AIM :

THE AIM OF THE IRIS FLOWER CLASSIFICATION IS TO PREDICT FLOWERS BASED ON THEIR SPECIFIC FEATURES. IRIS FLOWER DATASET CONTAIN THREE CLASSES THAT ARE 1.VERSICOLOR, 2.VERGINICAA AND 3.SETOSA.

Dataset Resource:

1.Kaggle (https://www.kaggle.com/datasets)

2.Datahun.io (https://datahub.io/collections)

3.Github (https://github.com/topics/machine-learning-datasets)

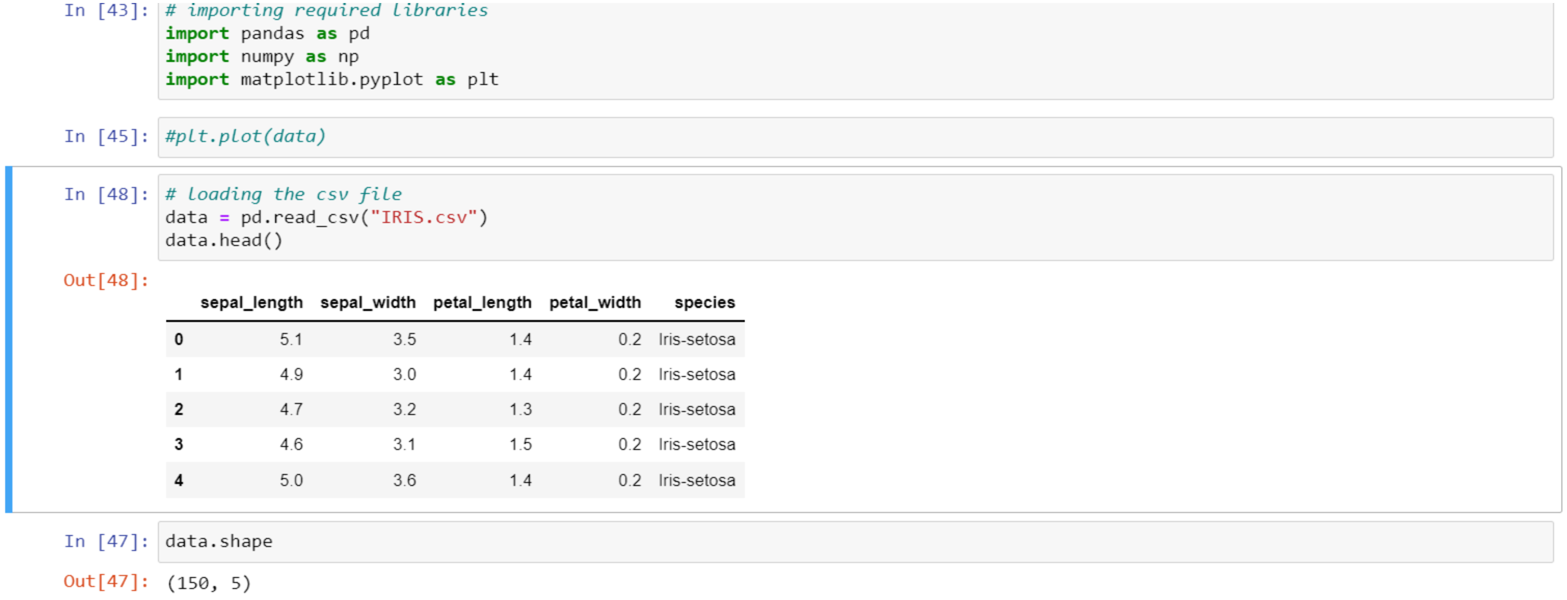
Size Of Data :

The dataset contains 150 rows and 5 columns.

Ans size of dataset is 5kb.

We take data from Kaggle

by using data.shape



```
In [43]: # importing required libraries
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt

In [45]: #plt.plot(data)

In [48]: # loading the csv file
         data = pd.read_csv("IRIS.csv")
         data.head()

Out[48]:
```

