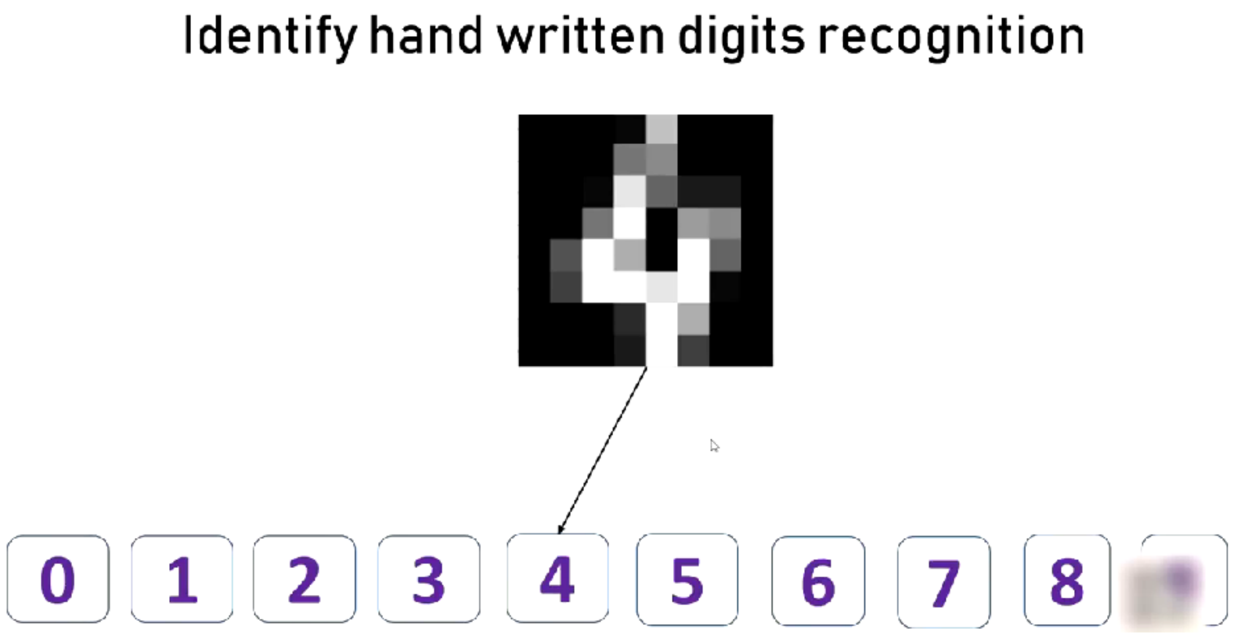
Logistic Regression: Multiclass Classification

Problem:

Hand written recognition is the problem we are trying to solve. We will use a training set with a lot of hand digit carachtors and then we build a model using logistic regression.



[for more information check this site (https://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html)](https://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html)

In [2]:

```
%matplotlib inline
import matplotlib.pyplot as plt

from sklearn.datasets import load_digits
```

In [3]:

```
# To load my training set
digits = load_digits()
```

In [5]:

```
# Explore what this training set contains?
dir(digits)
```

Out[5]:

```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names']
```

As you can see from the site above there are **1797** 8*8 samples.

In [8]:

```
#print the first sample
digits.data[0]
```

Out[8]:

```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
       10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

It's an array as such it is an 8*8 image but the image is represented as a 1 dimentional array. If you want to see this particular element you can use matplotlib.

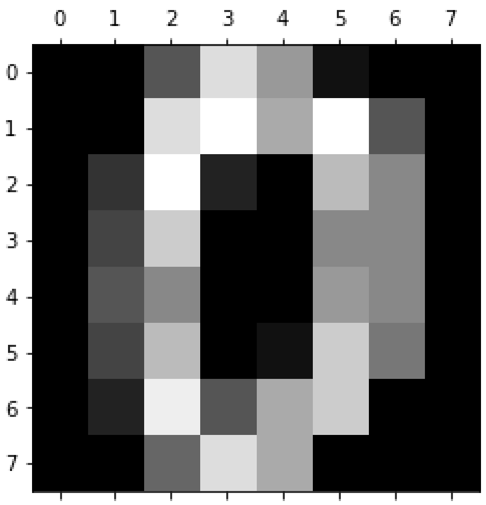
In [11]:

