Logistic Regression

Logistic Regression is a technique that is used in classification.
Logistic Regression predict catagorical Values.
E.g: Problem such as either email is spam or not, will customer buy an insurance, which party is a person going to vote and etc.
Clssification is of two types:

Binary Classification:

Binary classification have only two binary numbers 0 and 1. E.g: Either person buy an insurance or not.

Muliclass classification:

Used when we have more than one catagories e.g which party a person is going to vote in Karachi. There could a serveral answers such as PTI, PPP,PMLN, MQM, JI, JUI, or etc.

Linear Regression vs Logistic Regression

Whenever we have a continuos values, the linear regression performs well. While linear regression don't when our target is discrete.
Logistic Regression works with sigmoid function.

Sigmoid Function or Logit Function:

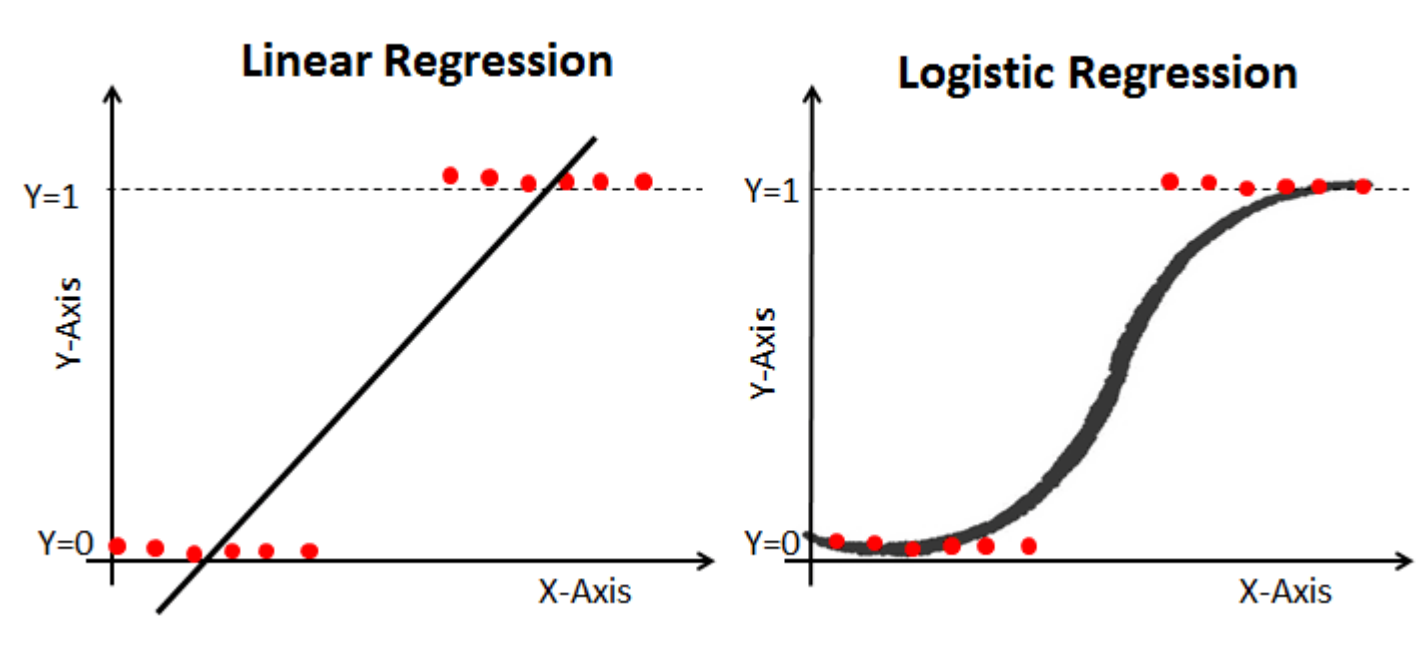
Sigmoid function is used to map the entire number line into a small range such as between 0 & 1. Sigmoid function draw a S shape curve.

Formula for Sigmoid Function

$$S(x) = \frac{1}{1 + e^{-x}}$$

for logistic regression we would be using

x = wa + b This is nothing but a formula of linear regression. We would be converting linear equation between 0 & 1.