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

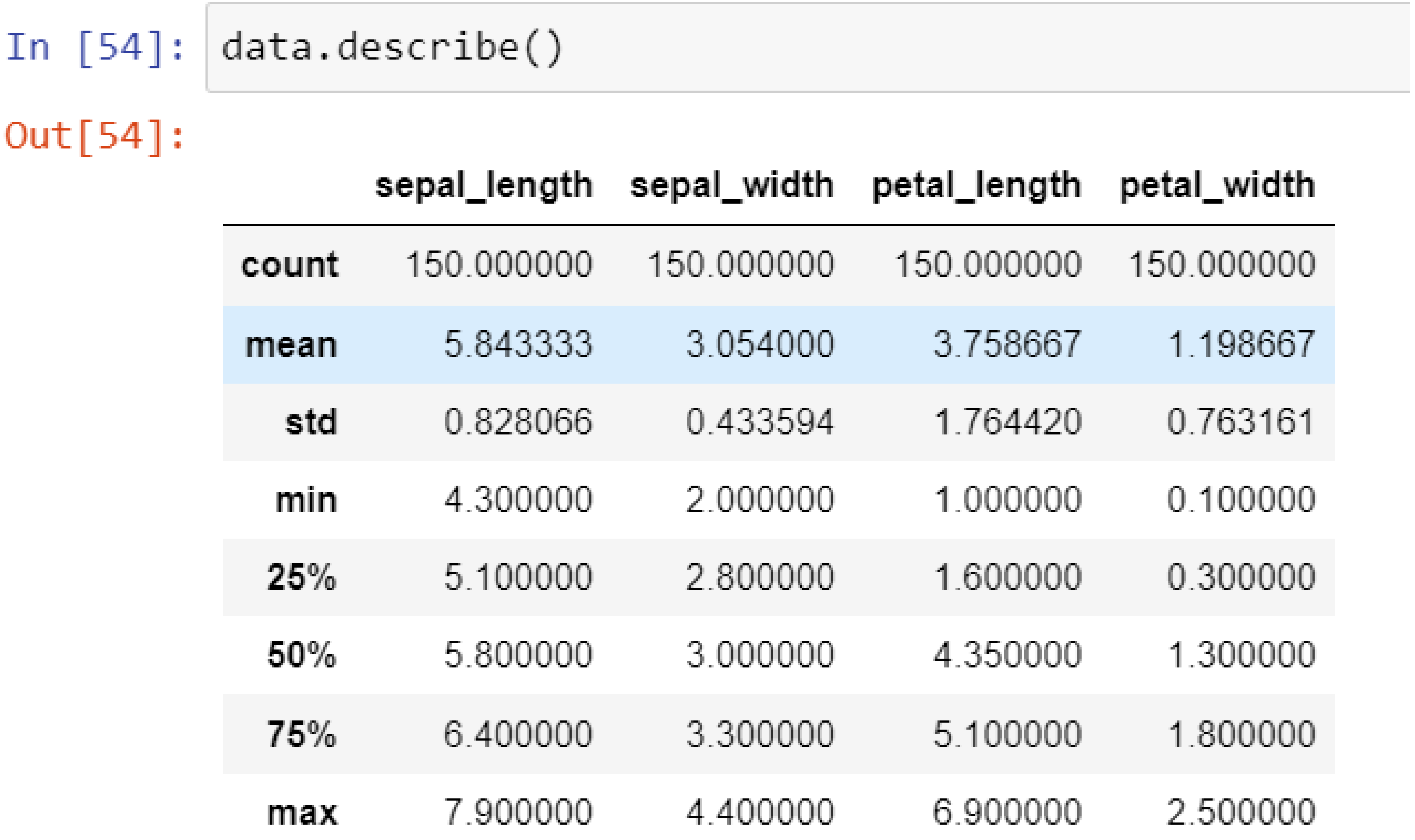
```
In [47]: data.shape

Out[47]: (150, 5)
```

Exploratory data Analysis :

EDA helps to identify errors in dataset with the help of some functions . And also visualize the data with the help of graph, pie and bar chart whenever its needed .

Mainly in EDA the shape of dataset is checked ,Null values,
Unique values,
information about data (info()).

Describe() tells us about count of each columns,min,max,mean and percentile as shown below