```
# Show an actual image for numeric data
plt.gray()
plt.matshow(digits.images[0])
```

Out[11]:

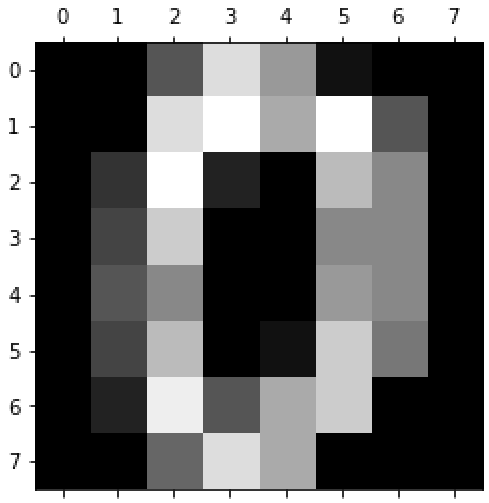
```
<matplotlib.image.AxesImage at 0xa7647c0>
```

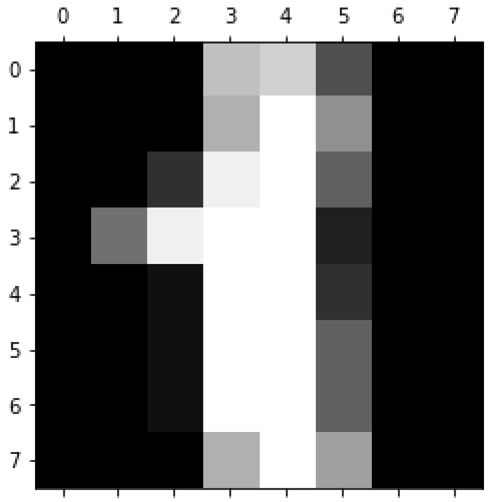
```
<Figure size 432x288 with 0 Axes>
```

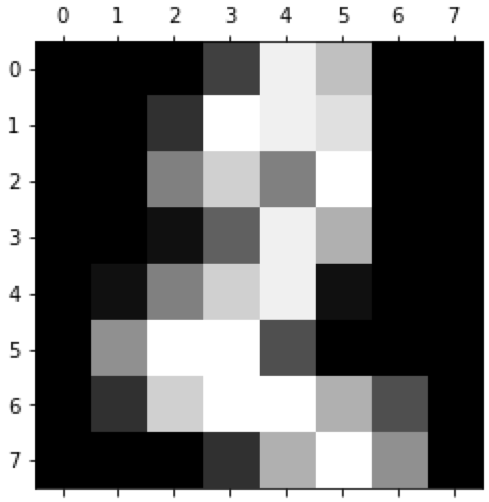


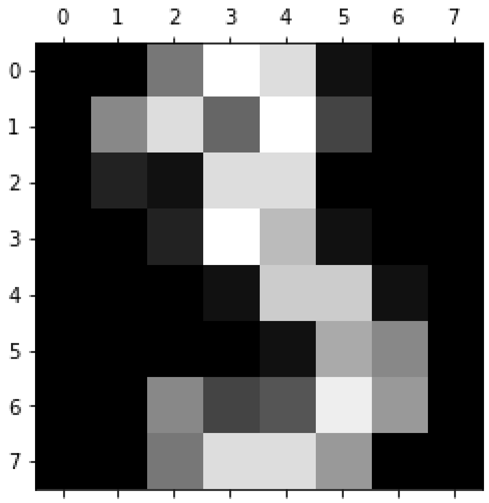
In [12]:
```
#Print the first 5 samples
plt.gray()
for i in range(5):
    plt.matshow(digits.images[i])
```

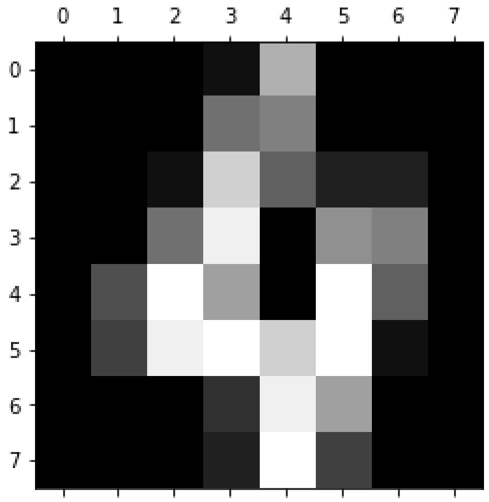
```
<Figure size 432x288 with 0 Axes>
```











In [14]:
```
dir(digits)
```

Out[14]:
```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names']
```

In [15]:
```
digits.target[0:5]
```

Out[15]:
```
array([0, 1, 2, 3, 4])
```

We can use **data** and **target** to train our model.

In [17]:
```
from sklearn.model_selection import train_test_split
```

In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(digits.data,digits.target, tes
```

