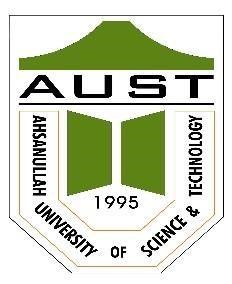
AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY DHAKA-1208, BANGLADESH.



Department of Computer Science and Engineering

Spring 2019

# Program: Bachelor of Science in Computer Science and Engineering

Course No: CSE 4108

Course Title: Artificial Intelligence Lab

Assignment No: 05

Date of Submission: 30/09/2019

Submitted to

Dr. S.M. Abdullah Al-Mamun

Professor, Department of CSE, AUST.

Md. Siam Ansary

Adjunct Faculty, Department of CSE, AUST.

# Submitted By

# Name : Robiul Hasan Nowshad

# Student ID : 16.01.04.061

# Lab Group : B1

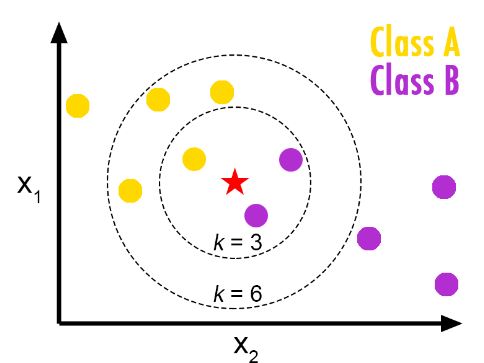
|  |  |
| --- | --- |
|  |  |
| **0 |** P a g e |  |

* **Question:** Implement k-Nearest Neighbour Classifier without using Scikit-learn.

**KNN:** k Nearest Neighbours is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure.

K= Number of Nearest Neighbours

**How it Works**

****

Let’s assume we have 2 types of data class A and class B. Now we got another new data the star now we have to predict which class it belongs. Now take k=3, so we take 3 nearest neighbours of star. If no of data of class A is more than B then it belongs to class A, otherwise B. So simple.

**Dataset:** The dataset we use is HeadBrain.csv

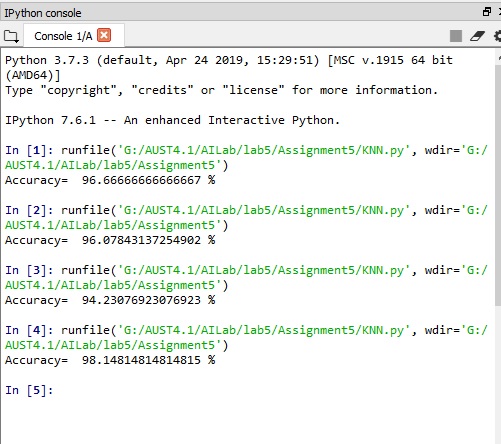
From [**https://www.kaggle.com/saarthaksangam/headbrain**](https://www.kaggle.com/saarthaksangam/headbrain)

**Source Code:**

1. # -\*- coding: utf-8 -\*-
2. """
3. Created on Mon Sep 30 03:12:40 2019
5. @author: nowshad
6. """
8. **import** csv
9. **import** random
10. **import** math
11. **import** operator
13. **def** loadDataset(filename, split,trainingSet=[],testSet=[]):
14. with open(filename, 'r') as csvfile:
15. lines=csv.reader(csvfile)
16. dataset=list(lines)
17. **for** x **in** range(len(dataset)-1):
18. **for** y **in** range(4):
19. dataset[x][y]=float(dataset[x][y])
20. **if** random.random()<split:
21. trainingSet.append(dataset[x])
22. **else**:
23. testSet.append(dataset[x])
25. **def** euclideanDistance(instance1,instance2, length):
26. distance=0
27. **for** x **in** range(length):
28. distance +=pow((instance1[x]-instance2[x]),2)
29. **return** math.sqrt(distance)
31. **def** getNeighbours(trainingSet, testInstance, k):
32. distance=[]
33. length=len(testInstance)-1
34. **for** x **in** range(len(trainingSet)):
35. dist=euclideanDistance(testInstance,trainingSet[x], length)
36. distance.append((trainingSet[x],dist))
37. distance.sort(key=operator.itemgetter(1))
38. neighbors=[]
39. **for** x **in** range (k):
40. neighbors.append(distance[x][0])
41. **return** neighbors
43. **def** getResponse(neighbors):
44. classVotes={}
45. **for** x **in** range(len(neighbors)):
46. response=neighbors[x][-1]
47. **if** response **in** classVotes:
48. classVotes[response]+=1
49. **else**:
50. classVotes[response]=1
51. sortedVotes=sorted(classVotes.items(), key=operator.itemgetter(1), reverse=True)
52. **return** sortedVotes[0][0]
54. **def** getAccuracy(testSet, predictions):
55. correct=0
56. **for** x **in** range(len(testSet)):
57. **if** testSet[x][-1] == predictions[x]:
58. correct+=1
59. **return** (correct/float(len(testSet)))\*100.0
61. #main
62. with open(r'G:\AUST4.1\AILab\lab5\Assignment5\iris.data')as csvfile:
63. lines=csv.reader(csvfile)
64. trainingSet=[]
65. testSet=[]
66. split=0.67
67. loadDataset('iris.data',split,trainingSet,testSet)
68. predictions=[]
69. k=3
70. **for** x **in** range(len(testSet)):
71. neighbors=getNeighbours(trainingSet,testSet[x],k)
72. result=getResponse(neighbors)
73. predictions.append(result)
74. #print('Predicted=',result,', actual=',testSet[x][-1])
75. accuracy=getAccuracy(testSet,predictions)
76. **print**('Accuracy= ',accuracy,'%')

