Application 6

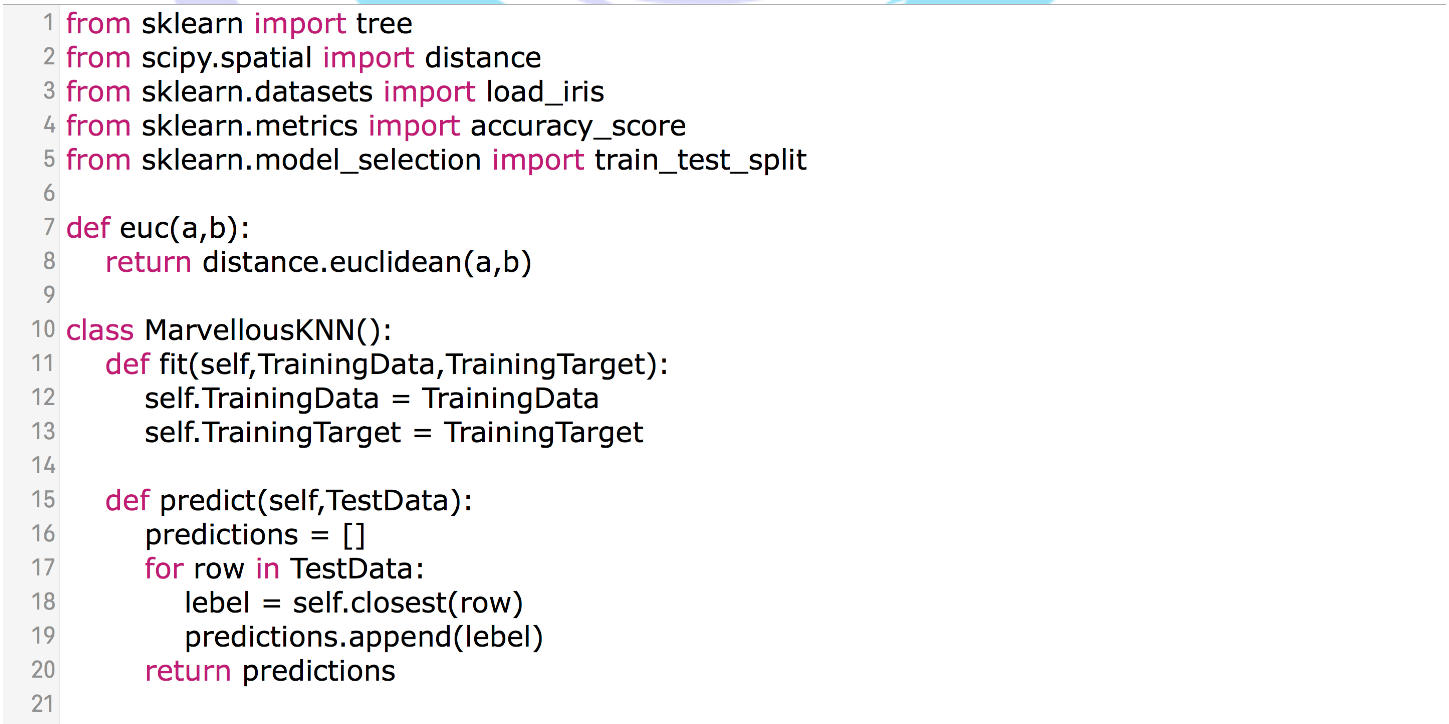
Supervised Machine Learning

User Defined K Nearest Neighbour Algorithm

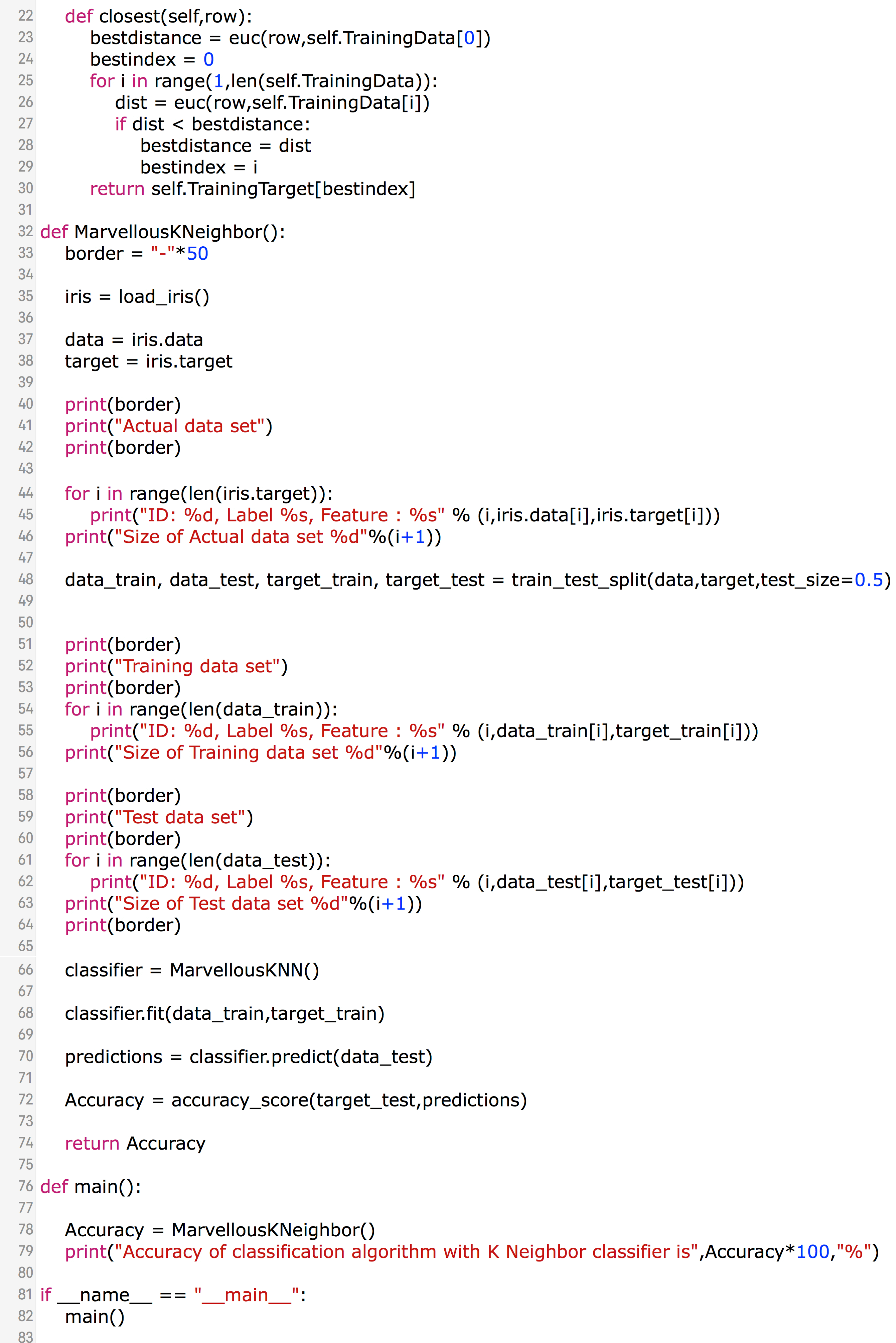
- In this application we create our own algorithm for classified machine learning.
- We create our own K Nearest Neighbour algorithm.
- For user defined algorithm we design one class named as MarvellousKNN.
- This class contains 3 methods as fit, predict, closest method.
- There is one naked method euc() which calculate distance between two points using Euclidean distance algorithm.
- fit() method initialises training data and its targets inside class.
- predict() method creates one array as prediction which stores shortest distance between all test data and training data elements.
- predict() method calls closest method which returns the shortest distance.