In [19]:
```
len(X_train)
```

Out[19]:
```
1437
```

1437=0.8∗1797

Create logistic regression model

In [23]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [24]:
```
model.fit(X_train, y_train)
```

```
C:\Users\Ehsan\AppData\Roaming\Python\Python38\site-packages\sklearn\linear_mode
l\_logistic.py:762: 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 (https://scikit-l
earn.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-regressio
n (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressio
n)
  n_iter_i = _check_optimize_result(
```

Out[24]:
```
LogisticRegression()
```

Measure accuracy of our model

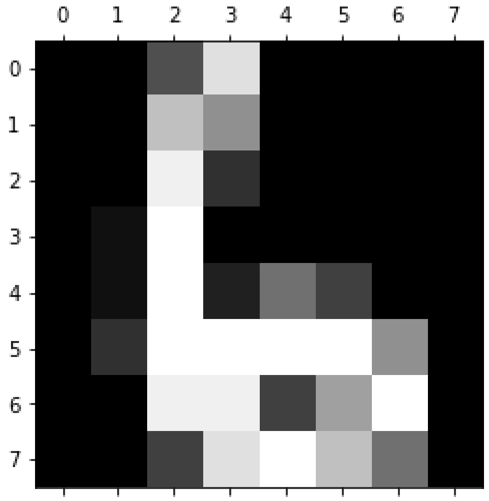
In [25]:
```
model.score(X_test, y_test)
```

Out[25]:
```
0.9694444444444444
```

Use the model and predict [67]

In [27]:
```
plt.matshow(digits.images[67])
```

Out[27]:
```
<matplotlib.image.AxesImage at 0xa8cd6d0>
```



In [28]:
```
digits.target[67]
```

Out[28]:
```
6
```

```
In [30]: model.predict(digits.data[[67]])
```

```
Out[30]: array([6])
```

Our model works fine.

```
In [31]: model.predict(digits.data[0:5])
```

```
Out[31]: array([0, 1, 2, 3, 4])
```

I want to know where my model fails in accuracy.