In [ ]:

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
```

Reading csv file

In [5]:
```
df = pd.read_csv("insurance_data.csv")
df.sample(5)
```

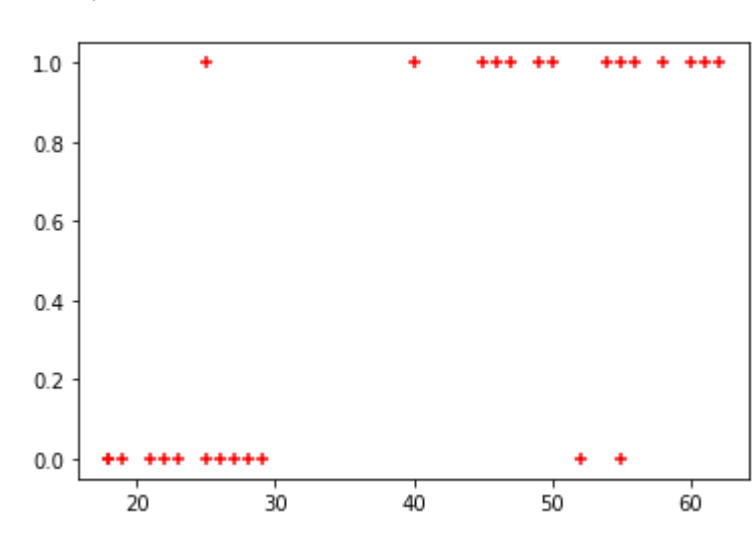
Out[5]:

	age	bought_insurance
9	61	1
25	54	1
13	29	0
26	23	0
18	19	0

Drawing Scatter of dataset

In [6]:
```
plt.scatter(df.age, df.bought_insurance, marker='+', color='red')
```

Out[6]:
```
<matplotlib.collections.PathCollection at 0x20340701130>
```



In [7]:
```
df.shape
```

Out[7]:
```
(27, 2)
```

Train test split technique is used to estimate the performance of machine learning algorithms which are used to make predictions on data not used to train the model. We just divide our dataset into training and testing part. For general we used 20 percent as test size and 80 percent as test size.

In [8]:
```
from sklearn .model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(df[['age']], df.bought_insurance, test_size=0.2)
```

In [9]:
```
X_test
```

Out[9]:

	age
22	40
18	19
14	49
10	18
16	25
19	18

In [11]:
```
Y_test
```

Out[11]:
```
22    1
18    0
14    1
10    0
16    1
19    0
Name: bought_insurance, dtype: int64
```

Applying logistic Regression

In [12]:
```
model = LogisticRegression()
```

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

Geting an overall score of test set. Score tell use the performance of model on test data

In [14]:
```
model.score(X_test, Y_test)
```

Out[14]:
```
0.6666666666666666
```

In [15]:
```
model.predict(X_test)
```

Out[15]:
```
array([0, 0, 1, 0, 0, 0], dtype=int64)