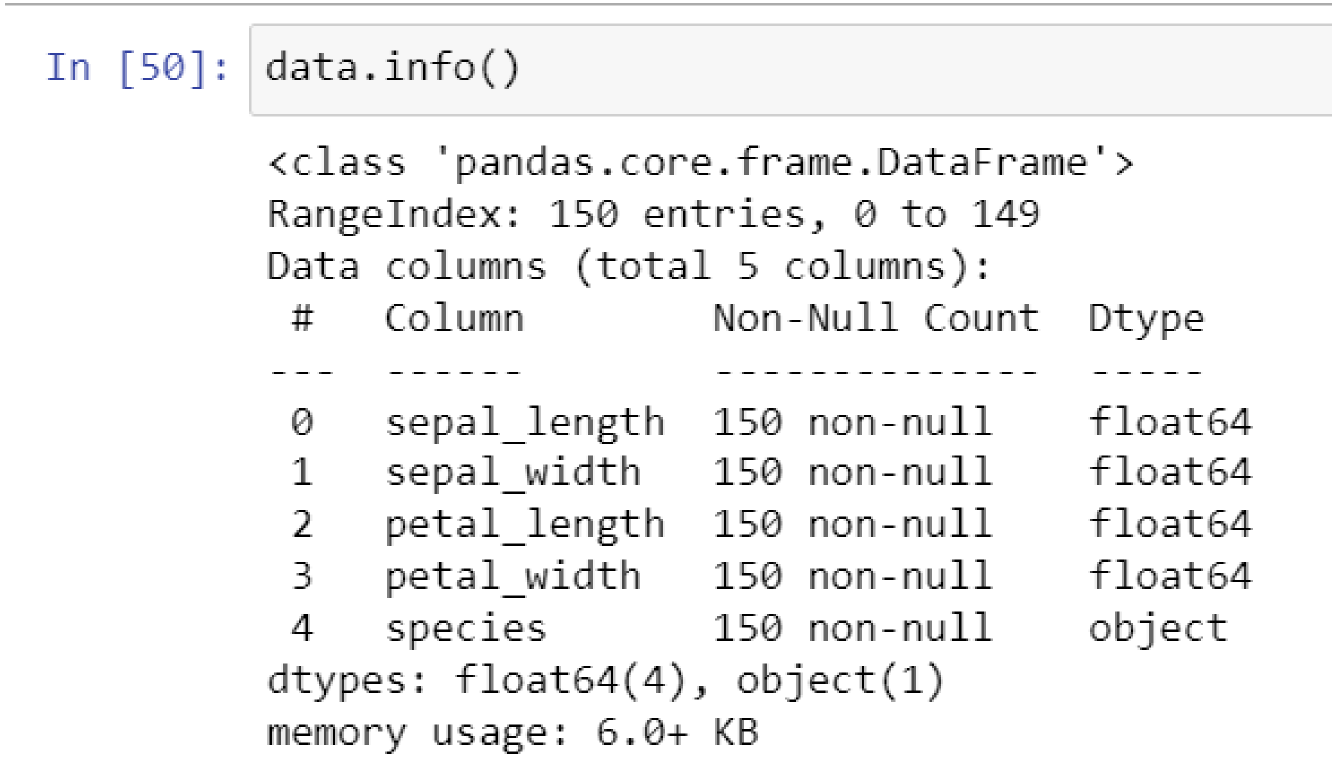


```
In [54]: data.describe()
```

Out[54]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

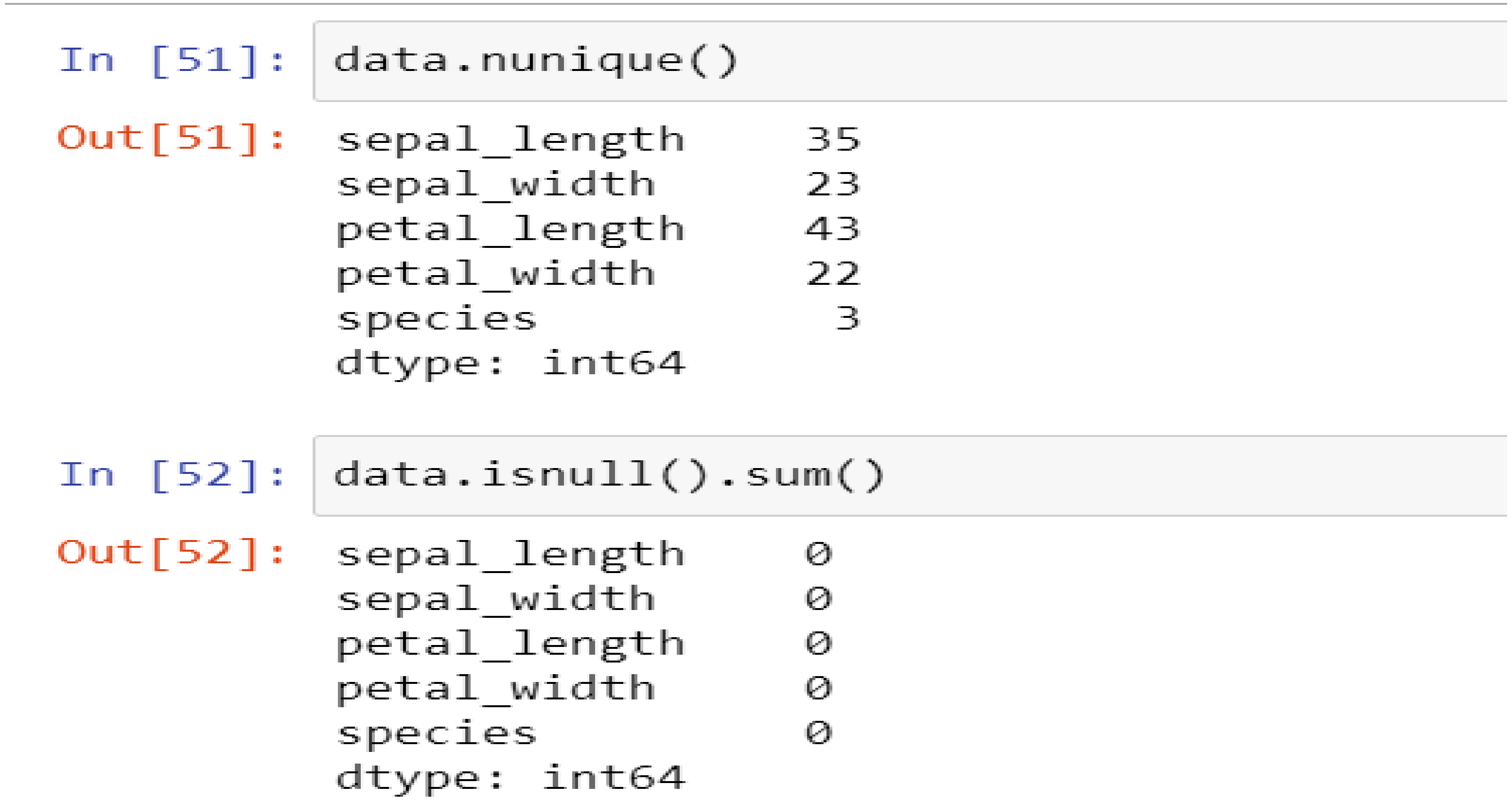
Info() tells about information about dataset means datatype of a columns either it is int or float etc.



```
In [50]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Isnull() tells tells how many null are there in datset and **nunique()** tells numbers of unique value for each columns.