Confusion Matrix

```
In [32]: y_predicted = model.predict(X_test)
```

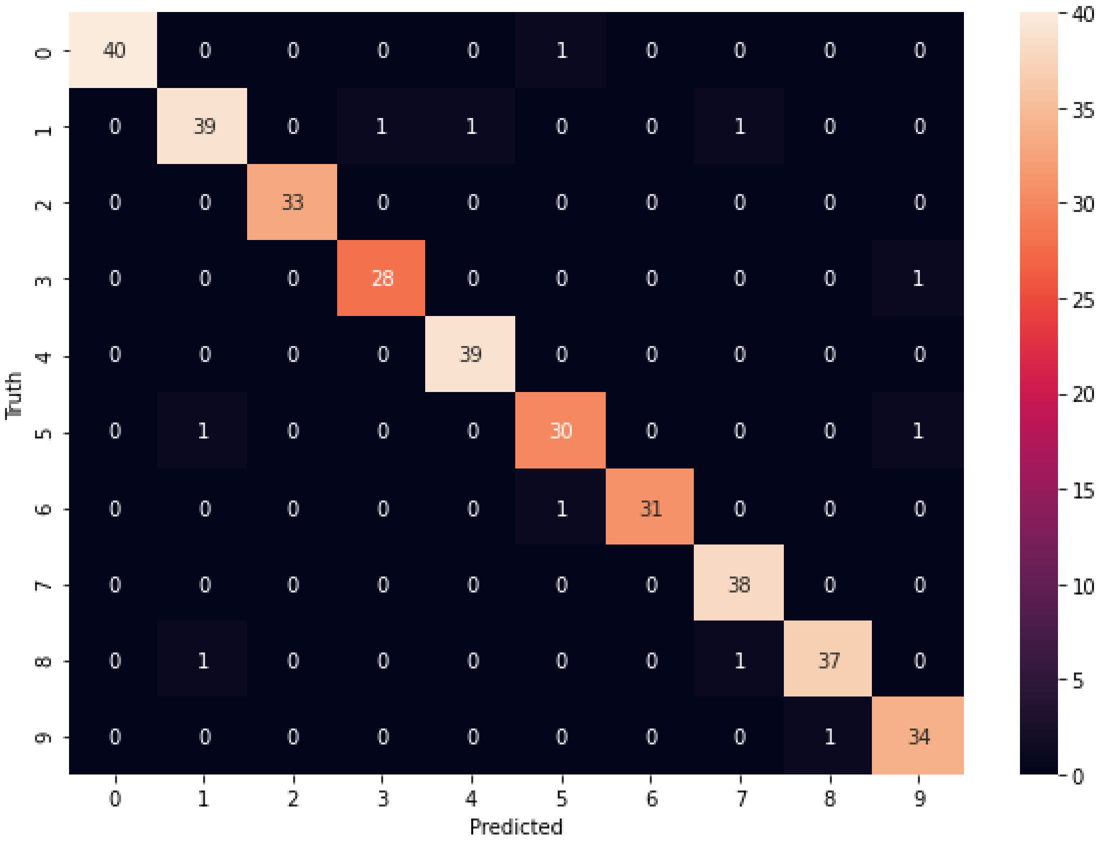
```
In [33]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_predicted)
         cm
```

```
Out[33]: array([[40,  0,  0,  0,  0,  1,  0,  0,  0,  0],
                [ 0, 39,  0,  1,  1,  0,  0,  1,  0,  0],
                [ 0,  0, 33,  0,  0,  0,  0,  0,  0,  0],
                [ 0,  0,  0, 28,  0,  0,  0,  0,  0,  1],
                [ 0,  0,  0,  0, 39,  0,  0,  0,  0,  0],
                [ 0,  1,  0,  0,  0, 30,  0,  0,  0,  1],
                [ 0,  0,  0,  0,  0,  1, 31,  0,  0,  0],
                [ 0,  0,  0,  0,  0,  0,  0, 38,  0,  0],
                [ 0,  1,  0,  0,  0,  0,  0,  1, 37,  0],
                [ 0,  0,  0,  0,  0,  0,  0,  0,  1, 34]], dtype=int64)
```

This is better visualize in matplotlib and seaborn.

```
In [34]: import seaborn as sn
         plt.figure(figsize = (10,7))
         sn.heatmap(cm, annot=True)
         plt.xlabel('Predicted')
         plt.ylabel('Truth')
```

```
Out[34]: Text(69.0, 0.5, 'Truth')
```



whenever the numbers other than main diogonal is not 0 means your model is inaccurate. For instance, the main diogonal is work fine 39 times the actual value and predicted one is 1. Number 5 is one time 1 and model inaccurate.