```

In [16]:
```
Y_test
```

Out[16]:
```
22    1
18    0
14    1
10    0
16    1
19    0
Name: bought_insurance, dtype: int64
```

In [ ]:

In [ ]:

Logistic Regression on Multiclass

Load Digit Data:

The load digit data contain hand writtern images of digits. The dataset contains 1797 images of 8X8.

In [ ]:

In [ ]:

In [17]:
```
from sklearn.datasets import load_digits
```

In [18]:
```
digits = load_digits()
```

Lets show the shape of our images dataset by using shape function. Here it shows that total number of images are 1797 and the size of images is 8X8.

In [44]:
```
digits.images.shape
```

Out[44]:
```
(1797, 8, 8)
```

Our images are already transformed into numpy array. Lets see the images in numpy array.

Here we are just displaying our first image

In [23]:
```
digits.data[1]
```

Out[23]:
```
array([ 0.,  0.,  0., 12., 13.,  5.,  0.,  0.,  0.,  0.,  0., 11., 16.,
        9.,  0.,  0.,  0.,  0.,  3., 15., 16.,  6.,  0.,  0.,  0.,  7.,
       15., 16., 16.,  2.,  0.,  0.,  0.,  0.,  1., 16., 16.,  3.,  0.,
        0.,  0.,  0.,  1., 16., 16.,  6.,  0.,  0.,  0.,  0.,  1., 16.,
       16.,  6.,  0.,  0.,  0.,  0.,  0., 11., 16., 10.,  0.,  0.])
```

Through target we are showing the y label of our first image

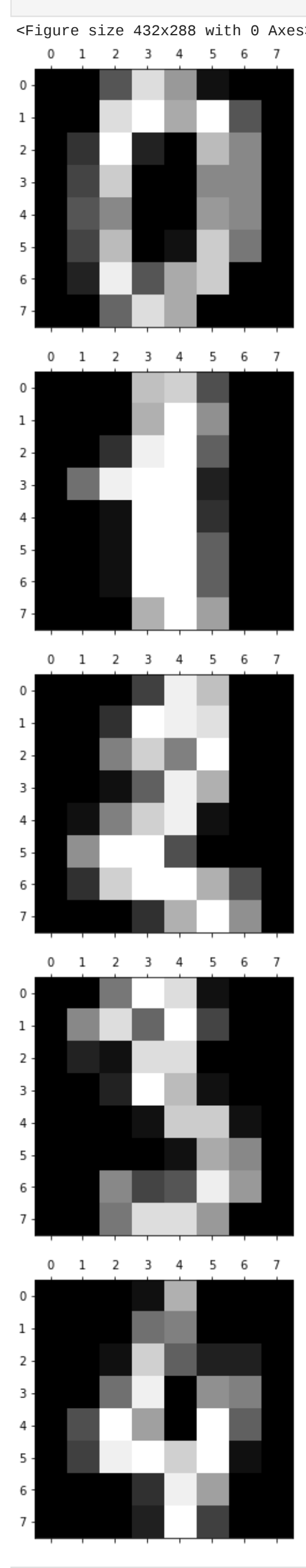
In [24]:
```
digits.target[1]
```

Out[24]:
```
1
```

Through matplotlib matshow function, we can display the images into orignal form.

In [21]:
```
plt.gray()
for i in range(5):
    plt.matshow(digits.images[i])
```