```
In [51]: data.nunique()
```

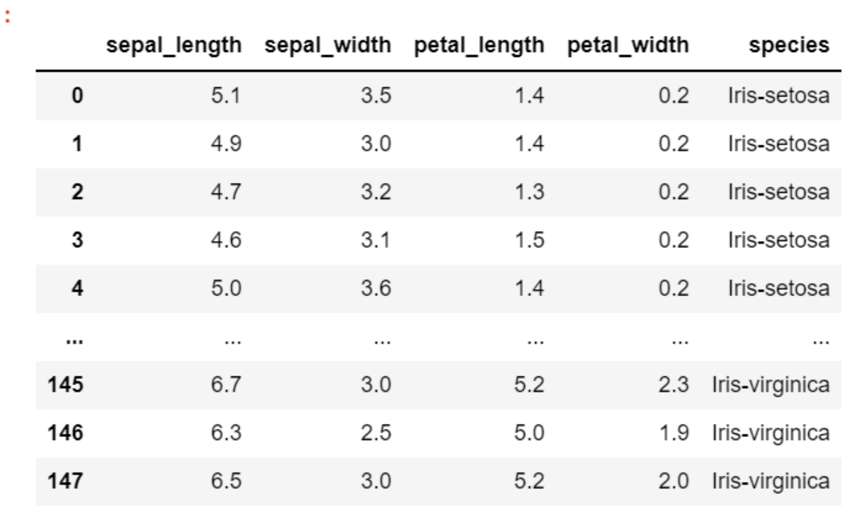
```
Out[51]: sepal_length    35
         sepal_width     23
         petal_length    43
         petal_width     22
         species          3
         dtype: int64
```