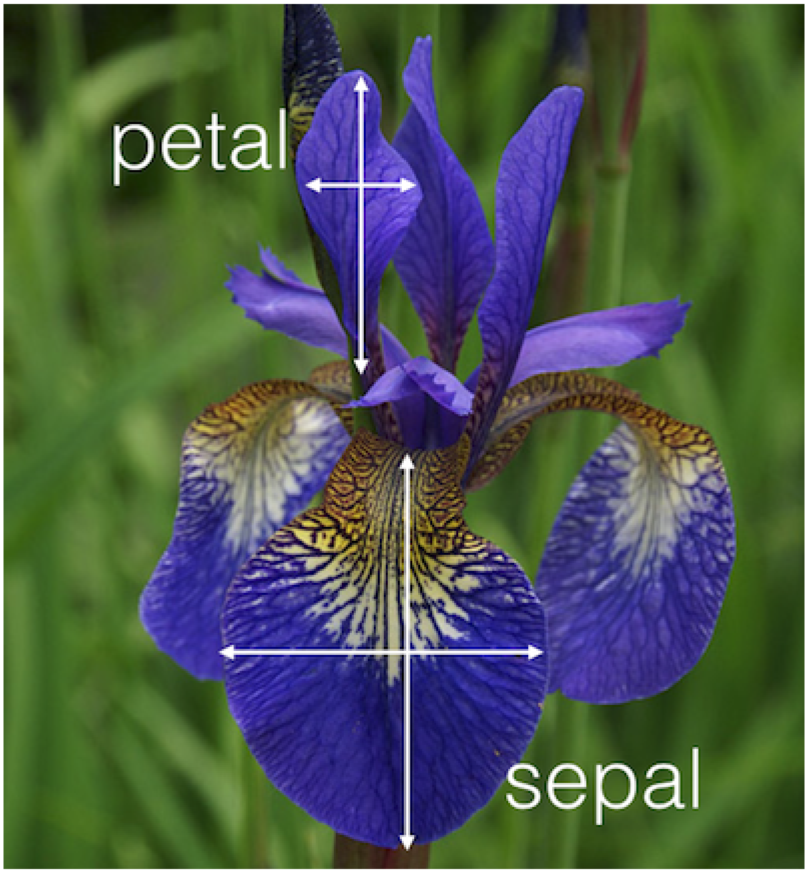
Exercise

Use sklearn.datasets iris flower dataset to train your model using logistic regression. You need to figure out accuracy of your model and use that to predict different samples in your test dataset. In iris dataset there are 150 samples containing following features,

1. Sepal Length
2. Sepal Width
3. Petal Length
4. Petal Width

Using above 4 features you will clasify a flower in one of the three categories,

1. Setosa
2. Versicolour
3. Virginica



Tip: Use **from sklearn.datasets import load_iris**

[clikc here for more information for exercise (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html)](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html)

[Also check this site (https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html)](https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html)

Solution:

Import dependencies

In [1]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

In [2]:

```
#Load the data set
data = sns.load_dataset("iris")
data.head()
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [3]:

```
# Prepare the training set
# X = feature values, all of the columns except the last column
X = data.iloc[:,:-1]
X
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [4]:

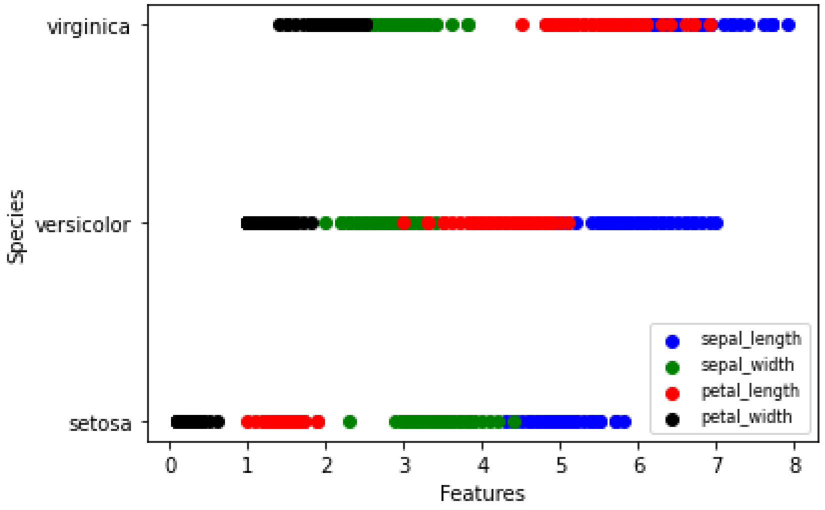
```
y=data.iloc[:,-1]
y
```

Out[4]:

```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
         ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

In [8]:

```
# Plot the relation of each feature with each species
plt.xlabel('Features')
plt.ylabel('Species')

pltX=data.loc[:,'sepal_length']
pltY=data.loc[:,'species']
plt.scatter(pltX,pltY,color='b',label='sepal_length')

pltX=data.loc[:,'sepal_width']
pltY=data.loc[:,'species']
plt.scatter(pltX,pltY,color='g',label='sepal_width')

pltX=data.loc[:,'petal_length']
pltY=data.loc[:,'species']
plt.scatter(pltX,pltY,color='r',label='petal_length')

pltX=data.loc[:,'petal_width']
pltY=data.loc[:,'species']
plt.scatter(pltX,pltY,color='black',label='petal_width')

plt.legend(loc=4,prop={'size':8})
plt.show()
```



In [9]:

```
# Split the data into 80% training and 20% testing
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

In [11]:

```
# Train the model
model=LogisticRegression()
model.fit(X_train,y_train)
```

```
C:\Users\Ehsan\AppData\Roaming\Python\Python38\site-packages\sklearn\linear_mode
l\_logistic.py:762: 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 (https://scikit-l
earn.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-regressio
n (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressio
n)
  n_iter_i = _check_optimize_result(
```

Out[11]:

```
LogisticRegression()
```

In [13]:

```
# Test the model
y_predict = model.predict(X_test)
y_predict
```

Out[13]:

```
array(['versicolor', 'setosa', 'virginica', 'versicolor', 'versicolor',
       'setosa', 'versicolor', 'virginica', 'versicolor', 'versicolor',
       'virginica', 'setosa', 'setosa', 'setosa', 'setosa', 'versicolor',
       'virginica', 'versicolor', 'versicolor', 'virginica', 'setosa',
       'virginica', 'setosa', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'setosa', 'setosa'], dtype=object)
```

In [14]:

```
y_test
```

Out[14]:

```
73     versicolor
18         setosa
118     virginica
78     versicolor
76     versicolor
31         setosa
64     versicolor
141     virginica
68     versicolor
82     versicolor
110     virginica
12         setosa
36         setosa
9          setosa
19         setosa
56     versicolor
104     virginica
69     versicolor
55     versicolor
132     virginica
29         setosa
127     virginica
26         setosa
128     virginica
131     virginica
145     virginica
108     virginica
143     virginica
45         setosa
30         setosa
Name: species, dtype: object
```

In [15]:

```
# Check precision, recall, f1-score
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

In [16]:

```
# The accuracy
print(accuracy_score(y_test,y_predict))
```

```
1.0
```

Date	Author
2021-09-09	**Ehsan Zia**