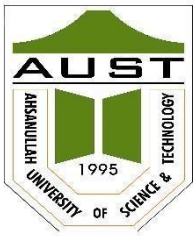
AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

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

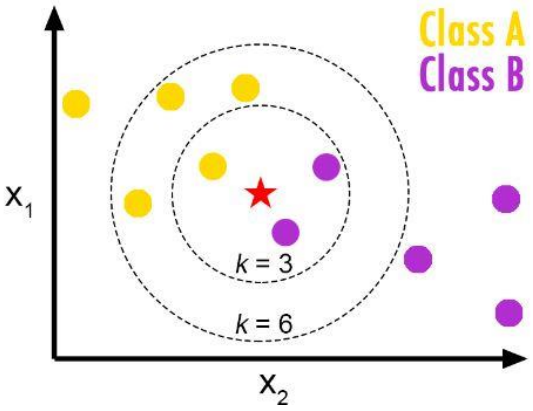
Lab Group : B1

- **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

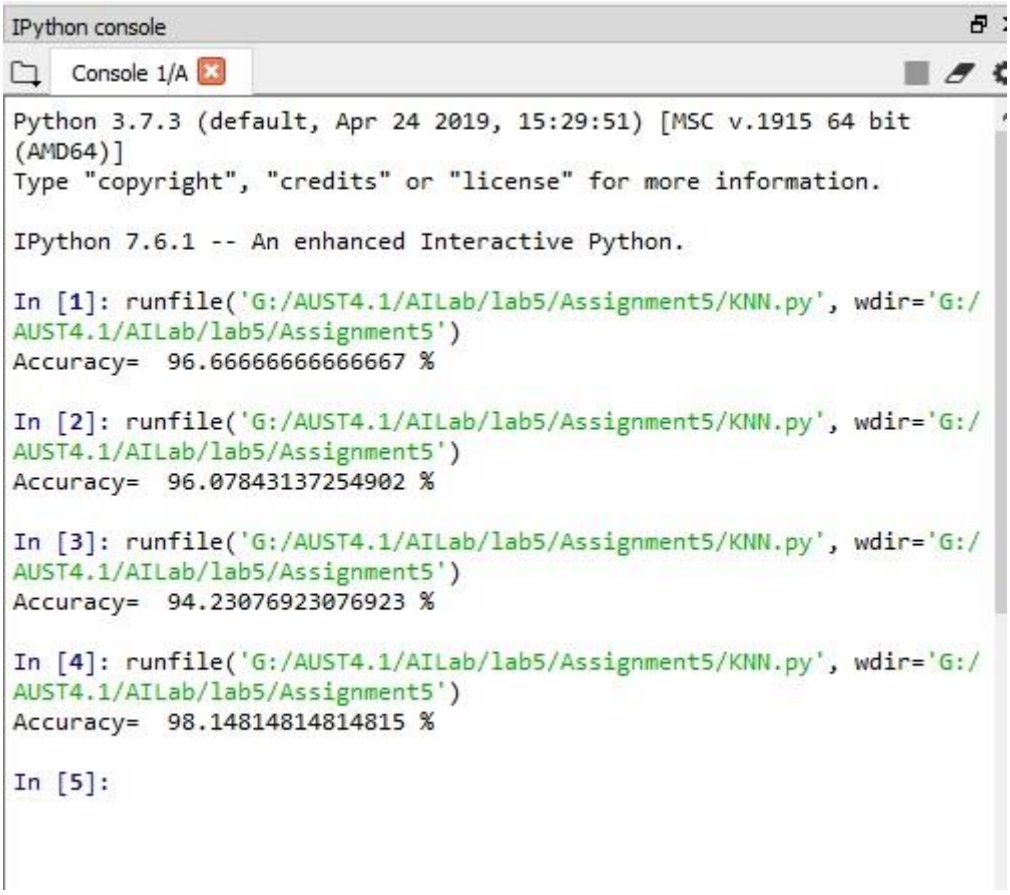
Source Code:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Sep 30 03:12:40 2019

@author: nowshad
"""

import csv
import random
import math
import operator

def loadDataset(filename, split,trainingSet=[],testSet=[]):
    with open(filename, 'r') as csvfile:
        lines=csv.reader(csvfile)
        dataset=list(lines)
        for x in range(len(dataset)-1):
            for y in range(4):
                dataset[x][y]=float(dataset[x][y])
            if random.random()<split:
                trainingSet.append(dataset[x])
            else:
                testSet.append(dataset[x])

def euclideanDistance(instance1,instance2, length):
    distance=0
```

```
    for x in range(length):
        distance +=pow((instance1[x]-instance2[x]),2)
    return math.sqrt(distance)

def getNeighbours(trainingSet, testInstance, k):
    distance=[]
    length=len(testInstance)-1
    for x in range(len(trainingSet)):
        dist=euclideanDistance(testInstance,trainingSet[x], length)
        distance.append((trainingSet[x],dist))
    distance.sort(key=operator.itemgetter(1))
    neighbors=[]
    for x in range (k):
        neighbors.append(distance[x][0])
    return neighbors

def getResponse(neighbors):
    classVotes={}
    for x in range(len(neighbors)):
        response=neighbors[x][-1]
        if response in classVotes:
            classVotes[response]+=1
        else:
            classVotes[response]=1
    sortedVotes=sorted(classVotes.items(), key=operator.itemgetter(1), reverse=True
)
    return sortedVotes[0][0]

def getAccuracy(testSet, predictions):
    correct=0
    for x in range(len(testSet)):
        if testSet[x][-1] == predictions[x]:
            correct+=1
    return (correct/float(len(testSet)))*100.0