```
In [52]: data.isnull().sum()
```

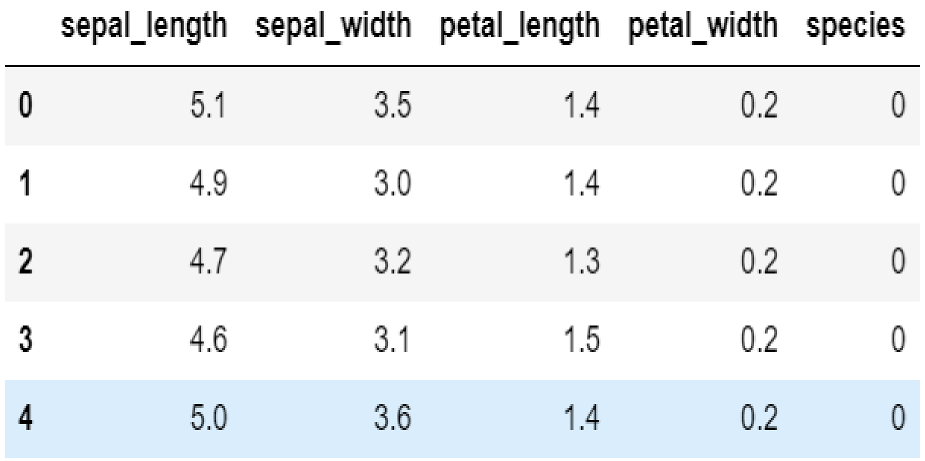
```
Out[52]: sepal_length    0
         sepal_width     0
         petal_length    0
         petal_width     0
         species         0
         dtype: int64
```

Label Encoding :

Label encoding is used for converting label into numeric form or into the machine readable form.



	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica



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

Before label encoding

after label encoding

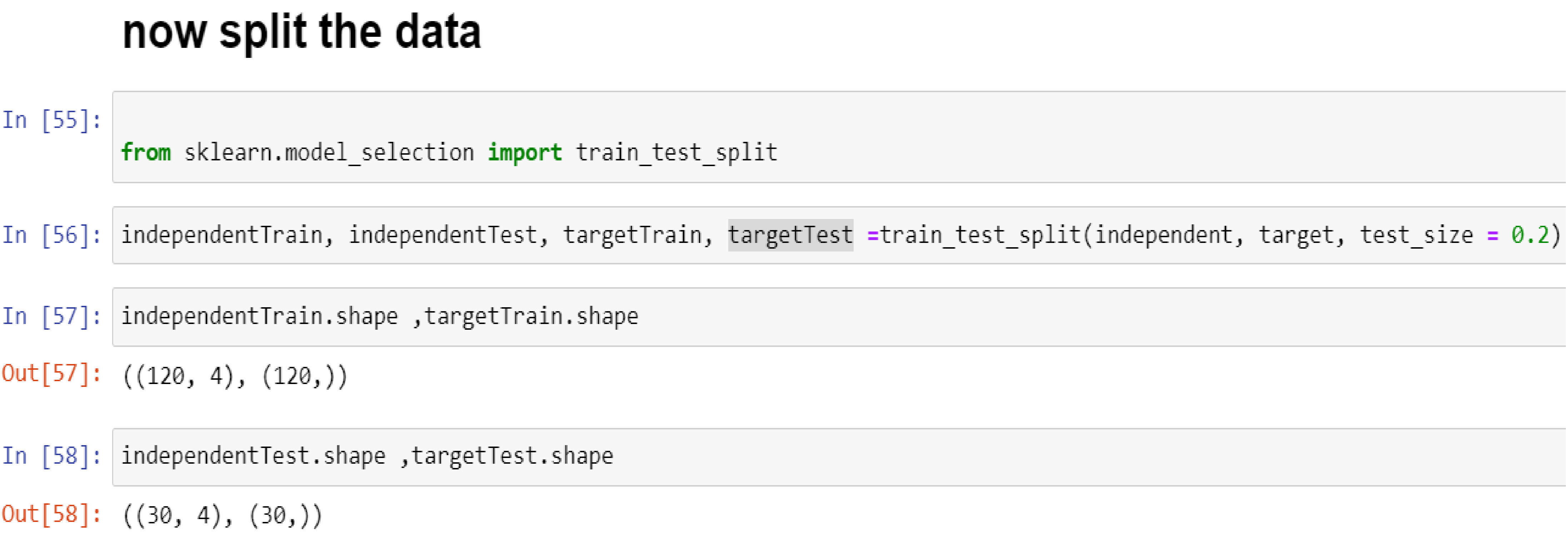
Training and testing data :

Generally data used for the training and testing is 80 % and 20%.
its choice of developer how much data give to training and how much data give to testing .

Here we split data for training and testing is 80% and 20%
the splitted data is:

```
        independentTrain = (120,4)
        independentTest = (30,1)


          targetTrain  = (120,4)
          targetTest   =  (30,1)
```

Splitted data:



now split the data

```
In [55]: 
         from sklearn.model_selection import train_test_split

In [56]: independentTrain, independentTest, targetTrain, targetTest =train_test_split(independent, target, test_size = 0.2)

In [57]: independentTrain.shape ,targetTrain.shape

Out[57]: ((120, 4), (120,))