```
<Figure size 432x288 with 0 Axes>
```



In [26]:
```
X_train, X_test, Y_train, Y_test = train_test_split(digits.data,digits.target, test_size= 0.2 )
```

In [27]:
```
model.fit(X_train, Y_train)
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[27]:
```
LogisticRegression()
```

In [28]:
```
model.score(X_test, Y_test)
```

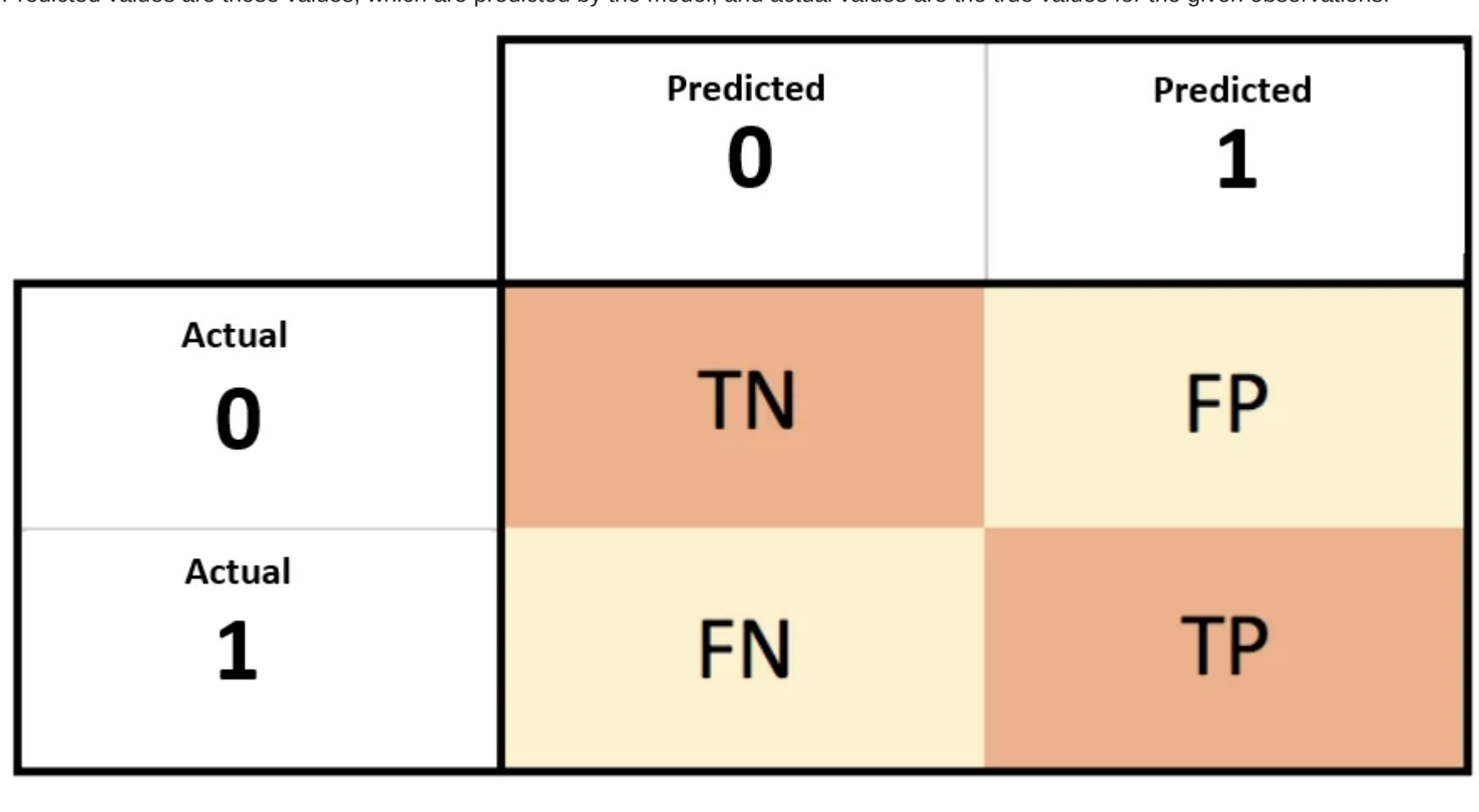
Out[28]:
```
0.975
```

Here our model perform 97.5 percent well but still it lacks the accuracy of 2.5 percent. So it identify where our model makes mistakes, we can draw a confusion matrix.

Confusion Matrix

Confusion matrix is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.

For the 2 prediction classes of classifiers, the matrix is of 2*2 table, for 3 classes, it is* 33 table, and so on.
The matrix is divided into two dimensions, that are predicted values and actual values along with the total number of predictions.
Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.



In [ ]:

In [30]:
```
from sklearn.metrics import confusion_matrix
```

Lets first se the model answers for our X_test, than we comapare it with Y_Test to see the model performance