Consider below characteristics of Machine Learning Application :

Classifier :	**User Defined K Nearest Neighbour**
DataSet :	**Iris Dataset**
Features :	**Sepal Width, Sepal Length, Petal Width, Petal Length**
Labels :	**Versicolor, Setosa , Virginica**
Training Dataset :	**75 Entries**
Testing Dataset :	**75 Entries**

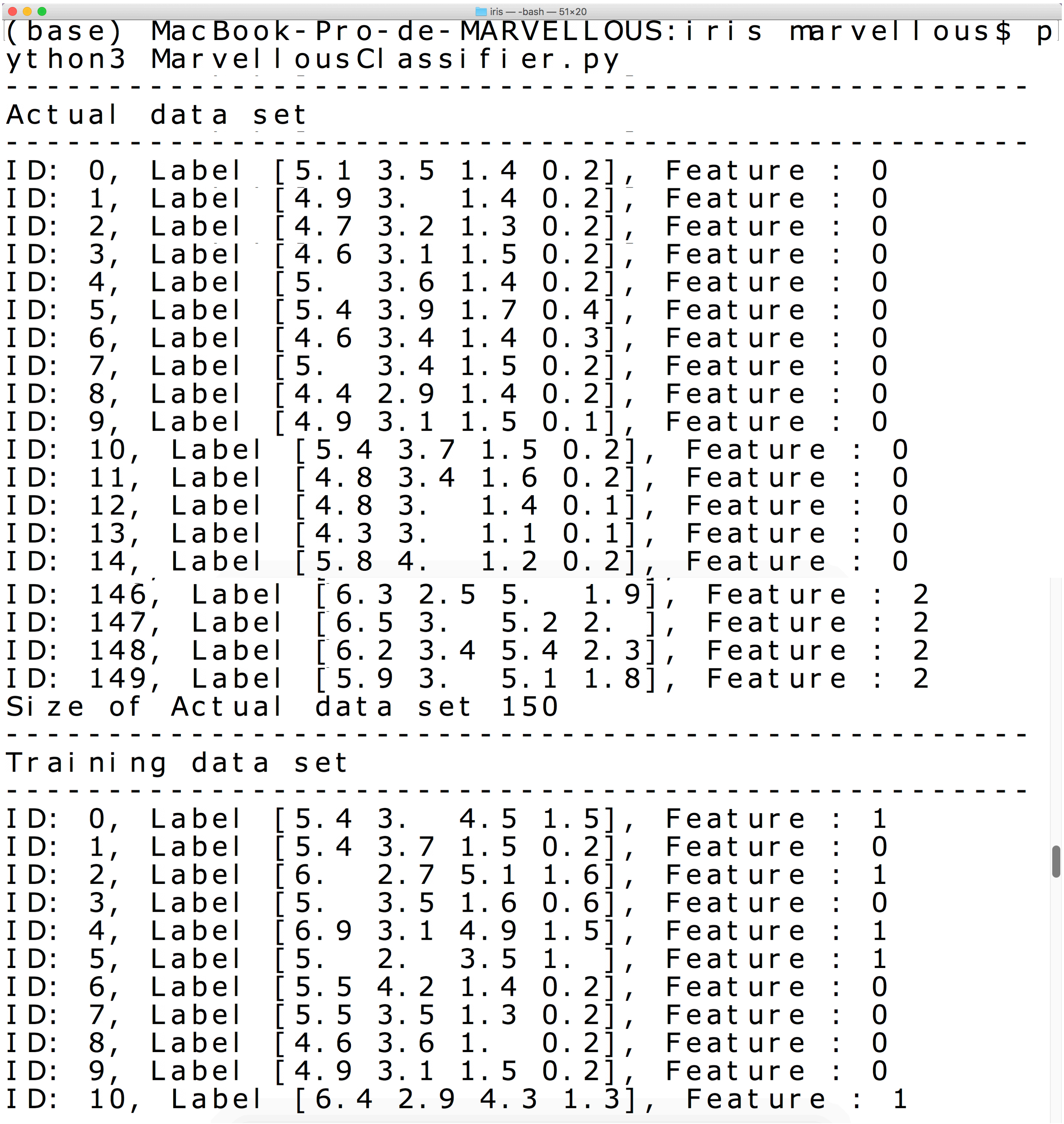


```
from sklearn import tree
from scipy.spatial import distance
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

def euc(a,b):
    return distance.euclidean(a,b)

class MarvellousKNN():
    def fit(self,TrainingData,TrainingTarget):
        self.TrainingData = TrainingData
        self.TrainingTarget = TrainingTarget

    def predict(self,TestData):
        predictions = []
        for row in TestData:
            lebel = self.closest(row)
            predictions.append(lebel)
        return predictions

```

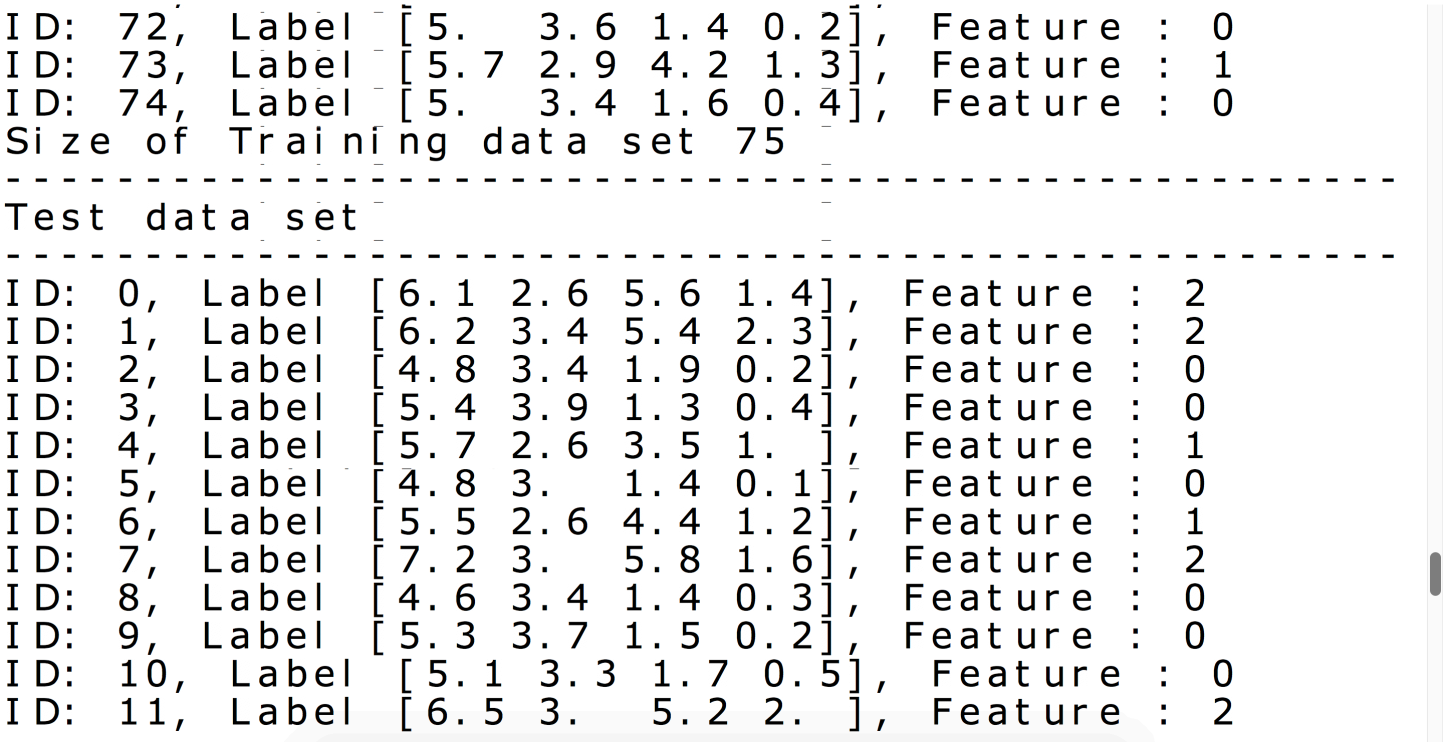


```python
    def closest(self,row):
        bestdistance = euc(row,self.TrainingData[0])
        bestindex = 0
        for i in range(1,len(self.TrainingData)):
            dist = euc(row,self.TrainingData[i])
            if dist < bestdistance:
                bestdistance = dist
                bestindex = i
        return self.TrainingTarget[bestindex]

def MarvellousKNeighbor():
    border = "-"*50

    iris = load_iris()

    data = iris.data
    target = iris.target

    print(border)
    print("Actual data set")
    print(border)

    for i in range(len(iris.target)):