In [58]: independentTest.shape ,targetTest.shape

Out[58]: ((30, 4), (30,))
```

Model Selection :

“ Model selection Is depends on data and understanding of data.”

how we decide proper model for data ,which model should be used or it will be appropriate for that data**.**

‘If we know the out put of the model Is a number then it is a regression problem.’

‘If we know the out put of the model Is a class that means it is a classification problem.’

Model used :

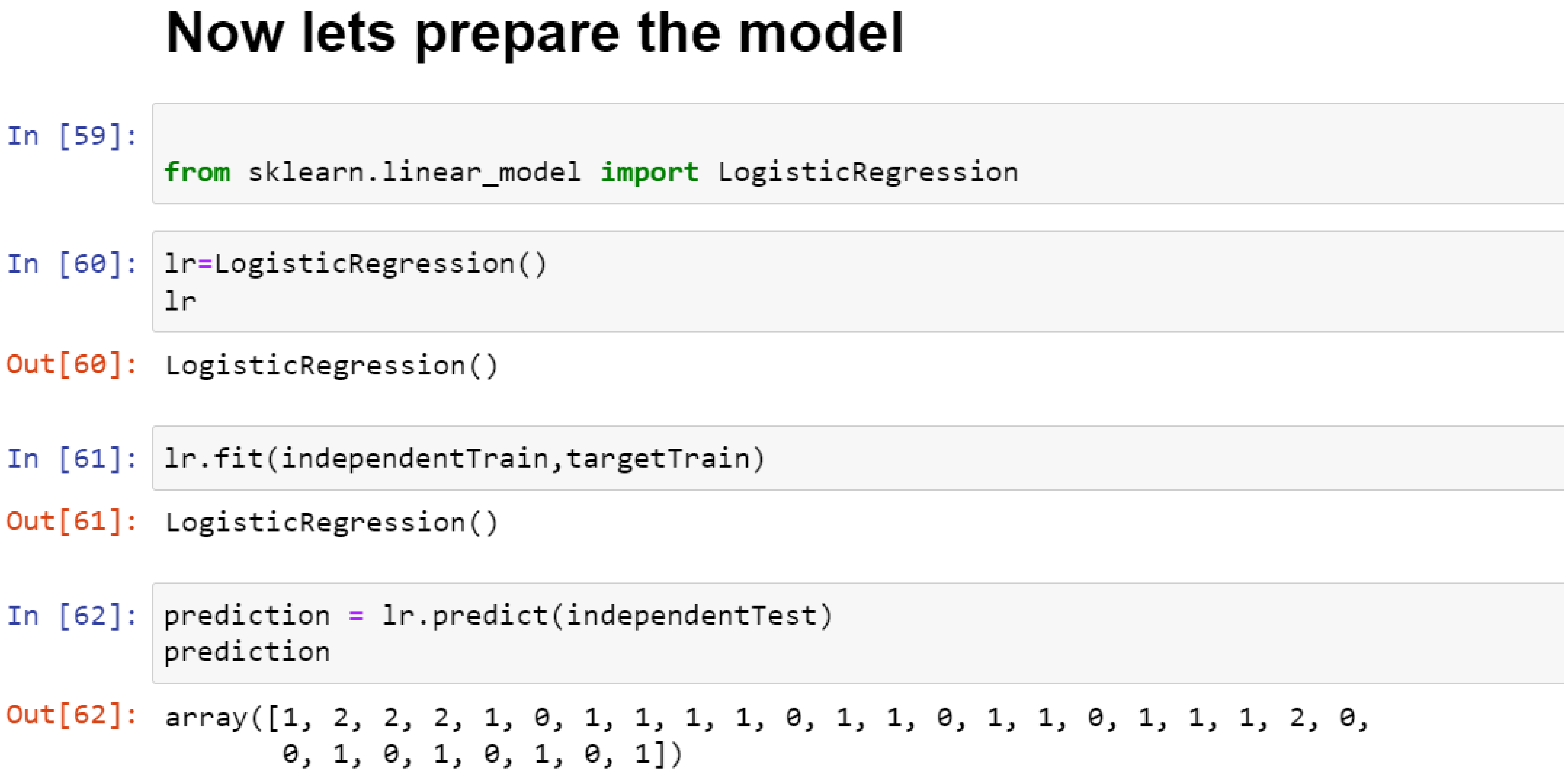