In [31]:
```
y_pred = model.predict(X_test)
```

In [33]:
```
y_pred
```

Out[33]:
```
array([5, 6, 3, 5, 3, 4, 4, 6, 4, 9, 2, 8, 3, 7, 6, 6, 7, 9, 2, 0, 0, 7,
       5, 7, 3, 4, 1, 1, 2, 2, 2, 9, 1, 8, 6, 8, 8, 4, 6, 2, 4, 3, 7, 4,
       3, 4, 7, 5, 7, 9, 3, 3, 2, 8, 7, 4, 7, 3, 9, 9, 1, 4, 0, 2, 7, 8,
       5, 5, 5, 0, 3, 2, 8, 1, 4, 1, 8, 4, 1, 8, 8, 1, 9, 9, 2, 4, 2, 6,
       6, 2, 1, 0, 4, 3, 2, 6, 3, 7, 0, 3, 1, 7, 8, 2, 6, 8, 6, 7, 6, 8,
       6, 3, 2, 5, 1, 1, 1, 0, 7, 3, 7, 3, 7, 2, 8, 8, 3, 4, 0, 7, 3, 0,
       1, 5, 3, 9, 9, 8, 9, 9, 6, 4, 6, 3, 9, 8, 1, 6, 2, 9, 6, 3, 8, 8,
       2, 8, 5, 5, 8, 6, 9, 3, 7, 8, 4, 5, 6, 7, 1, 8, 3, 2, 3, 5, 4, 3,
       1, 8, 3, 7, 7, 3, 0, 3, 9, 9, 0, 9, 5, 3, 4, 9, 1, 7, 1, 4, 9, 6,
       3, 5, 9, 3, 3, 0, 8, 0, 1, 2, 9, 4, 1, 0, 7, 2, 0, 2, 9, 7, 7, 0,
       5, 4, 0, 4, 1, 1, 5, 1, 1, 3, 2, 2, 3, 4, 6, 6, 7, 7, 0, 0, 2, 1,
       5, 6, 7, 5, 0, 4, 1, 2, 5, 9, 8, 2, 7, 5, 1, 7, 1, 7, 4, 7, 7, 5,
       1, 6, 1, 6, 5, 9, 5, 4, 1, 5, 7, 7, 4, 5, 9, 3, 1, 3, 0, 0, 2, 4,
       1, 4, 2, 7, 1, 7, 7, 2, 5, 0, 7, 5, 8, 3, 1, 9, 1, 6, 2, 7, 8, 6,
       0, 4, 4, 9, 8, 3, 7, 9, 3, 1, 4, 6, 6, 0, 6, 1, 2, 7, 6, 5, 1, 5,
       2, 9, 7, 0, 8, 7, 6, 7, 1, 9, 1, 6, 0, 6, 3, 9, 0, 6, 6, 1, 7, 6,
       7, 1, 4, 8, 3, 5, 4, 8])
```

In [34]:
```
Y_test
```

Out[34]:
```
array([5, 6, 3, 5, 3, 4, 4, 6, 4, 9, 2, 8, 3, 7, 6, 6, 7, 9, 2, 0, 0, 7,
       5, 7, 3, 4, 1, 1, 2, 2, 3, 9, 1, 8, 6, 8, 8, 4, 6, 2, 4, 3, 7, 4,
       3, 4, 7, 5, 7, 9, 3, 3, 2, 8, 7, 4, 7, 3, 9, 9, 1, 4, 0, 2, 7, 8,
       5, 5, 5, 0, 3, 2, 8, 1, 4, 1, 8, 4, 1, 8, 8, 1, 9, 9, 2, 4, 2, 6,
       6, 2, 1, 0, 4, 3, 2, 6, 3, 7, 0, 3, 1, 7, 8, 2, 6, 8, 6, 7, 6, 8,
       6, 3, 2, 5, 1, 1, 1, 0, 7, 3, 7, 3, 7, 2, 8, 8, 3, 4, 0, 7, 3, 0,
       1, 5, 3, 9, 9, 8, 9, 9, 6, 4, 6, 3, 9, 8, 1, 6, 2, 9, 6, 3, 8, 8,
       2, 8, 5, 5, 8, 6, 9, 3, 7, 8, 4, 5, 6, 7, 1, 8, 3, 2, 3, 5, 4, 3,
       1, 8, 3, 7, 7, 3, 0, 3, 9, 9, 0, 3, 5, 3, 4, 9, 1, 7, 1, 4, 9, 6,
       3, 8, 9, 3, 3, 0, 8, 0, 8, 2, 9, 4, 5, 0, 7, 2, 0, 2, 9, 7, 7, 0,
       5, 4, 0, 4, 1, 1, 5, 1, 1, 3, 2, 2, 3, 4, 6, 6, 7, 7, 0, 0, 2, 1,
       5, 6, 7, 5, 0, 4, 1, 1, 5, 9, 8, 2, 7, 5, 1, 7, 1, 7, 4, 7, 7, 5,
       1, 6, 1, 6, 5, 5, 5, 4, 1, 5, 7, 7, 4, 5, 9, 3, 1, 3, 0, 0, 2, 4,
       1, 4, 2, 7, 1, 7, 7, 2, 5, 0, 7, 5, 8, 3, 1, 9, 1, 6, 2, 7, 8, 6,
       0, 4, 4, 9, 8, 3, 7, 9, 3, 1, 4, 6, 6, 0, 6, 1, 2, 7, 6, 5, 1, 5,
       2, 5, 7, 0, 4, 7, 6, 7, 1, 9, 1, 6, 0, 6, 3, 9, 0, 6, 6, 1, 7, 6,
       7, 1, 4, 8, 3, 5, 4, 8])
```