#main
with open(r'G:\AUST4.1\AILab\lab5\Assignment5\iris.data')as csvfile:
    lines=csv.reader(csvfile)
trainingSet=[]
testSet=[]
split=0.67
loadDataset('iris.data',split,trainingSet,testSet)
predictions=[]
k=3
for x in range(len(testSet)):
    neighbors=getNeighbours(trainingSet,testSet[x],k)
    result=getResponse(neighbors)
    predictions.append(result)
    #print('Predicted=',result,', actual=',testSet[x][-1])
accuracy=getAccuracy(testSet,predictions)
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,

$Y = \alpha X + \beta$

Y – Random variable (response, dependent)

X – Random variable (predictor, independent)

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

$\alpha = \Sigma_{i=1:s} (x_i - x') (y_i - y') / \Sigma_{i=1:s} (x_i - x')^2$,

$\beta = y' - \alpha x'$,

where x' - average of $x_1, x_2, \dots, x_s$,

$y' = y_1, y_2, \dots, y_s$,

given sample data points $(x_1, y_1), (x_2, y_2), \dots, (x_s, y_s)$.

Dataset: The dataset we use is iris.data

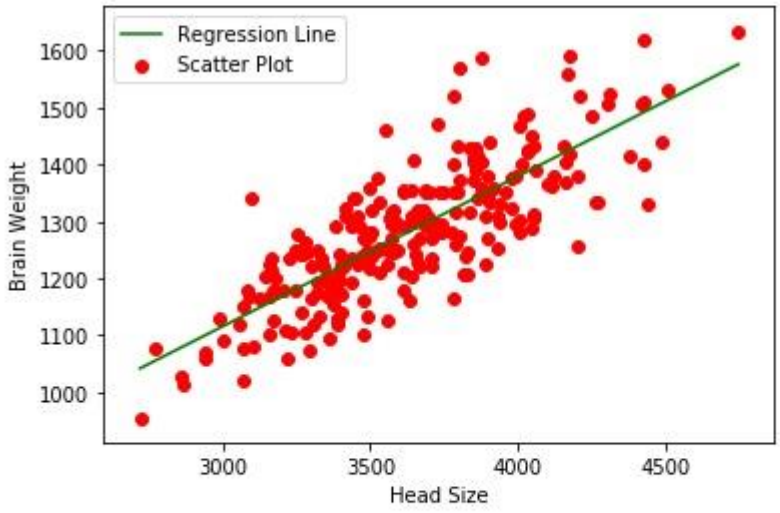
From https://www.kaggle.com/uciml/iris

Source Code:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Sep 30 04:12:40 2019

@author: nowshad
"""
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('headbrain.csv')
print(dataset.shape)
print(dataset.head(5))

X = dataset['Head Size(cm^3)'].values
Y = dataset['Brain Weight(grams)'].values
mean_x = np.mean(X)
mean_y= np.mean(Y)

n=len(X)

numer=0
denom=0
for i in range(n):
    numer += (X[i]-mean_x)*(Y[i]-mean_y)
    denom += (X[i]-mean_x)**2

m= numer/denom
c= mean_y - (m*mean_x)
print("m= ",m)
print("c= ",c)

max_x = np.max(X)+1
min_x= np.min(X)-1
x=np.linspace(min_x,max_x,5)
y=m*x+c

plt.plot(x,y,color='green',label='Regression Line')
plt.scatter(X,Y,color='red',label='Scatter Plot')
plt.xlabel('Head Size')
plt.ylabel('Brain Weight')
plt.legend()
plt.show()

err_nm=0
err_dn=0
for i in range(n):
    y_predict=m*X[i]+c
    err_nm += (Y[i]-y_predict) **2
    err_dn += (Y[i]-mean_y)**2
Accuracy= (1-(err_nm/err_dn))*100
print("\nAccuracy:",Accuracy,"%")
```

Output:

```
In [5]: runfile('G:/AUST4.1/AILab/lab5/Assignment5/LR.py', wdir='G:/
AUST4.1/AILab/lab5/Assignment5')
(237, 4)
   Gender  Age Range  Head Size(cm^3)  Brain Weight(grams)
0       1          1             4512                 1530
1       1          1             3738                 1297
2       1          1             4261                 1335
3       1          1             3777                 1282
4       1          1             4177                 1590
m=  0.26342933948939945
c=  325.57342104944223
```




```
Accuracy: 63.93117199570003 %

In [6]:
```

Accuracy is more than 50% and its good enough.