# MACHINE LEARNING LABORATORY (15CS76)

**1.Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.**

```
import csv
with open('data.csv','r') as f:
    reader=csv.reader(f)
    your_list=list(reader)
h=[['0','0','0','0','0','0']]
for i in your_list:
    print(i)
    if i[-1]=="yes":
        j=0
        for x in i:
            if x!="yes":
                if x!=h[0][j]and h[0][j]=='0':
                    h[0][j]=x
                elif x!=h[0][j] and h[0][j]!='0':
                    h[0][j]='?'
                else:
                    pass
            j=j+1
print("final hypothesis is")
print(h)
```

**Data Set: (file name: data.csv)**

```
sunny,warm,normal,strong,warm,same,yes
sunny,warm,high,strong,warm,same,yes
rain,cold,high,strong,warm,change,no
sunny,warm,hgh,strong,cool,change,yes
```

**output:**

```
C:\Users\admin\PycharmProjects\1rr16cs181\venv\Scripts\python.exe
C:/Users/admin/PycharmProjects/1rr16cs181/finds.py
['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes']
['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes']
['rain', 'cold', 'high', 'strong', 'warm', 'change', 'no']
['sunny', 'warm', 'hgh', 'strong', 'cool', 'change', 'yes']
final hypothesis is
[['sunny', 'warm', '?', 'strong', '?', '?']]

Process finished with exit code 0
```

**2.For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.**

**(NOTE: Install numpy, pandas)**

```
import numpy as np
import pandas as pd

data=pd.DataFrame(data=pd.read_csv('data.csv'))
concepts=np.array(data.iloc[:,0:-1])
target=np.array(data.iloc[:,-1])
def learn(concepts,target):
    specific_h=concepts[0].copy()
    print("intilization of specific_h and general_h")
    print(specific_h)
    general_h=[["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
    print(general_h)
    for i,h in enumerate(concepts):
        if target[i]=="yes":
            for x in range(len(specific_h)):
                if h[x]!=specific_h[x]:
                    specific_h[x]='?'
                    general_h[x][x]='?'
        if target[i]=="no":
            for x in range(len(specific_h)):
                if h[x]!=specific_h[x]:
                    general_h[x][x]=specific_h[x]
                else:
                    general_h[x][x]='?'
    print("steps of candidate elimination algorithm",i+1)
    print("specific_h",i+1,"\n")
    print(specific_h)
    print("general_h",i+1,"\n")
    print(general_h)

    indices=[i for i,val in enumerate(general_h) if val==['?','?','?','?','?','?']]
    for i in indices:
        general_h.remove(['?','?','?','?','?','?'])
    return specific_h,general_h
s_final,g_final=learn(concepts,target)
print("Final specific_h:",s_final,sep="\n")
print("final general_h:",g_final,sep="\n")
```

**Dataset: (file name: data.csv)**

```
sunny,warm,normal,strong,warm,same,yes
sunny,warm,high,strong,warm,same,yes
rain,cold,high,strong,warm,change,no
sunny,warm,hgh,strong,cool,change,yes
```

**Output:**

C:\Users\admin\PycharmProjects\1rr16cs181\venv\Scripts\python.exe
C:/Users/admin/PycharmProjects/1rr16cs181/candi.py

intilization of specific_h and general_h

['sunny' 'warm' 'high' 'strong' 'warm' 'same']

[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

steps of candidate elimination algorithm 3

specific_h 3

['sunny' 'warm' '?' 'strong' '?' '?']

general_h 3

[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Final specific_h:

['sunny' 'warm' '?' 'strong' '?' '?']

final general_h:

[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]

Process finished with exit code 0

**3.Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample**
**(NOTE: Install numpy)**

```
import numpy as np
import math
from data_loader import read_data
class Node:
    def __init__(self,attribute):
        self.attribute=attribute
        self.children=[]
        self.answer=""
    #def __str__(self):
        #return self.attribute
def sub(data,col,delete):
    dict={}
    items=np.unique(data[:,col])
    count=np.zeros((items.shape[0],1),dtype=np.int32)
    for x in range(items.shape[0]):
        for y in range(data.shape[0]):
            if data[y,col]==items[x]:
                count[x]+=1
    for x in range(items.shape[0]):
        dict[items[x]]=np.empty((int(count[x]),data.shape[1]),dtype='|S32')
        pos=0
        for y in range(data.shape[0]):
            if data[y,col]==items[x]:
                dict[items[x]][pos]=data[y]
                pos+=1
        if delete:
            dict[items[x]]=np.delete(dict[items[x]],col,1)
    return items,dict
def entropy(s):
    items=np.unique(s)
    if items.size==1:
        return 0
    counts=np.zeros((items.shape[0],1))
    sums=0
    for x in range(items.shape[0]):
        counts[x]=sum(s==items[x])/(s.size*1.0)
    for count in counts:
        sums+=-1*count*math.log(count,2)
    return sums
def gain(data,col):
    items,dict=sub(data,col,delete=False)
    total_size=data.shape[0]
    entropies=np.zeros((items.shape[0],1))
    intrinsic=np.zeros((items.shape[0],1))
    for x in range((items.shape[0])):
        ratio=dict[items[x]].shape[0]/(total_size*1.0)
```

```
        entropies[x]=ratio*entropy(dict[items[x]][:,-1])
        intrinsic[x]=ratio*math.log(ratio,2)
    total_entropy=entropy(data[:,-1])
    iv =-1*sum(intrinsic)
    for x in range(entropies.shape[0]):
        total_entropy-=entropies[x]
    return (total_entropy/iv)
def create(data,metadata):
    if(np.unique(data[:,-1])).shape[0]==1:
        node=Node("")
        node.answer=np.unique(data[:,-1])[0]
        return node

    gains=np.zeros((data.shape[1]-1,1))
    for col in range(data.shape[-1]-1):
        gains[col]=gain(data,col)
    split=np.argmax(gains)
    node=Node(metadata[split])
    metadata=np.delete(metadata,split,0)
    items,dict=sub(data,split,delete=True)

    for x in range(items.shape[0]):
        child=create(dict[items[x]],metadata)
        node.children.append((items[x],child))
    return node
def empty(size):
    s=""
    for x in range(size):
        s+=""
    return s
def print_tree(node,level):
    if node.answer!="":
        print(empty(level),node.answer)
        return
    print(empty(level),node.attribute)

    for value,n in node.children:
        print(empty(level+1),value)
        print_tree(n,level+2)
metadata,traindata=read_data("data1.csv")
data=np.array(traindata)
node=create(data,metadata)
print_tree(node,0)
```

## data_loader.py [another supporting file]

```
import csv


def read_data(filename):
    with open(filename, 'r') as csvfile:
        datareader = csv.reader(csvfile, delimiter=',')
        headers = next(datareader)
        metadata = []
        traindata = []
        for name in headers:
            metadata.append(name)

        for row in datareader:
            traindata.append(row)

    return (metadata, traindata)
```

**Data set: (file name: data1.csv)**

```
outlook,temprature,humidity,wind,palytennis
sunny,hot,high,weak,no
sunny,hot,high,strong,no
overcast,hot,high,weak,yes
rain,mild,high,weak,yes
rain,cool,normal,weak,yes
rain,cool,normal,strong,no
overcast,cool,normal,strong,yes
sunny,mild,high,weak,no
sunny,cool,normal,weak,yes
rain,mild,normal,weak,yes
sunny,mild,normal,strong,yes
overcast,mild,high,strong,yes
overcast,hot,normal,weak,yes
rain,mild,high,strong,no
```

**Output:**

```
C:\Users\admin\PycharmProjects\1rr16cs181\venv\Scripts\python.exe
C:/Users/admin/PycharmProjects/1rr16cs181/id3.py

 outlook

 overcast

 b'yes'
```

rain

wind

b'strong'

b'no'

b'weak'

b'yes'

sunny

humidity

b'high'

b'no'

b'normal'

b'yes'

Process finished with exit code 0

**4.Build an Artificial Neural Network by implementing the Back propagation algorithm and test the same using appropriate data sets**
**(NOTE: Install numpy)**

```
import numpy as np
x=np.array(([2,9],[1,5],[3,6]),dtype=float)
y=np.array(([92],[86],[89]),dtype=float)
x=x/np.amax(x,axis=0)
y=y/100

def sigmoid(x):
    return (1/(1+np.exp(-x)))
def derivatives_sigmoid(x):
    return x*(1-x)

epoch=7000
lr=0.1
inputlayer_neuron=2
hiddenlayer_neuron=3
output_neuron=1

wh=np.random.uniform(size=(inputlayer_neuron,hiddenlayer_neuron))
bh=np.random.uniform(size=(1,hiddenlayer_neuron))
```

```
wout=np.random.uniform(size=(hiddenlayer_neuron,output_neuron))
bout=np.random.uniform(size=(1,output_neuron))

for i in range(epoch):
    hinp1=np.dot(x,wh)
    hinp=hinp1+bh
    hlayer_act=sigmoid(hinp)
    outinp1=np.dot(hlayer_act,wout)
    outinp=outinp1+bout
    output=sigmoid(outinp)

    EO=y-output
    outgrad=derivatives_sigmoid(output)
    d_output=EO*outgrad
    EH=d_output.dot(wout.T)
    hiddengrad=derivatives_sigmoid(hlayer_act)
    d_hiddenlayer=EH*hiddengrad
    wout+=hlayer_act.T.dot(d_output*lr)
    wh+=x.T.dot(d_hiddenlayer)*lr
print("input:\n"+str(x))
print("actual output:\n"+str(y))
print("predicted output:\n",output)
```

**OUTPUT**

```
C:\Users\admin\PycharmProjects\1rr16cs181\venv\Scripts\python.exe
C:/Users/admin/PycharmProjects/1rr16cs181/nueral.py
input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
actual output:
[[0.92]
 [0.86]
 [0.89]]
predicted output:
 [[0.89612636]
 [0.87868519]
 [0.89460368]]

Process finished with exit code 0
```

**5. Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.**

```
print("\nNaive Bayes Classifier for concept learning problem")
import csv
import random
import math
import operator


def safe_div(x, y):
    if y == 0:
        return 0
    return x / y


def loadCsv(filename):
    lines = csv.reader(open(filename))
    dataset = list(lines)
    for i in range(len(dataset)):
        dataset[i] = [float(x) for x in dataset[i]]
    return dataset


def splitDataset(dataset, splitRatio):
    trainSize = int(len(dataset) * splitRatio)
    trainSet = []
    copy = list(dataset)
    i = 0
    while len(trainSet) < trainSize:
        # index = random.randrange(len(copy))

        trainSet.append(copy.pop(i))
    return [trainSet, copy]


def separateByClass(dataset):
    separated = {}
    for i in range(len(dataset)):
        vector = dataset[i]
        if (vector[-1] not in separated):
            separated[vector[-1]] = []
        separated[vector[-1]].append(vector)
    return separated


def mean(numbers):
    return safe_div(sum(numbers), float(len(numbers)))
```

```
def stdev(numbers):
    avg = mean(numbers)
    variance = safe_div(sum([pow(x - avg, 2) for x in numbers]), float(len(numbers) - 1))
    return math.sqrt(variance)


def summarize(dataset):
    summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)]
    del summaries[-1]
    return summaries


def summarizeByClass(dataset):
    separated = separateByClass(dataset)
    summaries = {}
    for classValue, instances in separated.items():
        summaries[classValue] = summarize(instances)
    return summaries


def calculateProbability(x, mean, stdev):
    exponent = math.exp(-safe_div(math.pow(x - mean, 2), (2 * math.pow(stdev, 2))))
    final = safe_div(1, (math.sqrt(2 * math.pi) * stdev)) * exponent
    return final


def calculateClassProbabilities(summaries, inputVector):
    probabilities = {}
    for classValue, classSummaries in summaries.items():
        probabilities[classValue] = 1
        for i in range(len(classSummaries)):
            mean, stdev = classSummaries[i]
            x = inputVector[i]
            probabilities[classValue] *= calculateProbability(x, mean, stdev)
    return probabilities


def predict(summaries, inputVector):
    probabilities = calculateClassProbabilities(summaries, inputVector)
    bestLabel, bestProb = None, -1
    for classValue, probability in probabilities.items():
        if bestLabel is None or probability > bestProb:
            bestProb = probability
            bestLabel = classValue
    return bestLabel


def getPredictions(summaries, testSet):
    predictions = []
    for i in range(len(testSet)):
        result = predict(summaries, testSet[i])
```

```
        predictions.append(result)
    return predictions


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


def main():
    filename = 'NaiveBayes ConceptLearning.csv'
    splitRatio = 0.75
    dataset = loadCsv(filename)
    trainingSet, testSet = splitDataset(dataset, splitRatio)
    print('Split {0} rows into'.format(len(dataset)))
    print('Number of Training data: ' + (repr(len(trainingSet))))
    print('Number of Test Data: ' + (repr(len(testSet))))
    print("\nThe values assumed for the concept learning attributes are\n")
    print(
        "OUTLOOK=> Sunny=1 Overcast=2 Rain=3\nTEMPERATURE=> Hot=1 Mild=2
Cool=3\nHUMIDITY=> High=1 Normal=2\nWIND=> Weak=1 Strong=2")
    print("TARGET CONCEPT:PLAY TENNIS=> Yes=10 No=5")
    print("\nThe Training set are:")
    for x in trainingSet:
        print(x)
    print("\nThe Test data set are:")
    for x in testSet:
        print(x)
    print("\n")
    # prepare model
    summaries = summarizeByClass(trainingSet)
    # test model
    predictions = getPredictions(summaries, testSet)
    actual = []
    for i in range(len(testSet)):
        vector = testSet[i]
        actual.append(vector[-1])
    # Since there are five attribute values, each attribute constitutes to 20% accuracy. So if all attributes
match with predictions then 100% accuracy
    print('Actual values: {0}%'.format(actual))
    print('Predictions: {0}%'.format(predictions))
    accuracy = getAccuracy(testSet, predictions)
    print('Accuracy: {0}%'.format(accuracy))

main()
```

**Data Set: (file name: NaiveBayes ConceptLearning.csv)**

**1,1,1,1,5**
**1,1,1,2,5**
**2,1,1,2,10**
**3,2,1,1,10**
**3,3,2,1,10**
**3,3,2,2,5**
**2,3,2,2,10**
**1,2,1,1,5**
**1,3,2,1,10**
**3,2,2,2,10**
**1,2,2,2,10**
**2,2,1,2,10**
**2,1,2,1,10**
**3,2,1,2,5**
**1,2,1,2,10**
**1,2,1,2,5**

**OUTPUT**

```
C:\Users\admin\PycharmProjects\ss\venv\Scripts\python.exe
"C:/Users/admin/Desktop/RNSIT_ML_LAB_PROGRAMS_JULY2018/RNSIT_ML_LAB_PROGRAM
S_JULY2018/5-naive-bayes/NaiveBayes ConceptLearning.py"

Naive Bayes Classifier for concept learning problem
Split 16 rows into
Number of Training data: 12
Number of Test Data: 4

The values assumed for the concept learning attributes are

OUTLOOK=> Sunny=1 Overcast=2 Rain=3
TEMPERATURE=> Hot=1 Mild=2 Cool=3
HUMIDITY=> High=1 Normal=2
WIND=> Weak=1 Strong=2
TARGET CONCEPT:PLAY TENNIS=> Yes=10 No=5

The Training set are:
[1.0, 1.0, 1.0, 1.0, 5.0]
[1.0, 1.0, 1.0, 2.0, 5.0]
[2.0, 1.0, 1.0, 2.0, 10.0]
[3.0, 2.0, 1.0, 1.0, 10.0]
[3.0, 3.0, 2.0, 1.0, 10.0]
[3.0, 3.0, 2.0, 2.0, 5.0]
[2.0, 3.0, 2.0, 2.0, 10.0]
[1.0, 2.0, 1.0, 1.0, 5.0]
[1.0, 3.0, 2.0, 1.0, 10.0]
```

```
[3.0, 2.0, 2.0, 2.0, 10.0]
[1.0, 2.0, 2.0, 2.0, 10.0]
[2.0, 2.0, 1.0, 2.0, 10.0]

The Test data set are:
[2.0, 1.0, 2.0, 1.0, 10.0]
[3.0, 2.0, 1.0, 2.0, 5.0]
[1.0, 2.0, 1.0, 2.0, 10.0]
[1.0, 2.0, 1.0, 2.0, 5.0]


Actual values: [10.0, 5.0, 10.0, 5.0]%
Predictions: [5.0, 10.0, 5.0, 5.0]%
Accuracy: 25.0%

Process finished with exit code 0
```

**6.Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your data set.**

**Note:(install pandas and sklearn)**

```
import pandas as pd
msg=pd.read_csv('naivetext1.txt',names=['message','label'])
print('The dimensions of the dataset',msg.shape)
msg['labelnum']=msg.label.map({'pos':1,'neg':0})
X=msg.message
y=msg.labelnum
print(X)
print(y)

#splitting the dataset into train and test data
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(X,y)
print(xtest.shape)
print(xtrain.shape)
print(ytest.shape)
print(ytrain.shape)
#output of count vectoriser is a sparse matrix
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
xtrain_dtm = count_vect.fit_transform(xtrain)
xtest_dtm=count_vect.transform(xtest)
print(count_vect.get_feature_names())

df=pd.DataFrame(xtrain_dtm.toarray(),columns=count_vect.get_feature_names())
print(df)#tabular representation
print(xtrain_dtm) #sparse matrix representation
```

```
# Training Naive Bayes (NB) classifier on training data.
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)

#printing accuracy metrics
from sklearn import metrics
print('Accuracy metrics')
print('Accuracy of the classifer is',metrics.accuracy_score(ytest,predicted))
print('Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
print('Recall and Precison ')
print(metrics.recall_score(ytest,predicted))
print(metrics.precision_score(ytest,predicted))

'''docs_new = ['I like this place', 'My boss is  not my saviour']
X_new_counts = count_vect.transform(docs_new)
predictednew = clf.predict(X_new_counts)
for doc, category in zip(docs_new, predictednew):
  print('%s->%s' % (doc, msg.labelnum[category]))'''
```

**Data Set:**

**Note: In Notepad save the input with filename:** `naivetext1.txt`

```
I love this sandwich,pos
This is an amazing place,pos
I feel very good about these beers,pos
This is my best work,pos
What an awesome view,pos
I do not like this restaurant,neg
I am tired of this stuff,neg
I can't deal with this,neg
He is my sworn enemy,neg
My boss is horrible,neg
This is an awesome place,pos
I do not like the taste of this juice,neg
I love to dance,pos
I am sick and tired of this place,neg
What a great holiday,pos
That is a bad locality to stay,neg
We will have good fun tomorrow,pos
I went to my enemy's house today,neg
```

**OUTPUT**

```
C:\Users\admin\PycharmProjects\ss\venv\Scripts\python.exe
C:/Users/admin/pythonml/6.naivtext.py
The dimensions of the dataset (18, 2)
0                I love this sandwich
1           This is an amazing place
```

```
2        I feel very good about these beers
3                This is my best work
4                What an awesome view
5           I do not like this restaurant
6              I am tired of this stuff
7              I can't deal with this
8                He is my sworn enemy
9                My boss is horrible
10             This is an awesome place
11   I do not like the taste of this juice
12                    I love to dance
13       I am sick and tired of this place
14               What a great holiday
15          That is a bad locality to stay
16          We will have good fun tomorrow
17         I went to my enemy's house today
Name: message, dtype: object
0     1
1     1
2     1
3     1
4     1
5     0
6     0
7     0
8     0
9     0
10    1
11    0
12    1
13    0
14    1
15    0
16    1
17    0
Name: labelnum, dtype: int64
(5,)
(13,)
(5,)
(13,)
['about', 'am', 'amazing', 'an', 'and', 'awesome', 'bad', 'beers', 'best', 'boss', 'can', 'deal',
'do', 'enemy', 'feel', 'good', 'great', 'he', 'holiday', 'horrible', 'house', 'is', 'juice', 'like',
'locality', 'my', 'not', 'of', 'place', 'restaurant', 'sick', 'stay', 'sworn', 'taste', 'that', 'the',
'these', 'this', 'tired', 'to', 'today', 'very', 'went', 'what', 'with', 'work']
about  am  amazing  an  and  ...  very  went  what  with  work
0      0   0        0   1    0  ...     0     0     0     0     0
```

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | ... | 1 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 1 | 0 |
| 8 | 0 | 1 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 1 | 1 | 0 | ... | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 1 |
| 11 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 1 | 0 | 0 | 0 |

[13 rows x 46 columns]

| | |
|---|---|
| (0, 28) | 1 |
| (0, 5) | 1 |
| (0, 3) | 1 |
| (0, 21) | 1 |
| (0, 37) | 1 |
| (1, 22) | 1 |
| (1, 27) | 1 |
| (1, 33) | 1 |
| (1, 35) | 1 |
| (1, 23) | 1 |
| (1, 26) | 1 |
| (1, 12) | 1 |
| (1, 37) | 1 |
| (2, 19) | 1 |
| (2, 9) | 1 |
| (2, 25) | 1 |
| (2, 21) | 1 |
| (3, 29) | 1 |
| (3, 23) | 1 |
| (3, 26) | 1 |
| (3, 12) | 1 |
| (3, 37) | 1 |
| (4, 13) | 1 |
| (4, 32) | 1 |
| (4, 17) | 1 |
| : | : |
| (8, 4) | 1 |
| (8, 30) | 1 |
| (8, 1) | 1 |
| (8, 27) | 1 |
| (8, 28) | 1 |
| (8, 37) | 1 |

**(9, 2) 1**
**(9, 28) 1**
**(9, 3) 1**
**(9, 21) 1**
**(9, 37) 1**
**(10, 45) 1**
**(10, 8) 1**
**(10, 25) 1**
**(10, 21) 1**
**(10, 37) 1**
**(11, 18) 1**
**(11, 16) 1**
**(11, 43) 1**
**(12, 40) 1**
**(12, 20) 1**
**(12, 42) 1**
**(12, 39) 1**
**(12, 13) 1**
**(12, 25) 1**
**Accuracy metrics**
**Accuracy of the classifer is 0.6**
**Confusion matrix**
**[[1 0]**
**[2 2]]**
**Recall and Precison**
**0.5**
**1.0**

**Process finished with exit code 0**

**7.Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.**
**(Note: Install bayespy, numpy, colorama)**

```
import bayespy as bp
import numpy as np
import csv
from colorama import init
from colorama import Fore, Back, Style
init()

# Define Parameter Enum values
#Age
```

```
ageEnum = {'SuperSeniorCitizen':0, 'SeniorCitizen':1, 'MiddleAged':2, 'Youth':3, 'Teen':4}
# Gender
genderEnum = {'Male':0, 'Female':1}
# FamilyHistory
familyHistoryEnum = {'Yes':0, 'No':1}
# Diet(Calorie Intake)
dietEnum = {'High':0, 'Medium':1, 'Low':2}
# LifeStyle
lifeStyleEnum = {'Athlete':0, 'Active':1, 'Moderate':2, 'Sedetary':3}
# Cholesterol
cholesterolEnum = {'High':0, 'BorderLine':1, 'Normal':2}
# HeartDisease
heartDiseaseEnum = {'Yes':0, 'No':1}
#heart_disease_data.csv
with open('BBN heart_disease_data.csv') as csvfile:
    lines = csv.reader(csvfile)
    dataset = list(lines)
    data = []
for x in dataset:

data.append([ageEnum[x[0]],genderEnum[x[1]],familyHistoryEnum[x[2]],dietEnum[x[3]],lifeSt
yleEnum[x[4]],cholesterolEnum[x[5]],heartDiseaseEnum[x[6]]])
# Training data for machine learning todo: should import from csv
data = np.array(data)
N = len(data)

# Input data column assignment
p_age = bp.nodes.Dirichlet(1.0*np.ones(5))
age = bp.nodes.Categorical(p_age, plates=(N,))
age.observe(data[:,0])

p_gender = bp.nodes.Dirichlet(1.0*np.ones(2))
gender = bp.nodes.Categorical(p_gender, plates=(N,))
gender.observe(data[:,1])

p_familyhistory = bp.nodes.Dirichlet(1.0*np.ones(2))
familyhistory = bp.nodes.Categorical(p_familyhistory, plates=(N,))
familyhistory.observe(data[:,2])

p_diet = bp.nodes.Dirichlet(1.0*np.ones(3))
diet = bp.nodes.Categorical(p_diet, plates=(N,))
diet.observe(data[:,3])

p_lifestyle = bp.nodes.Dirichlet(1.0*np.ones(4))
lifestyle = bp.nodes.Categorical(p_lifestyle, plates=(N,))
lifestyle.observe(data[:,4])
```

```
p_cholesterol = bp.nodes.Dirichlet(1.0*np.ones(3))
cholesterol = bp.nodes.Categorical(p_cholesterol, plates=(N,))
cholesterol.observe(data[:,5])

# Prepare nodes and establish edges
# np.ones(2) ->HeartDisease has 2 options Yes/No
# plates(5, 2, 2, 3, 4, 3)  ->  corresponds to options present for domain values
p_heartdisease = bp.nodes.Dirichlet(np.ones(2), plates=(5, 2, 2, 3, 4, 3))
heartdisease = bp.nodes.MultiMixture([age, gender, familyhistory, diet, lifestyle, cholesterol],
bp.nodes.Categorical, p_heartdisease)
heartdisease.observe(data[:,6])
p_heartdisease.update()

# Sample Test with hardcoded values
#print("Sample Probability")
#print("Probability(HeartDisease|Age=SuperSeniorCitizen, Gender=Female,
FamilyHistory=Yes, DietIntake=Medium, LifeStyle=Sedetary, Cholesterol=High)")
#print(bp.nodes.MultiMixture([ageEnum['SuperSeniorCitizen'], genderEnum['Female'],
familyHistoryEnum['Yes'], dietEnum['Medium'], lifeStyleEnum['Sedetary'],
cholesterolEnum['High']], bp.nodes.Categorical,
p_heartdisease).get_moments()[0][heartDiseaseEnum['Yes']])

# Interactive Test
m = 0
while m == 0:
   print("\n")
   res = bp.nodes.MultiMixture([int(input('Enter Age: ' + str(ageEnum))), int(input('Enter
Gender: ' + str(genderEnum))), int(input('Enter FamilyHistory: ' + str(familyHistoryEnum))),
int(input('Enter dietEnum: ' + str(dietEnum))), int(input('Enter LifeStyle: ' +
str(lifeStyleEnum))), int(input('Enter Cholesterol: ' + str(cholesterolEnum)))],
bp.nodes.Categorical, p_heartdisease).get_moments()[0][heartDiseaseEnum['Yes']]
   print("Probability(HeartDisease) = " +  str(res))
#print(Style.RESET_ALL)
m = int(input("Enter for Continue:0, Exit :1  "))
```

**Data Set:**

**(filename: 'BBN heart_disease_data.csv')**

**SuperSeniorCitizen,Male,Yes,Medium,Sedetary,High,Yes**
**SuperSeniorCitizen,Female,Yes,Medium,Sedetary,High,Yes**
**SeniorCitizen,Male,No,High,Moderate,BorderLine,Yes**
**Teen,Male,Yes,Medium,Sedetary,Normal,No**
**Youth,Female,Yes,High,Athlete,Normal,No**
**MiddleAged,Male,Yes,Medium,Active,High,Yes**

**Teen,Male,Yes,High,Moderate,High,Yes**
**SuperSeniorCitizen,Male,Yes,Medium,Sedetary,High,Yes**
**Youth,Female,Yes,High,Athlete,Normal,No**
**SeniorCitizen,Female,No,High,Athlete,Normal,Yes**
**Teen,Female,No,Medium,Moderate,High,Yes**
**Teen,Male,Yes,Medium,Sedetary,Normal,No**
**MiddleAged,Female,No,High,Athlete,High,No**
**MiddleAged,Male,Yes,Medium,Active,High,Yes**
**Youth,Female,Yes,High,Athlete,BorderLine,No**
**SuperSeniorCitizen,Male,Yes,High,Athlete,Normal,Yes**
**SeniorCitizen,Female,No,Medium,Moderate,BorderLine,Yes**
**Youth,Female,Yes,Medium,Athlete,BorderLine,No**
**Teen,Male,Yes,Medium,Sedetary,Normal,No**

**OUTPUT**

Enter Age: {'SuperSeniorCitizen': 0, 'SeniorCitizen': 1, 'MiddleAged': 2, 'Youth': 3, 'Teen': 4}1
Enter Gender: {'Male': 0, 'Female': 1}1
Enter FamilyHistory: {'Yes': 0, 'No': 1}1 Enter dietEnum: {'High': 0, 'Medium': 1, 'Low': 2}2
Enter LifeStyle: {'Athlete': 0, 'Active': 1, 'Moderate': 2, 'Sedetary': 3}2 Enter Cholesterol: {'High': 0, 'BorderLine': 1, 'Normal': 2}1 Probability(HeartDisease) = 0.5 Enter for Continue:0, Exit :1 0 Enter Age: {'SuperSeniorCitizen': 0, 'SeniorCitizen': 1, 'MiddleAged': 2, 'Youth': 3, 'Teen': 4}0 Enter Gender: {'Male': 0, 'Female': 1}0 Enter FamilyHistory: {'Yes': 0, 'No': 1}0 Enter dietEnum: {'High': 0, 'Medium': 1, 'Low': 2}0 Enter LifeStyle: {'Athlete': 0, 'Active': 1, 'Moderate': 2, 'Sedetary': 3}3 Enter Cholesterol: {'High': 0, 'BorderLine': 1, 'Normal': 2}0 Probability(HeartDisease) = 0.5 Enter for Continue:0, Exit :1

**8. Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program**
**(Note: Install numpy, pandas, sklearn)**

```
from sklearn.cluster import KMeans

#from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
data=pd.read_csv("kmeansdata.csv")
df1=pd.DataFrame(data)
print(df1)
f1 = df1['Distance_Feature'].values
f2 = df1['Speeding_Feature'].values
```

```
X=np.matrix(list(zip(f1,f2)))
plt.plot()
plt.xlim([0, 100])
plt.ylim([0, 50])
plt.title('Dataset')
plt.ylabel('Speeding_Feature')
plt.xlabel('Distance_Feature')
plt.scatter(f1,f2)
plt.show()


# create new plot and data
plt.plot()
colors = ['b', 'g', 'r']
markers = ['o', 'v', 's']

# KMeans algorithm
#K = 3
kmeans_model = KMeans(n_clusters=3).fit(X)

plt.plot()
for i, l in enumerate(kmeans_model.labels_):
    plt.plot(f1[i], f2[i], color=colors[l], marker=markers[l],ls='None')
    plt.xlim([0, 100])
    plt.ylim([0, 50])

plt.show()
```

**Data Set: ( filename: kmeansdata.csv)**

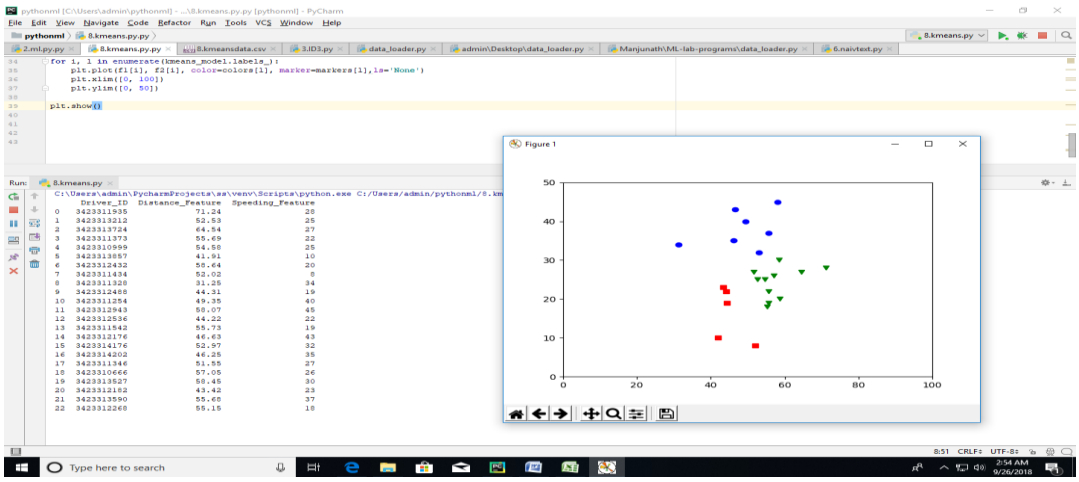
**Driver_ID,Distance_Feature,Speeding_Feature**
**3423311935,71.24,28**
**3423313212,52.53,25**
**3423313724,64.54,27**
**3423311373,55.69,22**
**3423310999,54.58,25**
**3423313857,41.91,10**
**3423312432,58.64,20**
**3423311434,52.02,8**
**3423311328,31.25,34**
**3423312488,44.31,19**
**3423311254,49.35,40**
**3423312943,58.07,45**
**3423312536,44.22,22**
**3423311542,55.73,19**
**3423312176,46.63,43**
**3423314176,52.97,32**
**3423314202,46.25,35**
**3423311346,51.55,27**

**3423310666,57.05,26**
**3423313527,58.45,30**
**3423312182,43.42,23**
**3423313590,55.68,37**
**3423312268,55.15,18**

## OUTPUT

C:\Users\admin\PycharmProjects\ss\venv\Scripts\python.exe C:/Users/admin/pythonml/8.kmeans.py.py

| | Driver_ID | Distance_Feature | Speeding_Feature |
|---|---|---|---|
| 0 | 3423311935 | 71.24 | 28 |
| 1 | 3423313212 | 52.53 | 25 |
| 2 | 3423313724 | 64.54 | 27 |
| 3 | 3423311373 | 55.69 | 22 |
| 4 | 3423310999 | 54.58 | 25 |
| 5 | 3423313857 | 41.91 | 10 |
| 6 | 3423312432 | 58.64 | 20 |
| 7 | 3423311434 | 52.02 | 8 |
| 8 | 3423311328 | 31.25 | 34 |
| 9 | 3423312488 | 44.31 | 19 |
| 10 | 3423311254 | 49.35 | 40 |
| 11 | 3423312943 | 58.07 | 45 |
| 12 | 3423312536 | 44.22 | 22 |
| 13 | 3423311542 | 55.73 | 19 |
| 14 | 3423312176 | 46.63 | 43 |
| 15 | 3423314176 | 52.97 | 32 |
| 16 | 3423314202 | 46.25 | 35 |
| 17 | 3423311346 | 51.55 | 27 |
| 18 | 3423310666 | 57.05 | 26 |
| 19 | 3423313527 | 58.45 | 30 |
| 20 | 3423312182 | 43.42 | 23 |
| 21 | 3423313590 | 55.68 | 37 |
| 22 | 3423312268 | 55.15 | 18 |



**9. Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.**

**KNN ALGORITHM**

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix

from sklearn import datasets
iris=datasets.load_iris()
iris_data=iris.data
iris_labels=iris.target
print(iris_data)
print(iris_labels)
x_train, x_test, y_train, y_test=train_test_split(iris_data,iris_labels,test_size=0.30)

classifier=KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
print('confusion matrix is as follows')
print(confusion_matrix(y_test,y_pred))
print('Accuracy metrics')
print(classification_report(y_test,y_pred))
```

**Data Set:**

**5.1,3.5,1.4,0.2,Iris-setosa,**
**4.9,3,1.4,0.2,Iris-setosa,**
**4.7,3.2,1.3,0.2,Iris-setosa,**
**4.6,3.1,1.5,0.2,Iris-setosa,**
**5,3.6,1.4,0.2,Iris-setosa,**
**5.4,3.9,1.7,0.4,Iris-setosa,**
**4.6,3.4,1.4,0.3,Iris-setosa,**
**5,3.4,1.5,0.2,Iris-setosa,**
**4.4,2.9,1.4,0.2,Iris-setosa,**
**4.9,3.1,1.5,0.1,Iris-setosa,**
**5.4,3.7,1.5,0.2,Iris-setosa,**
**4.8,3.4,1.6,0.2,Iris-setosa,**
**4.8,3,1.4,0.1,Iris-setosa,**
**4.3,3,1.1,0.1,Iris-setosa,**
**5.8,4,1.2,0.2,Iris-setosa,**
**5.7,4.4,1.5,0.4,Iris-setosa,**
**5.4,3.9,1.3,0.4,Iris-setosa,**
**5.1,3.5,1.4,0.3,Iris-setosa,**
**5.7,3.8,1.7,0.3,Iris-setosa,**

**5.1**,**3.8**,**1.5**,**0.3**,**Iris-setosa**,
**5.4**,**3.4**,**1.7**,**0.2**,**Iris-setosa**,
**5.1**,**3.7**,**1.5**,**0.4**,**Iris-setosa**,
**4.6**,**3.6**,**1**,**0.2**,**Iris-setosa**,
**5.1**,**3.3**,**1.7**,**0.5**,**Iris-setosa**,
**4.8**,**3.4**,**1.9**,**0.2**,**Iris-setosa**,
**5**,**3**,**1.6**,**0.2**,**Iris-setosa**,
**5**,**3.4**,**1.6**,**0.4**,**Iris-setosa**,
**5.2**,**3.5**,**1.5**,**0.2**,**Iris-setosa**,
**5.2**,**3.4**,**1.4**,**0.2**,**Iris-setosa**,
**4.7**,**3.2**,**1.6**,**0.2**,**Iris-setosa**,
**4.8**,**3.1**,**1.6**,**0.2**,**Iris-setosa**,
**5.4**,**3.4**,**1.5**,**0.4**,**Iris-setosa**,
**5.2**,**4.1**,**1.5**,**0.1**,**Iris-setosa**,
**5.5**,**4.2**,**1.4**,**0.2**,**Iris-setosa**,
**4.9**,**3.1**,**1.5**,**0.1**,**Iris-setosa**,
**5**,**3.2**,**1.2**,**0.2**,**Iris-setosa**,
**5.5**,**3.5**,**1.3**,**0.2**,**Iris-setosa**,
**4.9**,**3.1**,**1.5**,**0.1**,**Iris-setosa**,
**4.4**,**3**,**1.3**,**0.2**,**Iris-setosa**,
**5.1**,**3.4**,**1.5**,**0.2**,**Iris-setosa**,
**5**,**3.5**,**1.3**,**0.3**,**Iris-setosa**,
**4.5**,**2.3**,**1.3**,**0.3**,**Iris-setosa**,
**4.4**,**3.2**,**1.3**,**0.2**,**Iris-setosa**,
**5**,**3.5**,**1.6**,**0.6**,**Iris-setosa**,
**5.1**,**3.8**,**1.9**,**0.4**,**Iris-setosa**,
**4.8**,**3**,**1.4**,**0.3**,**Iris-setosa**,
**5.1**,**3.8**,**1.6**,**0.2**,**Iris-setosa**,
**4.6**,**3.2**,**1.4**,**0.2**,**Iris-setosa**,
**5.3**,**3.7**,**1.5**,**0.2**,**Iris-setosa**,
**5**,**3.3**,**1.4**,**0.2**,**Iris-setosa**,
**7**,**3.2**,**4.7**,**1.4**,**Iris-versicolor**,
**6.4**,**3.2**,**4.5**,**1.5**,**Iris-versicolor**,
**6.9**,**3.1**,**4.9**,**1.5**,**Iris-versicolor**,
**5.5**,**2.3**,**4**,**1.3**,**Iris-versicolor**,
**6.5**,**2.8**,**4.6**,**1.5**,**Iris-versicolor**,
**5.7**,**2.8**,**4.5**,**1.3**,**Iris-versicolor**,
**6.3**,**3.3**,**4.7**,**1.6**,**Iris-versicolor**,
**4.9**,**2.4**,**3.3**,**1**,**Iris-versicolor**,
**6.6**,**2.9**,**4.6**,**1.3**,**Iris-versicolor**,
**5.2**,**2.7**,**3.9**,**1.4**,**Iris-versicolor**,
**5**,**2**,**3.5**,**1**,**Iris-versicolor**,
**5.9**,**3**,**4.2**,**1.5**,**Iris-versicolor**,
**6**,**2.2**,**4**,**1**,**Iris-versicolor**,
**6.1**,**2.9**,**4.7**,**1.4**,**Iris-versicolor**,
**5.6**,**2.9**,**3.6**,**1.3**,**Iris-versicolor**,

**6.7**,**3.1**,**4.4**,**1.4**,**Iris-versicolor**,
**5.6**,**3**,**4.5**,**1.5**,**Iris-versicolor**,
**5.8**,**2.7**,**4.1**,**1**,**Iris-versicolor**,
**6.2**,**2.2**,**4.5**,**1.5**,**Iris-versicolor**,
**5.6**,**2.5**,**3.9**,**1.1**,**Iris-versicolor**,
**5.9**,**3.2**,**4.8**,**1.8**,**Iris-versicolor**,
**6.1**,**2.8**,**4**,**1.3**,**Iris-versicolor**,
**6.3**,**2.5**,**4.9**,**1.5**,**Iris-versicolor**,
**6.1**,**2.8**,**4.7**,**1.2**,**Iris-versicolor**,
**6.4**,**2.9**,**4.3**,**1.3**,**Iris-versicolor**,
**6.6**,**3**,**4.4**,**1.4**,**Iris-versicolor**,
**6.8**,**2.8**,**4.8**,**1.4**,**Iris-versicolor**,
**6.7**,**3**,**5**,**1.7**,**Iris-versicolor**,
**6**,**2.9**,**4.5**,**1.5**,**Iris-versicolor**,
**5.7**,**2.6**,**3.5**,**1**,**Iris-versicolor**,
**5.5**,**2.4**,**3.8**,**1.1**,**Iris-versicolor**,
**5.5**,**2.4**,**3.7**,**1**,**Iris-versicolor**,
**5.8**,**2.7**,**3.9**,**1.2**,**Iris-versicolor**,
**6**,**2.7**,**5.1**,**1.6**,**Iris-versicolor**,
**5.4**,**3**,**4.5**,**1.5**,**Iris-versicolor**,
**6**,**3.4**,**4.5**,**1.6**,**Iris-versicolor**,
**6.7**,**3.1**,**4.7**,**1.5**,**Iris-versicolor**,
**6.3**,**2.3**,**4.4**,**1.3**,**Iris-versicolor**,
**5.6**,**3**,**4.1**,**1.3**,**Iris-versicolor**,
**5.5**,**2.5**,**4**,**1.3**,**Iris-versicolor**,
**5.5**,**2.6**,**4.4**,**1.2**,**Iris-versicolor**,
**6.1**,**3**,**4.6**,**1.4**,**Iris-versicolor**,
**5.8**,**2.6**,**4**,**1.2**,**Iris-versicolor**,
**5**,**2.3**,**3.3**,**1**,**Iris-versicolor**,
**5.6**,**2.7**,**4.2**,**1.3**,**Iris-versicolor**,
**5.7**,**3**,**4.2**,**1.2**,**Iris-versicolor**,
**5.7**,**2.9**,**4.2**,**1.3**,**Iris-versicolor**,
**6.2**,**2.9**,**4.3**,**1.3**,**Iris-versicolor**,
**5.1**,**2.5**,**3**,**1.1**,**Iris-versicolor**,
**5.7**,**2.8**,**4.1**,**1.3**,**Iris-versicolor**,
**6.3**,**3.3**,**6**,**2.5**,**Iris-virginica**,
**5.8**,**2.7**,**5.1**,**1.9**,**Iris-virginica**,
**7.1**,**3**,**5.9**,**2.1**,**Iris-virginica**,
**6.3**,**2.9**,**5.6**,**1.8**,**Iris-virginica**,
**6.5**,**3**,**5.8**,**2.2**,**Iris-virginica**,
**7.6**,**3**,**6.6**,**2.1**,**Iris-virginica**,
**4.9**,**2.5**,**4.5**,**1.7**,**Iris-virginica**,
**7.3**,**2.9**,**6.3**,**1.8**,**Iris-virginica**,
**6.7**,**2.5**,**5.8**,**1.8**,**Iris-virginica**,
**7.2**,**3.6**,**6.1**,**2.5**,**Iris-virginica**,
**6.5**,**3.2**,**5.1**,**2**,**Iris-virginica**,

**6.4**,**2.7**,**5.3**,**1.9**,**Iris-virginica**,
**6.8**,**3**,**5.5**,**2.1**,**Iris-virginica**,
**5.7**,**2.5**,**5**,**2**,**Iris-virginica**,
**5.8**,**2.8**,**5.1**,**2.4**,**Iris-virginica**,
**6.4**,**3.2**,**5.3**,**2.3**,**Iris-virginica**,
**6.5**,**3**,**5.5**,**1.8**,**Iris-virginica**,
**7.7**,**3.8**,**6.7**,**2.2**,**Iris-virginica**,
**7.7**,**2.6**,**6.9**,**2.3**,**Iris-virginica**,
**6**,**2.2**,**5**,**1.5**,**Iris-virginica**,
**6.9**,**3.2**,**5.7**,**2.3**,**Iris-virginica**,
**5.6**,**2.8**,**4.9**,**2**,**Iris-virginica**,
**7.7**,**2.8**,**6.7**,**2**,**Iris-virginica**,
**6.3**,**2.7**,**4.9**,**1.8**,**Iris-virginica**,
**6.7**,**3.3**,**5.7**,**2.1**,**Iris-virginica**,
**7.2**,**3.2**,**6**,**1.8**,**Iris-virginica**,
**6.2**,**2.8**,**4.8**,**1.8**,**Iris-virginica**,
**6.1**,**3**,**4.9**,**1.8**,**Iris-virginica**,
**6.4**,**2.8**,**5.6**,**2.1**,**Iris-virginica**,
**7.2**,**3**,**5.8**,**1.6**,**Iris-virginica**,
**7.4**,**2.8**,**6.1**,**1.9**,**Iris-virginica**,
**7.9**,**3.8**,**6.4**,**2**,**Iris-virginica**,
**6.4**,**2.8**,**5.6**,**2.2**,**Iris-virginica**,
**6.3**,**2.8**,**5.1**,**1.5**,**Iris-virginica**,
**6.1**,**2.6**,**5.6**,**1.4**,**Iris-virginica**,
**7.7**,**3**,**6.1**,**2.3**,**Iris-virginica**,
**6.3**,**3.4**,**5.6**,**2.4**,**Iris-virginica**,
**6.4**,**3.1**,**5.5**,**1.8**,**Iris-virginica**,
**6**,**3**,**4.8**,**1.8**,**Iris-virginica**,
**6.9**,**3.1**,**5.4**,**2.1**,**Iris-virginica**,
**6.7**,**3.1**,**5.6**,**2.4**,**Iris-virginica**,
**6.9**,**3.1**,**5.1**,**2.3**,**Iris-virginica**,
**5.8**,**2.7**,**5.1**,**1.9**,**Iris-virginica**,
**6.8**,**3.2**,**5.9**,**2.3**,**Iris-virginica**,
**6.7**,**3.3**,**5.7**,**2.5**,**Iris-virginica**,
**6.7**,**3**,**5.2**,**2.3**,**Iris-virginica**,
**6.3**,**2.5**,**5**,**1.9**,**Iris-virginica**,
**6.5**,**3**,**5.2**,**2**,**Iris-virginica**,
**6.2**,**3.4**,**5.4**,**2.3**,**Iris-virginica**,
**5.9**,**3**,**5.1**,**1.8**,**Iris-virginica**,

**OUTPUT**

```
C:\Users\admin\PycharmProjects\ss\venv\Scripts\python.exe
C:/Users/admin/pythonml/9.KNN.py
[[5.1 3.5 1.4 0.2]
[4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
[4.8 3.  1.4 0.1]
[4.3 3.  1.1 0.1]
[5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
[4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.  3.  1.6 0.2]
 [5.  3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
 [5.5 4.2 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.  3.2 1.2 0.2]
 [5.5 3.5 1.3 0.2]
 [4.9 3.1 1.5 0.1]
[4.4 3.  1.3 0.2]
 [5.1 3.4 1.5 0.2]
 [5.  3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
```

[5.  3.5 1.6 0.6]
[5.1 3.8 1.9 0.4]
[4.8 3.  1.4 0.3]
[5.1 3.8 1.6 0.2]
[4.6 3.2 1.4 0.2]
[5.3 3.7 1.5 0.2]
[5.  3.3 1.4 0.2]
[7.  3.2 4.7 1.4]
[6.4 3.2 4.5 1.5]
[6.9 3.1 4.9 1.5]
[5.5 2.3 4.  1.3]
[6.5 2.8 4.6 1.5]
[5.7 2.8 4.5 1.3]
[6.3 3.3 4.7 1.6]
[4.9 2.4 3.3 1. ]
[6.6 2.9 4.6 1.3]
[5.2 2.7 3.9 1.4]
[5.  2.  3.5 1. ]
[5.9 3.  4.2 1.5]
[6.  2.2 4.  1. ]
[6.1 2.9 4.7 1.4]
[5.6 2.9 3.6 1.3]
[6.7 3.1 4.4 1.4]
[5.6 3.  4.5 1.5]
[5.8 2.7 4.1 1. ]
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.  1.3]
[6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.  4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.5.  1.7]
[6.  2.9 4.5 1.5]
[5.7 2.6 3.5 1. ]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1. ]
[5.8 2.7 3.9 1.2]
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
[6.  3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.  4.1 1.3]

[5.5 2.5 4.  1.3]
 [5.5 2.6 4.4 1.2]
[6.1 3.  4.6 1.4]
[5.8 2.6 4.  1.2]
 [5.  2.3 3.3 1. ]
 [5.6 2.7 4.2 1.3]
[5.7 3.  4.2 1.2]
 [5.7 2.9 4.2 1.3]
 [6.2 2.9 4.3 1.3]
[5.1 2.5 3.  1.1]
 [5.7 2.8 4.1 1.3]
[6.3 3.3 6.  2.5]
 [5.8 2.7 5.1 1.9]
[7.1 3.  5.9 2.1]
 [6.3 2.9 5.6 1.8]
[6.5 3.  5.8 2.2]
[7.6 3.  6.6 2.1]
 [4.9 2.5 4.5 1.7]
 [7.3 2.9 6.3 1.8]
 [6.7 2.5 5.8 1.8]
 [7.2 3.6 6.1 2.5]
 [6.5 3.2 5.1 2. ]
 [6.4 2.7 5.3 1.9]
[6.8 3.  5.5 2.1]
[5.7 2.5 5.2. ]
 [5.8 2.8 5.1 2.4]
 [6.4 3.2 5.3 2.3]
[6.5 3.  5.5 1.8]
 [7.7 3.8 6.7 2.2]
 [7.7 2.6 6.9 2.3]
 [6.  2.2 5.  1.5]
 [6.9 3.2 5.7 2.3]
 [5.6 2.8 4.9 2. ]
 [7.7 2.8 6.7 2. ]
 [6.3 2.7 4.9 1.8]
 [6.7 3.3 5.7 2.1]
[7.2 3.2 6.  1.8]
 [6.2 2.8 4.8 1.8]
[6.1 3.  4.9 1.8]
 [6.4 2.8 5.6 2.1]
[7.2 3.  5.8 1.6]
 [7.4 2.8 6.1 1.9]
 [7.9 3.8 6.4 2. ]
 [6.4 2.8 5.6 2.2]
 [6.3 2.8 5.1 1.5]
 [6.1 2.6 5.6 1.4]

[7.7 3.  6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.  3.  4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
[6.7 3.  5.2 2.3]
[6.3 2.5 5.  1.9]
[6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
[5.9 3.  5.1 1.8]]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
confusion matrix is as follows
[[21  0  0]
[ 0 11  2]
[ 0  0 11]]
Accuracy metrics

| precision | recall | f1-score | support |
|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 21 |
| 1 | 1.00 | 0.85 | 0.92 | 13 |
| 2 | 0.85 | 1.00 | 0.92 | 11 |
| avg / total | 0.96 | 0.96 | 0.96 | 45 |

Process finished with exit code 0

**10. Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.**

**LOCALLY WEIGHTED REGRESSION**

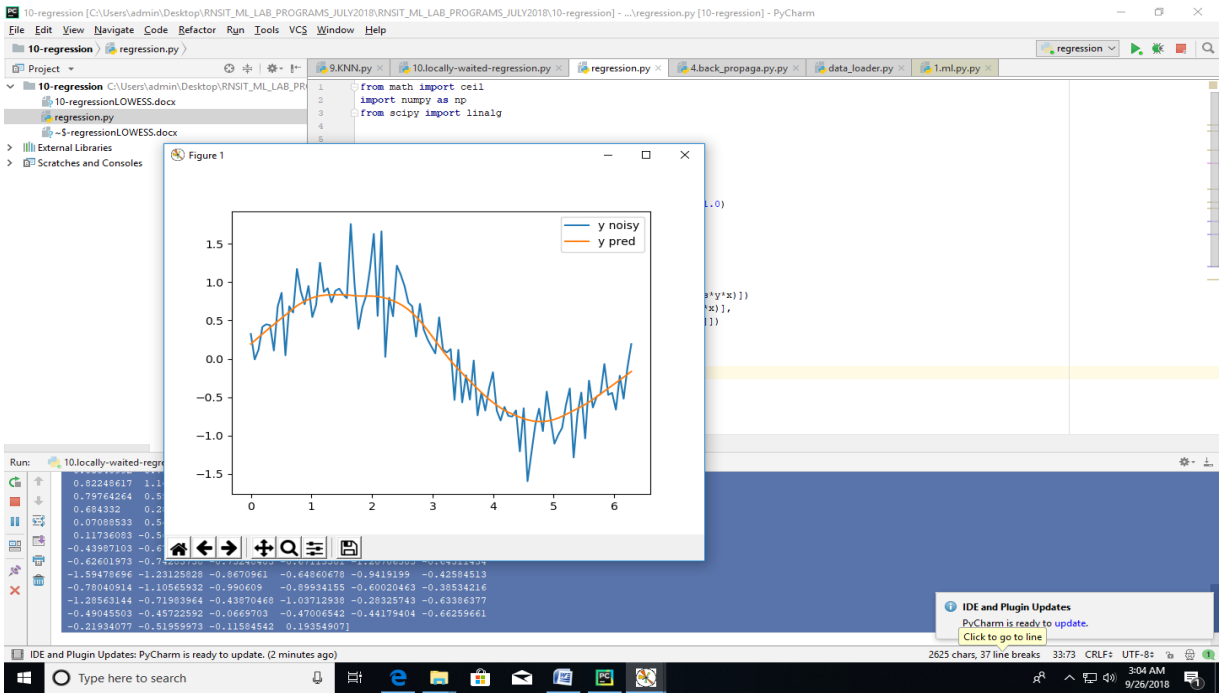
**(Note: Install** numpy, scipy)

```
from math import ceil
import numpy as np
from scipy import linalg


def lowess(x, y, f=2./3., iter=3):
    n = len(x)
    r = int(ceil(f*n))
    h = [np.sort(np.abs(x - x[i]))[r] for i in range(n)]
    w = np.clip(np.abs((x[:,None] - x[None,:]) / h), 0.0, 1.0)
    w = (1 - w**3)**3
    yest = np.zeros(n)
    delta = np.ones(n)
    for iteration in range(iter):
        for i in range(n):
            weights = delta * w[:,i]
            b = np.array([np.sum(weights*y), np.sum(weights*y*x)])
            A = np.array([[np.sum(weights), np.sum(weights*x)],
                  [np.sum(weights*x), np.sum(weights*x*x)]])
            beta = linalg.solve(A, b)
            yest[i] = beta[0] + beta[1]*x[i]

        residuals = y - yest
        s = np.median(np.abs(residuals))
        delta = np.clip(residuals / (6.0 * s), -1, 1)
        delta = (1 - delta**2)**2

    return yest

if __name__ == '__main__':
    import math
    n = 100
    x = np.linspace(0, 2 * math.pi, n)
    print("==========================values of x====================")
    print(x)
    y = np.sin(x) + 0.3*np.random.randn(n)
    print("================================Values of y=================")
    print(y)
    f = 0.25
    yest = lowess(x, y, f=f, iter=3)

    import pylab as pl
    pl.clf()
    pl.plot(x, y, label='y noisy')
    pl.plot(x, yest, label='y pred')
```

```
    pl.legend()
    pl.show()
```

OUTPUT

```
C:\Users\admin\PycharmProjects\ss\venv\Scripts\python.exe
C:/Users/admin/Desktop/RNSIT_ML_LAB_PROGRAMS_JULY2018/RNSIT_ML_LAB_PROGRAMS
_JULY2018/10-regression/regression.py
==========values of x==========
[0.         0.06346652 0.12693304 0.19039955 0.25386607 0.31733259
 0.38079911 0.44426563 0.50773215 0.57119866 0.63466518 0.6981317
 0.76159822 0.82506474 0.88853126 0.95199777 1.01546429 1.07893081
 1.14239733 1.20586385 1.26933037 1.33279688 1.3962634  1.45972992
 1.52319644 1.58666296 1.65012947 1.71359599 1.77706251 1.84052903
 1.90399555 1.96746207 2.03092858 2.0943951  2.15786162 2.22132814
 2.28479466 2.34826118 2.41172769 2.47519421 2.53866073 2.60212725
 2.66559377 2.72906028 2.7925268  2.85599332 2.91945984 2.98292636
 3.04639288 3.10985939 3.17332591 3.23679243 3.30025895 3.36372547
 3.42719199 3.4906585  3.55412502 3.61759154 3.68105806 3.74452458
3.8079911  3.87145761 3.93492413 3.99839065 4.06185717 4.12532369
4.1887902  4.25225672 4.31572324 4.37918976 4.44265628 4.5061228
 4.56958931 4.63305583 4.69652235 4.75998887 4.82345539 4.88692191
 4.95038842 5.01385494 5.07732146 5.14078798 5.2042545  5.26772102
 5.33118753 5.39465405 5.45812057 5.52158709 5.58505361 5.64852012
 5.71198664 5.77545316 5.83891968 5.9023862  5.96585272 6.02931923
 6.09278575 6.15625227 6.21971879 6.28318531]
==========Values of y==========
[ 0.32536008 -0.0080573   0.11946369  0.41612046  0.45098579  0.43815367
0.10801193  0.68389606  0.86074625  0.04549917  0.68505644  0.60342634
1.17247156  0.88083937  0.71119685  0.95001511  0.54481781  0.7051224
1.25351458  0.8712536   0.92022204  0.7352142   0.88698095  0.91535147
0.83840992  0.7904273   1.75713902  0.9658919   0.39042121  0.66715723
0.82248617  1.16770788  1.62890879  0.55892447  1.66198264  0.02503305
0.79764264  0.55443527  1.21535481  1.09842121  0.94842294  0.73174791
  0.684332    0.28964437  0.71744902  0.37907153  0.2530457   0.15897645
0.07088533  0.54206641  0.12110612  0.08384214  0.12731212 -0.53552899
  0.11736083 -0.56747834 -0.21437779 -0.53090037 -0.02105477 -0.7363005
 -0.43987103 -0.67372833 -0.38014677 -0.17410718 -0.67528673 -0.80375547
 -0.62601973 -0.74283758 -0.75248483 -0.67113581 -1.20706585 -0.64311434
 -1.59478696 -1.23125828 -0.8670961  -0.64860678 -0.9419199  -0.42584513
 -0.78040914 -1.10565932 -0.990609   -0.89934155 -0.60020463 -0.38534216
 -1.28563144 -0.71983964 -0.43870468 -1.03712938 -0.28325743 -0.63386377
 -0.49045503 -0.45722592 -0.0669703  -0.47006542 -0.44179404 -0.66259661
 -0.21934077 -0.51959973 -0.11584542  0.19354907]
```


10-regression [C:\Users\admin\Desktop\RNSIT_ML_LAB_PROGRAMS_JULY2018\RNSIT_ML_LAB_PROGRAMS_JULY2018\10-regression] - ...\regression.py [10-regression] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
10-regression
regression.py
regression
Project
9.KNN.py
10.locally-waited-regression.py
regression.py
4.back_propaga.py.py
data_loader.py
1.ml.py.py
10-regression C:\Users\admin\Desktop\RNSIT_ML_LAB_PR
10-regressionLOWESS.docx
regression.py
~$-regressionLOWESS.docx
External Libraries
Scratches and Consoles
from math import ceil
import numpy as np
from scipy import linalg
Figure 1
y noisy
y pred
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
1
2
3
4
5
6
Run:
10.locally-waited-regre
0.82248617 1.1
0.79764264 0.5
0.684332 0.2
0.07088533 0.5
0.11736083 -0.5
-0.43987103 -0.6
-0.62601973
-1.59478696 -1.23125828 -0.8670961 -0.64860678 -0.9419199 -0.42584513
-0.78040914 -1.10565932 -0.990609 -0.89934155 -0.60020463 -0.38534216
-1.28563144 -0.71983964 -0.43870468 -1.03712938 -0.28325743 -0.63386377
-0.49045503 -0.45722592 -0.0669703 -0.47006542 -0.44179404 -0.66259661
-0.21934077 -0.51959973 -0.11584542 0.19354907]
IDE and Plugin Updates
PyCharm is ready to update.
Click to go to line
IDE and Plugin Updates: PyCharm is ready to update. (2 minutes ago)
2625 chars, 37 line breaks
33:73
CRLF
UTF-8
Type here to search
3:04 AM
9/26/2018