**Output:**

The max accuracy we got is 98.14%. the Screen Shot given bellow.

****

* **Question:** Implement Linear Regression Classifier without using Scikit-learn.

**Linear Regression:** Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent and independent variable.

**Mathematical Part,**

Y = αX + β

Y – Random variable (response, dependent)

X – Random variable (predictor, independent)

α, β - regression coefficients, that are to be learned

α = Σi=1:s (xi – x′) (yi- y′) / Σi=1:s (xi – x′)2 ,

β = y′ - αx′,

where x′ - average of x1, x2, … , xs ,

y′ = y1, y2, … , ys ,

given sample data points (x1, y1), (x2, y2), …, (xs, ys).

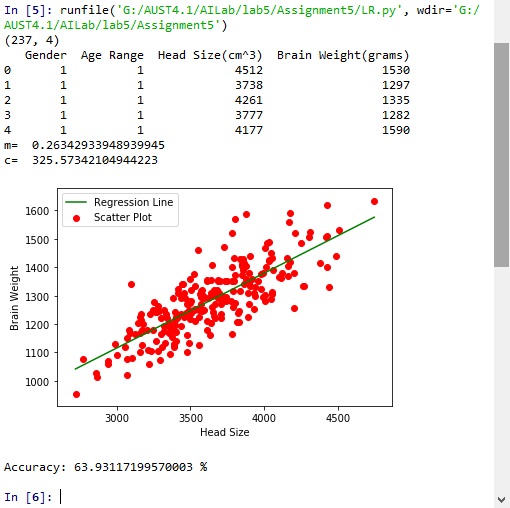
**Dataset:** The dataset we use is iris.data

From <https://www.kaggle.com/uciml/iris>

**Source Code:**

1. # -\*- coding: utf-8 -\*-
2. """
3. Created on Mon Sep 30 04:12:40 2019
5. @author: nowshad
6. """
7. **import** numpy as np
8. **import** matplotlib.pyplot as plt
9. **import** pandas as pd
11. dataset = pd.read\_csv('headbrain.csv')
12. **print**(dataset.shape)
13. **print**(dataset.head(5))
15. X = dataset['Head Size(cm^3)'].values
16. Y = dataset['Brain Weight(grams)'].values
17. mean\_x = np.mean(X)
18. mean\_y= np.mean(Y)
20. n=len(X)
22. numer=0
23. denom=0
24. **for** i **in** range(n):
25. numer += (X[i]-mean\_x)\*(Y[i]-mean\_y)
26. denom += (X[i]-mean\_x)\*\*2
28. m= numer/denom
29. c= mean\_y - (m\*mean\_x)
30. **print**("m= ",m)
31. **print**("c= ",c)
33. max\_x = np.max(X)+1
34. min\_x= np.min(X)-1
35. x=np.linspace(min\_x,max\_x,5)
36. y=m\*x+c
38. plt.plot(x,y,color='green',label='Regression Line')
39. plt.scatter(X,Y,color='red',label='Scatter Plot')
40. plt.xlabel('Head Size')
41. plt.ylabel('Brain Weight')
42. plt.legend()
43. plt.show()
45. err\_nm=0
46. err\_dn=0
47. **for** i **in** range(n):
48. y\_predict=m\*X[i]+c
49. err\_nm += (Y[i]-y\_predict) \*\*2
50. err\_dn += (Y[i]-mean\_y)\*\*2
51. Accuracy= (1-(err\_nm/err\_dn))\*100
52. **print**("\nAccuracy:",Accuracy,"%")

**Output:**

****

Accuracy is more than 50% and its good enough.