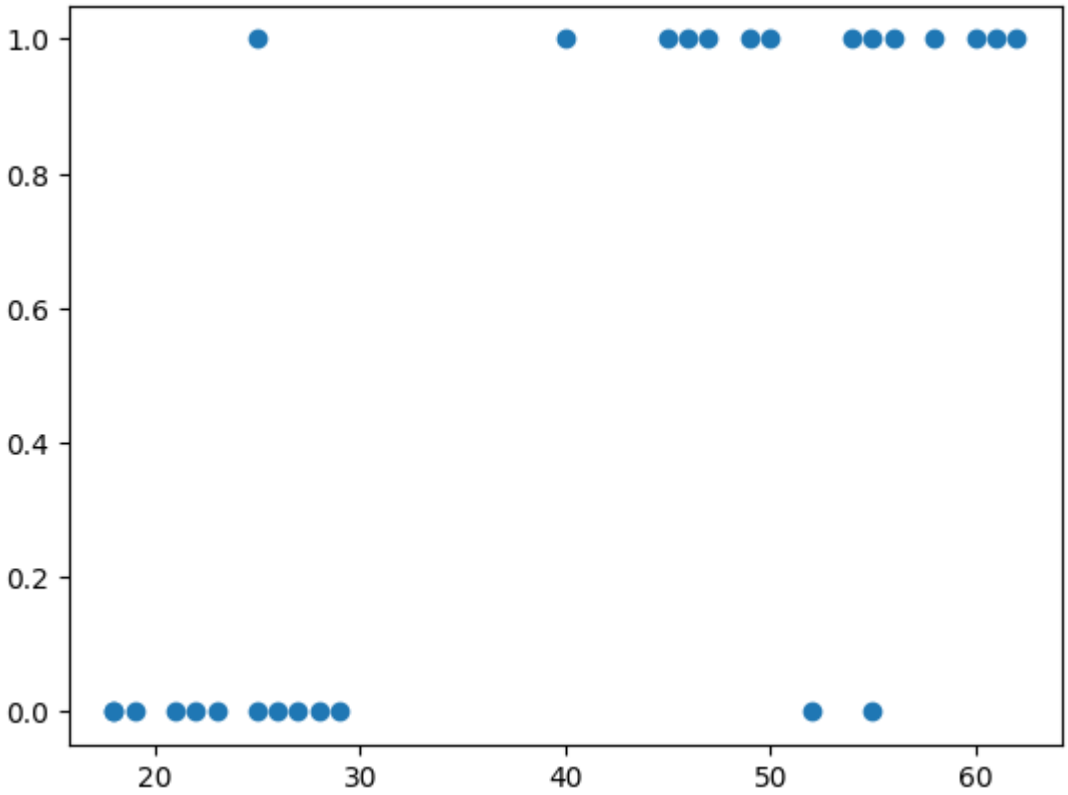
So basically, we applied the Sigmoid function on top of the Linear regression line and thats how we got our S shaped line for Logistic Regression

All this maths are just for understanding, when you code, everything is already calculated so you dont have to worry but we must know how the calculation process is

Plotting scatter plot

In [63]:
```
plt.scatter(data.age, data.bought_insurance)
```

Out[63]:
```
<matplotlib.collections.PathCollection at 0x1ee5ee54e10>
```



We know the answer is either 0 or 1, so in the graph we can also see that, the younger customer dont likely buy the insurance hence they are at 0, and the older the customer is the more likely they are to buy the insurance hence at 1

In [64]:
```
data.shape
```

Out[64]:
```
(27, 2)
```

Train Test Splitting

In [65]:
```
from sklearn.model_selection import train_test_split
```

In [66]:
```
X = data[['age']] # Because we need X as 2D array
y = data.bought_insurance
```

In [67]:
```
X_train, X_test, y_train, y_test =  train_test_split(X, y, test_size=0.2)
```

In [68]:
```
X_test
```

Out[68]:

	age
23	45
4	46
24	50
20	21
25	54
1	25

Choosing Logistic Regression as our model

In [69]:
```
model = LogisticRegression()
```

In [70]:
```
model.fit(X_train, y_train)
```

Out[70]:

▼ LogisticRegression

LogisticRegression()

Predicting

In [71]:
```
predict = model.predict(X_test)
```

Making a Data Frame of the Actual and Predicted data

In [100…
```
predicted_data = {
#     "Age" : X_test,
    "Actual Value" : y_test,
    "Predicted Value" : predict
}

newdf = pd.DataFrame(predicted_data)
newdf
```

Out[100]:

	Actual Value	Predicted Value
23	1	1
4	1	1
24	1	1
20	0	0
25	1	1
1	0	0

Checking score

In [102…
```
model.score(X_test, y_test)
```

Out[102]:
```
1.0
```

Predicting a sinle value

In [103…
```
model.predict([[41]])
```

```
C:\Users\User\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklea
rn\base.py:465: UserWarning: X does not have valid feature names, but Logisti
cRegression was fitted with feature names
  warnings.warn(
```

Out[103]:
```
array([0], dtype=int64)
```

Checking prediction probability of X_test

In [104…
```
model.predict_proba(X_test)
```

Out[104]:
```
array([[0.38810639, 0.61189361],
       [0.36126965, 0.63873035],
       [0.26343843, 0.73656157],
       [0.90842994, 0.09157006],
       [0.18444936, 0.81555064],
       [0.8625103 , 0.1374897 ]])
```