In this Project we know the output of the dataset i.e. class of a flower as there are three classes 1.versicolor, setosa, verginica.

Output is class so the model used is on the basis of classification :

As there are **6 algorithms 1.logisticRegression(sigmoid), 2.decision tree, 3.Random Forest, 4.SVM, 5.KNN, 6. Naïve Bayes**

For this project we are using the logistic Regression.

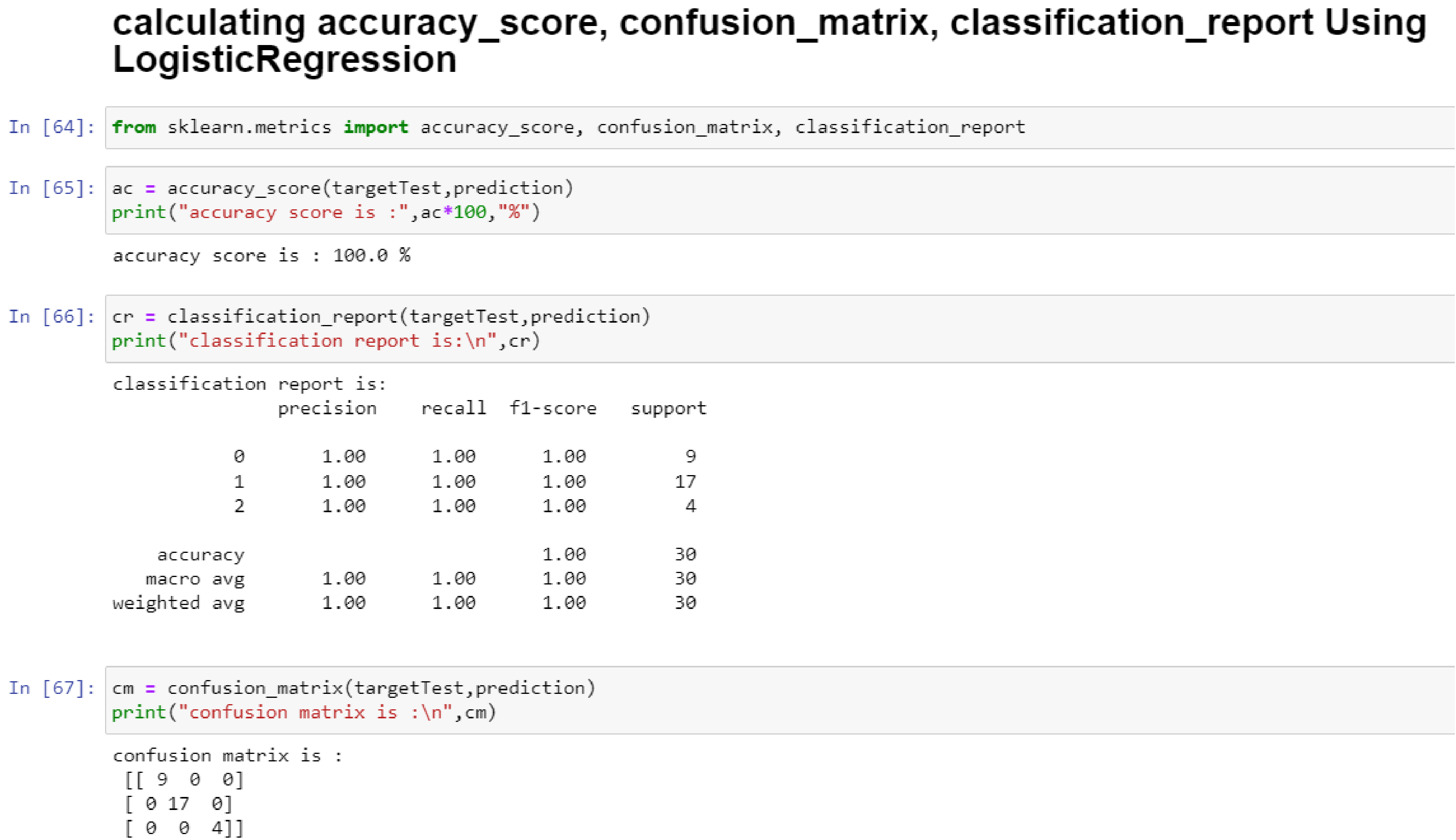
Here we use Logistic Regression model



Now lets prepare the model