        print("ID: %d, Label %s, Feature : %s" % (i,iris.data[i],iris.target[i]))
    print("Size of Actual data set %d"%(i+1))

    data_train, data_test, target_train, target_test = train_test_split(data,target,test_size=0.5)


    print(border)
    print("Training data set")
    print(border)
    for i in range(len(data_train)):
        print("ID: %d, Label %s, Feature : %s" % (i,data_train[i],target_train[i]))
    print("Size of Training data set %d"%(i+1))

    print(border)
    print("Test data set")
    print(border)
    for i in range(len(data_test)):
        print("ID: %d, Label %s, Feature : %s" % (i,data_test[i],target_test[i]))
    print("Size of Test data set %d"%(i+1))
    print(border)

    classifier = MarvellousKNN()

    classifier.fit(data_train,target_train)

    predictions = classifier.predict(data_test)

    Accuracy = accuracy_score(target_test,predictions)

    return Accuracy

def main():

    Accuracy = MarvellousKNeighbor()
    print("Accuracy of classification algorithm with K Neighbor classifier is",Accuracy*100,"%")

if __name__ == "__main__":
    main()
```

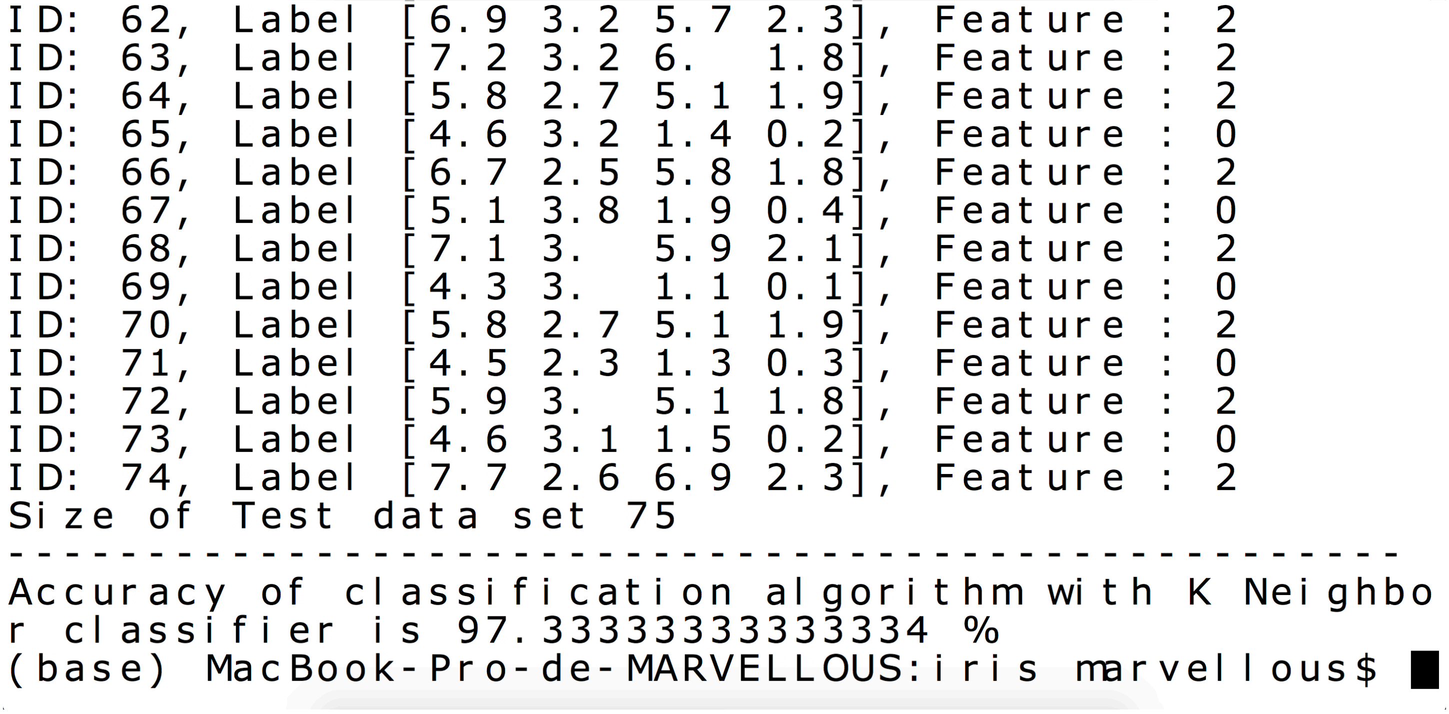
Output of above Application :



```
(base) MacBook-Pro-de-MARVELLOUS:iris marvellous$ p
ython3 MarvellousClassifier.py
--------------------------------------------------
Actual data set
--------------------------------------------------
ID: 0, Label [5.1 3.5 1.4 0.2], Feature : 0
ID: 1, Label [4.9 3.  1.4 0.2], Feature : 0