Drawing a confusion matrix

In [35]:
```
cm = confusion_matrix(Y_test, y_pred)
```

In [36]:
```
cm
```

Out[36]:
```
array([[28,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 42,  1,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 31,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  1, 41,  0,  0,  0,  0,  0,  1],
       [ 0,  0,  0,  0, 35,  0,  0,  0,  1,  0],
       [ 0,  1,  0,  0,  0, 30,  0,  0,  0,  2],
       [ 0,  0,  0,  0,  0,  0, 37,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 47,  0,  0],
       [ 0,  1,  0,  0,  0,  1,  0,  0, 31,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0, 29]], dtype=int64)
```

In [37]:
```
import seaborn as sn
```

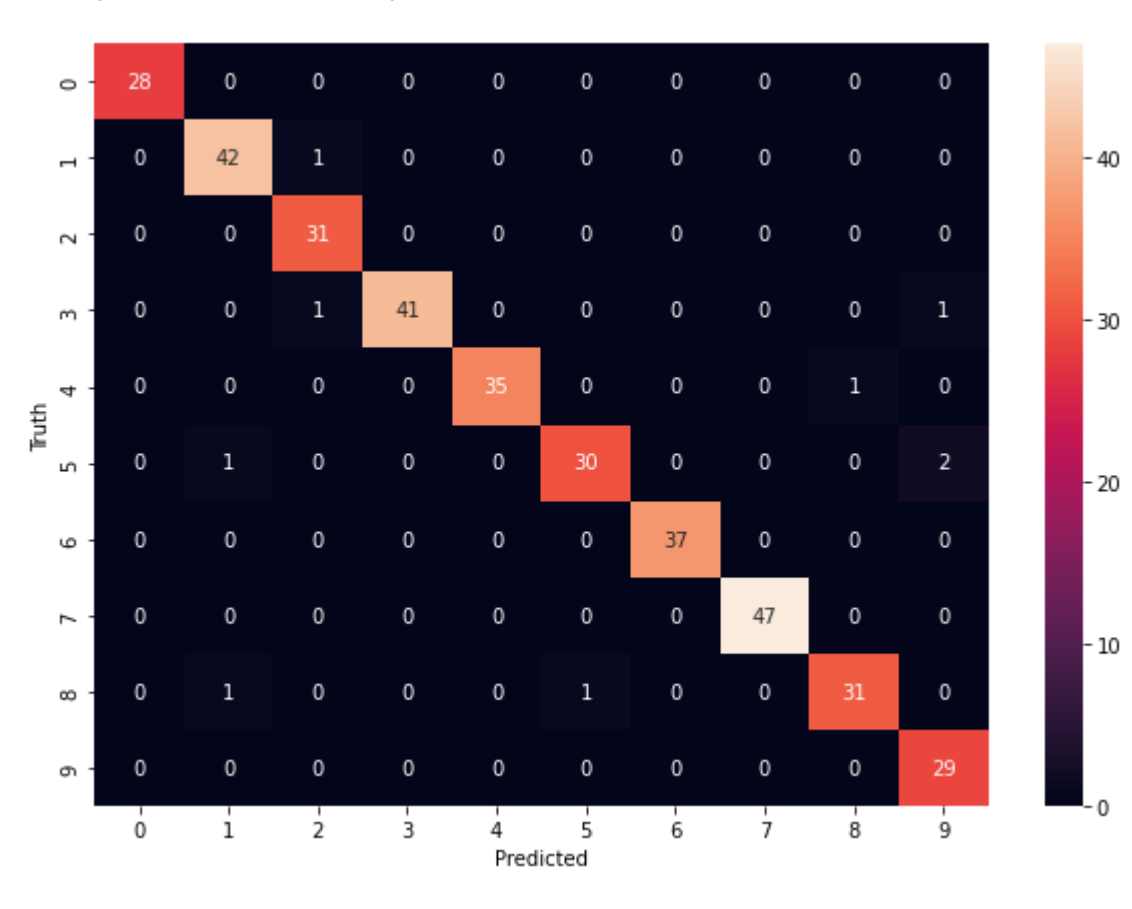
Using a seaborn to add colors on our confusion matrix.