```
In [59]: 
         from sklearn.linear_model import LogisticRegression

In [60]: lr=LogisticRegression()
         lr

Out[60]: LogisticRegression()

In [61]: lr.fit(independentTrain,targetTrain)

Out[61]: LogisticRegression()

In [62]: prediction = lr.predict(independentTest)
         prediction

Out[62]: array([1, 2, 2, 2, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 2, 0,
                0, 1, 0, 1, 0, 1, 0, 1])
```

As we did the prediction using **logistic regression** lets check it how accurate it is.



calculating accuracy_score, confusion_matrix, classification_report Using LogisticRegression

In [64]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [65]:
```
ac = accuracy_score(targetTest,prediction)
print("accuracy score is :",ac*100,"%")
```

```
accuracy score is : 100.0 %
```

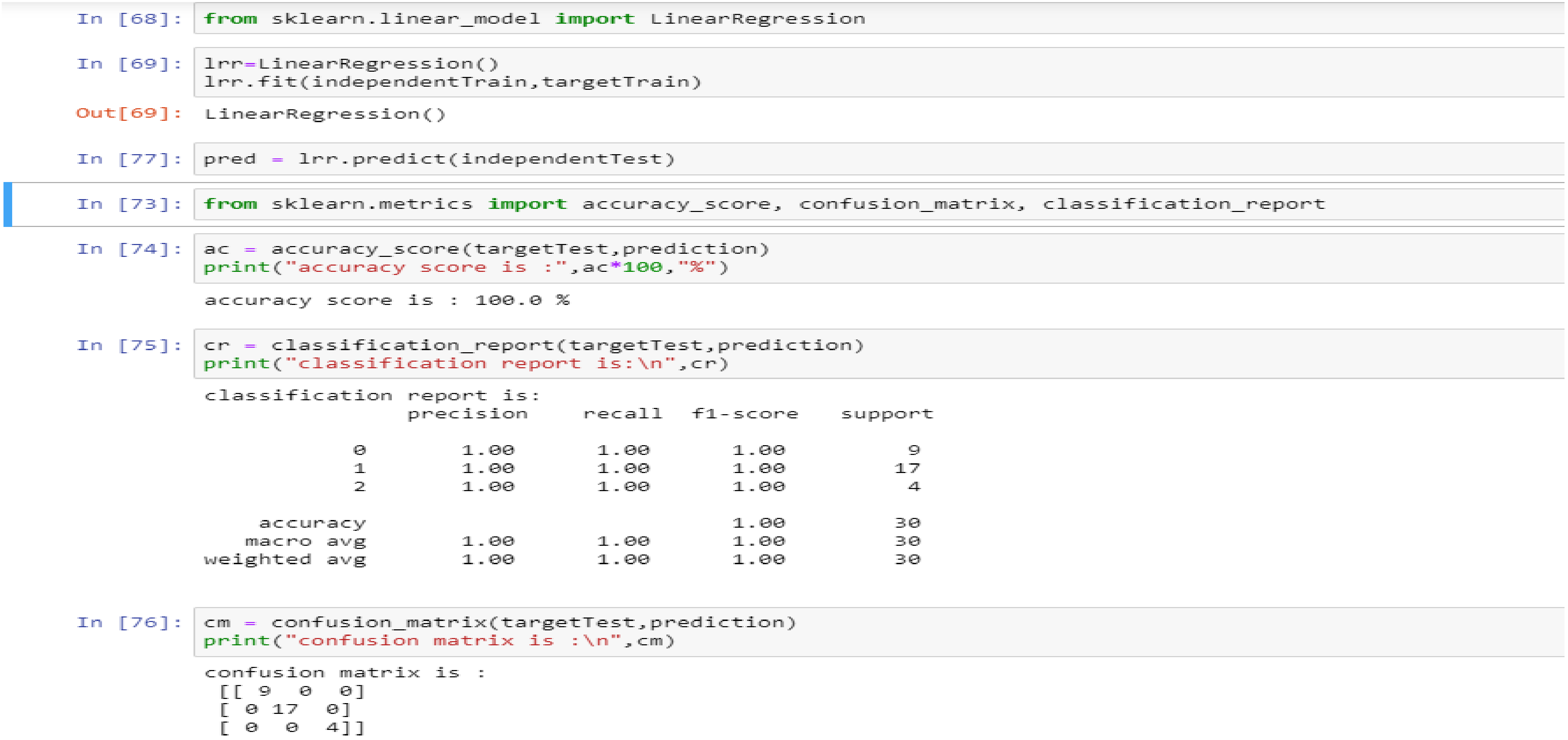
In [66]:
```
cr = classification_report(targetTest,prediction)
print("classification report is:\n",cr)
```

```
classification report is:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       1.00      1.00      1.00        17
           2       1.00      1.00      1.00         4

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

In [67]:
```
cm = confusion_matrix(targetTest,prediction)
print("confusion matrix is :\n",cm)
```

```
confusion matrix is :
 [[ 9  0  0]
 [ 0 17  0]
 [ 0  0  4]]
```

As we did the prediction using **Rinear Regression** lets check it how accurate it is.



```
In [68]: from sklearn.linear_model import LinearRegression

In [69]: lrr=LinearRegression()
         lrr.fit(independentTrain,targetTrain)

Out[69]: LinearRegression()

In [77]: pred = lrr.predict(independentTest)

In [73]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

In [74]: ac = accuracy_score(targetTest,prediction)
         print("accuracy score is :",ac*100,"%")

         accuracy score is : 100.0 %

In [75]: cr = classification_report(targetTest,prediction)
         print("classification report is:\n",cr)

         classification report is:
                        precision    recall  f1-score   support

                    0       1.00      1.00      1.00         9
                    1       1.00      1.00      1.00        17
                    2       1.00      1.00      1.00         4

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


In [76]: cm = confusion_matrix(targetTest,prediction)
         print("confusion matrix is :\n",cm)

         confusion matrix is :
          [[ 9  0  0]
          [ 0 17  0]
          [ 0  0  4]]
```

- **Conclusion:**
  Accuracy of both model i.e *Logistic Regression And linear Regression* is same and confusion matrix is also same. if the upper and lower diagonal element are zero and diagonal element are positive means accuracy is 100% .

  Every time model give different result as we got 100% accuracy sometimes it may be 99% or 98%.

Thank You

https://www.github.com/datta184/Machine-Learning-Projects/tree/main/iris