ID: 2, Label [4.7 3.2 1.3 0.2], Feature : 0
ID: 3, Label [4.6 3.1 1.5 0.2], Feature : 0
ID: 4, Label [5.  3.6 1.4 0.2], Feature : 0
ID: 5, Label [5.4 3.9 1.7 0.4], Feature : 0
ID: 6, Label [4.6 3.4 1.4 0.3], Feature : 0
ID: 7, Label [5.  3.4 1.5 0.2], Feature : 0
ID: 8, Label [4.4 2.9 1.4 0.2], Feature : 0
ID: 9, Label [4.9 3.1 1.5 0.1], Feature : 0
ID: 10, Label [5.4 3.7 1.5 0.2], Feature : 0
ID: 11, Label [4.8 3.4 1.6 0.2], Feature : 0
ID: 12, Label [4.8 3.  1.4 0.1], Feature : 0
ID: 13, Label [4.3 3.  1.1 0.1], Feature : 0
ID: 14, Label [5.8 4.  1.2 0.2], Feature : 0
ID: 146, Label [6.3 2.5 5.  1.9], Feature : 2
ID: 147, Label [6.5 3.  5.2 2. ], Feature : 2
ID: 148, Label [6.2 3.4 5.4 2.3], Feature : 2
ID: 149, Label [5.9 3.  5.1 1.8], Feature : 2
Size of Actual data set 150
--------------------------------------------------
Training data set
--------------------------------------------------
ID: 0, Label [5.4 3.  4.5 1.5], Feature : 1
ID: 1, Label [5.4 3.7 1.5 0.2], Feature : 0
ID: 2, Label [6.  2.7 5.1 1.6], Feature : 1
ID: 3, Label [5.  3.5 1.6 0.6], Feature : 0
ID: 4, Label [6.9 3.1 4.9 1.5], Feature : 1
ID: 5, Label [5.  2.  3.5 1. ], Feature : 1
ID: 6, Label [5.5 4.2 1.4 0.2], Feature : 0
ID: 7, Label [5.5 3.5 1.3 0.2], Feature : 0
ID: 8, Label [4.6 3.6 1.  0.2], Feature : 0
ID: 9, Label [4.9 3.1 1.5 0.2], Feature : 0
ID: 10, Label [6.4 2.9 4.3 1.3], Feature : 1
```



```
ID: 72, Label [5.  3.6 1.4 0.2], Feature : 0
ID: 73, Label [5.7 2.9 4.2 1.3], Feature : 1
ID: 74, Label [5.  3.4 1.6 0.4], Feature : 0