A heatmap is a two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colours.

annot: If True, write the data value in each cell.

In [38]:
```
plt.figure(figsize=(10, 7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[38]:
```
Text(69.0, 0.5, 'Truth')
```



In [ ]:

Classification Report

A classification report is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score, and support of your trained classification model.

In [39]:
```
from sklearn.metrics import classification_report
```

In [ ]:

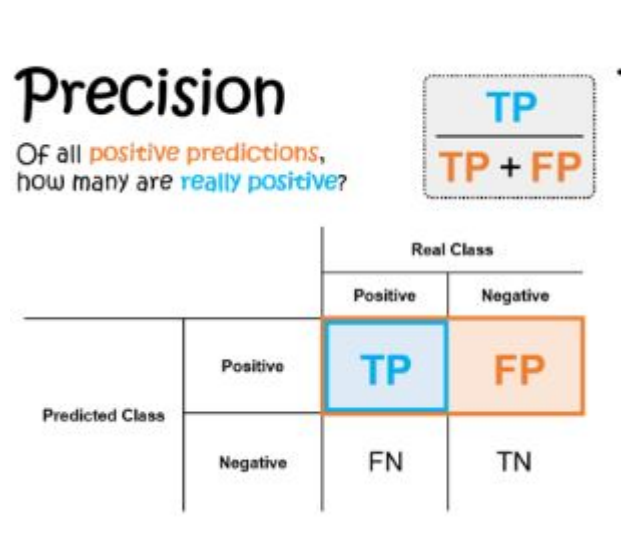
In [40]:
```
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        28
           1       0.95      0.98      0.97        43
           2       0.94      1.00      0.97        31
           3       1.00      0.95      0.98        43
           4       1.00      0.97      0.99        36
           5       0.97      0.91      0.94        33
           6       1.00      1.00      1.00        37
           7       1.00      1.00      1.00        47
           8       0.97      0.94      0.95        33
           9       0.91      1.00      0.95        29

    accuracy                           0.97       360
   macro avg       0.97      0.98      0.97       360
weighted avg       0.98      0.97      0.98       360
```

In [ ]:

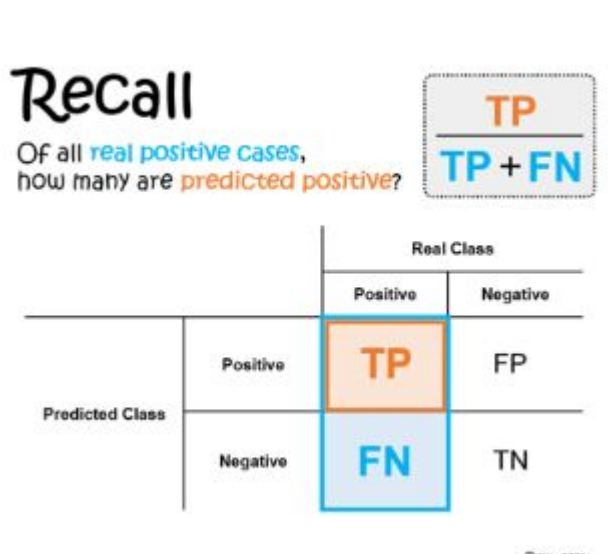
Percision

Percision is the quality of a positive prediction made by the model. Precision refers to the number of true positives divided by the total number of positive predictions



Recall:

The recall is calculated as the ratio between the number of Positive samples correctly classified as Positive to the total number of Positive samples.



F1 Score

F1 score is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model.

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Accuracy

Accuracy is the number of correct predictions Total number of predictions.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

All formulas

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

In [ ]: