07/06/2021 **Mohd. Aftab Alam**
02013302717

EXPERIMENT 7

Aim :

Develop a Machine Learning Method to Predict how People would Rate Movies, Books, etc.

Introduction :

k*-Nearest Neighbors (*k*-NN)** are a classical method for recommender systems. Ratings or items are predicted by using the past ratings/items of the ***k most similar users and/or items. If the influence of the similar users/items is weighted by the similarity, all users/items may be used for the prediction. Popular similarity metrics are the Pearson Correlation and the Cosine Similarity.

Using the user-item matrix to compute the similarity is often called collaborative filtering. Computing the item similarities from the item attributes leads to content-based filtering. The ***k***-nearest neighbors (***k***-NN) algorithm uses an idea distance between each instance. The distance between two messages can be measured and it can be determined how close they are to each other. The algorithm attempts to classify the message be looking at the nearest neighbors. The distance is usually measured by using the Euclidian distance. The formula for the Euclidian distance is given by:

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

The goal is to measure the distance of ***k*** closest neighbors and take the majority vote of those neighbors to classify the current message.

Training: To create the training data simply store the feature vectors and the label of each message.

Classification:

Take a message ***m*,** find the ***k*** nearest neighbors and count the number of each label (spam or not) that are given from the neighbors. If there are more spam messages in the ***k*** nearest neighbors then it is classified as spam. If not, then that message is classified as legitimate mail.

One advantage of this algorithm is that there isn't really a training phase. However, to classify a message, all distances between that message and all the training examples must be calculated and the ***k*** nearest neighbors need to be found and counted.

Dataset used: MovieLens

Implementation :

```
import csv

import random

import math

import operator


def loadDataset(filename, split, trainingSet=[] , testSet=[]):

with open(filename, 'rb') 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 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.iteritems(), 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] is predictions[x]:

correct += 1

return (correct/float(len(testSet))) * 100.0
```

```
def main():

# prepare data
trainingSet=[]
testSet=[]
split = 0.67
loadDataset('iris.data', split, trainingSet, testSet)

print 'Train set: ' + repr(len(trainingSet))
print 'Test set: ' + repr(len(testSet))

# generate predictions
predictions=[]
k = 3
for x in range(len(testSet)):
neighbors = getNeighbors(trainingSet, testSet[x], k)
result = getResponse(neighbors)
predictions.append(result)

print('> predicted=' + repr(result) + ', actual=' + repr(testSet[x][-1]))
accuracy = getAccuracy(testSet, predictions)
print('Accuracy: ' + repr(accuracy) + '%')

main()
```