Size of Training data set 75
--------------------------------------------------
Test data set
--------------------------------------------------
ID: 0, Label [6.1 2.6 5.6 1.4], Feature : 2
ID: 1, Label [6.2 3.4 5.4 2.3], Feature : 2
ID: 2, Label [4.8 3.4 1.9 0.2], Feature : 0
ID: 3, Label [5.4 3.9 1.3 0.4], Feature : 0
ID: 4, Label [5.7 2.6 3.5 1. ], Feature : 1
ID: 5, Label [4.8 3.  1.4 0.1], Feature : 0
ID: 6, Label [5.5 2.6 4.4 1.2], Feature : 1
ID: 7, Label [7.2 3.  5.8 1.6], Feature : 2
ID: 8, Label [4.6 3.4 1.4 0.3], Feature : 0
ID: 9, Label [5.3 3.7 1.5 0.2], Feature : 0
ID: 10, Label [5.1 3.3 1.7 0.5], Feature : 0
ID: 11, Label [6.5 3.  5.2 2. ], Feature : 2
```



```
ID: 62, Label [6.9 3.2 5.7 2.3], Feature : 2
ID: 63, Label [7.2 3.2 6.  1.8], Feature : 2
ID: 64, Label [5.8 2.7 5.1 1.9], Feature : 2
ID: 65, Label [4.6 3.2 1.4 0.2], Feature : 0
ID: 66, Label [6.7 2.5 5.8 1.8], Feature : 2
ID: 67, Label [5.1 3.8 1.9 0.4], Feature : 0
ID: 68, Label [7.1 3.  5.9 2.1], Feature : 2
ID: 69, Label [4.3 3.  1.1 0.1], Feature : 0
ID: 70, Label [5.8 2.7 5.1 1.9], Feature : 2
ID: 71, Label [4.5 2.3 1.3 0.3], Feature : 0
ID: 72, Label [5.9 3.  5.1 1.8], Feature : 2
ID: 73, Label [4.6 3.1 1.5 0.2], Feature : 0
ID: 74, Label [7.7 2.6 6.9 2.3], Feature : 2
Size of Test data set 75
--------------------------------------------------
Accuracy of classification algorithm with K Neighbo
r classifier is 97.33333333333334 %
(base) MacBook-Pro-de-MARVELLOUS:iris marvellous$ █
```