DATE: PAGE NO.

Lab Program - 8

Write a program to Implement K-nearest neighbour algorithm to classify Iris dataset. Print both correct & wrong predication using phython machine learning.

Theory :- K-nearest neighbours algorithm (k-nn) is a non-parametric method used for classification & regression. In both cases the input consist of K closest training examples in the feature space. The output depends on whether k-nn is used for classification or regression.

K-nn is a type of instance based learning or lazy learning where the function is only approximated locally & all computation is deferred until classification. The k-nn algorithm is among the simplest of all machine learning algorithm.

The knn task can be broken down into writing 3 primary functions:

1. Calculate the distance b/w any two points.
2. Find the nearest neighbour based on these pair wise distances.
3. Majority vote on class labels based on the nearest neighbour list

Teacher's Remarks

Teacher's Signature

DATE: PAGE NO.

PROGRAM :

```
import csv
import random
import math
import operator

def loadDataset(filename, split, trainingSet=[], testSet=[]).
        with open(filename) 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]),
        return math.sqrt(distance)
```

DATE: PAGE NO.

```
def getNeighbors (trainingSet, testInstance K):
    distances = []
    length = len(testInstance) - 1
    for x in range (len(trainingSet)):
        dist = euclideanDistance (testInstance, trainingSet[x]
                                  , length)
        distances.append((trainingSet[x], dist))
    distances.sort (Key = operator.itemgetter (1))
    neighbors = []
    for x in range (K):
        neighbors.append (distances[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]
```

PAGE NO.

DATE:

```
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

def main():

    # prepare data
    trainingSet =[]
    testSet = []
    split = 0.67
    loadDataset('iris_data.csv', split, trainingSet, testSet)
    print('\n Number of Training data:' + (repr(len(trainingSet))))
    print('Number of Test Data:' + (repr(len(testSet))))
    # generate predictions
    predictions = []
    K = 3
    print('\n The predictions are:')
    for x in range (len(testSet)):
            neighbors = getNeighbours(trainingSet, testSet[x], K)
            result = getResponse(neighbors)
            predictions.append(result)
            print('predicted =' + repr(result) + ', actual =' + repr
                                    (testSet[x][-1]))

    accuracy = getAccuracy(testSet, predictions)
```

Teacher's Remarks

Teacher's Signature

DATE: PAGE NO.

print ('\n The Accuracy is : ' + repr (accuracy) + '%')

main()

Output-

Iris Data set loaded...
Dataset is split into training and testing...
Size of trainng data and its label (135, 4) (135,)
Size of trainng data and its label (15, 4) (15,)
Label 0 - setosa
Label 1 - versicolor
Label 2 - virginica
Results of Classification using K-nn with K=1
Sample: [5.8 4. 1.2 0.2] Actual-label: 0 Predicted-label: 0
Sample: [6.4 3.2 4.5 1.5] Actual-label: 1 Predicted-label: 1
Sample: [5.6 3. 4.1 1.3] Actual-label: 1 Predicted-label: 1
Sample: [5.8 2.7 5.1 1.9] Actual-label: 2 Predicted-label: 2
Sample: [6.7 3.1 4.7 1.5] Actual-label: 1 Predicted-label: 1
Sample: [6.7 3.1 4.4 1.4] Actual-label: 1 Predicted-label: 1
Sample: [6.8 3.2 5.9 2.3] Actual-label: 2 Predicted-label: 2
Sample: [5. 3.3 1.4 0.2] Actual-label: 0 Predicted-label: 0
Sample: [6.8 3. 5.5 2.1] Actual-label: 2 Predicted-label: 2
Sample: [6.1 2.8 4.7 1.2] Actual-label: 1 Predicted-label: 1
Sample: [4.9 3. 1.4 0.2] Actual-label: 0 Predicted-label: 0
Sample: [5.5 3.5 1.3 0.2] Actual-label: 0 Predicted-label: 0
Sample: [5.8 2.7 5.1 1.9] Actual-label: 2 Predicted-label: 2
Sample: [6.3 2.3 4.4 1.3] Actual-label: 1 Predicted-label: 1
Sample: [6.5 3. 5.8 2.2] Actual-label: 2 Predicted-label: 2
Classification Accuracy : 1.0

Teacher's Remarks

Teacher's Signature