A07

Knn takes arguments

data -> trainingData

q -> point whose class is to be predicted

k -> value of k to be used

In [160…

```
def knn(data,q,k):

    def give_dist(c1,c2): # function gives distance between two points
        sum = 0
        for i in range(len(c1)):
            sum += (c1[i] - c2[i])**2
        return sum**0.5

    AB_dists = [(give_dist(q,x[1]),x[0]) for x in data]

    # used to print data for markdown tables
    # for x in range(4):
    #     print(f"|D{x+1}|1|{x+1}|{data[x][0]}|{AB_dists[x][0]}|")

    AB_dists = sorted(AB_dists,key = lambda s : s[0])

    # counts labels of closest k labels
    d = {}
    for i in range(k):
        if AB_dists[i][1] in d: d[AB_dists[i][1]] += 1
        else: d[AB_dists[i][1]] = 1

    #get label with maximum neighbours
    AB_max = 0
    AB_maxkey = ''
    for x,y in d.items():
        if(y>AB_max):
            AB_max = y
            AB_maxkey = x

    return AB_maxkey
```

Q1 For Comedy and Action movies

In [161…

```
# co-ordinate system (comedy scenes,action scenes)
AB_trainingData = [('comedy',(100,0)),('action',(0,100)),('action',(15,90)),('cc
AB_validationData = [('action',(10,95)),('comedy',(85,15))]

AB_AB_maxk = 0
AB_AB_maxcur = 0

#loop to get best k
for AB_k in range(1,len(AB_trainingData) + 1,2):
    AB_cor = 0
    for AB_y,AB_x in AB_validationData:
        if knn(AB_trainingData,AB_x,AB_k) == AB_y:
            AB_cor += 1
    if AB_cor > AB_AB_maxcur:
        AB_AB_maxcur = AB_cor
        AB_AB_maxk = AB_k

print("best k = ",AB_AB_maxk)

#predictions
AB_testData = [(6,70),(93,23),(50,50)]
AB_preds = []
for x in AB_testData:
        print(f"{x} -> {knn(AB_trainingData,x,AB_AB_maxk)}")
```

```
best k =  1
(6, 70) -> action
(93, 23) -> comedy
(50, 50) -> comedy
```

Table 1: Unsorted Distances

S.No	Validation Data	Train Data	Predicted Class	Euclidean Distance
D1	1	1	comedy	130.86252328302402
D2	1	2	action	11.180339887498949
D3	1	3	action	7.0710678118654755
D4	1	4	comedy	106.06601717798213
D5	1	1	comedy	21.213203435596427
D6	1	2	action	120.20815280171308
D7	1	3	action	102.59142264341595
D8	1	4	comedy	5.0

Table 2: Sorted Distances

S.No	Validation Data	Train Data	Predicted Class	Euclidean Distance
D8	1	4	comedy	5.0
D3	1	3	action	7.0710678118654755
D2	1	2	action	11.180339887498949
D5	1	1	comedy	21.213203435596427
D7	1	3	action	102.59142264341595
D4	1	4	comedy	106.06601717798213
D6	1	2	action	120.20815280171308
D1	1	1	comedy	130.86252328302402

Table 3: Accuracy for different values of 'k'

S.No	Validation Data	Accuracy for K=1	Accuracy for K=3
1	(10,95),Action	Action	Action
1	(85,15),Comedy	Comedy	Comedy

Table 4: Predictions using k = 1

S.No	No of Comedy Scene	No of action scene	Predicted Class	Value of 'k'
1	6	70	Action	1
1	93	23	Comedy	1
1	50	50	Comedy	1

Q2 Using KNN on IRIS Dataset

Dataset obtained from iris.csv on https://gist.github.com/curran/a08a1080b88344b0c8a7

Since we are using random to shuffle iris dataset best k depends on the random shuffle

In [174…

```
import random

iris = []

with open("iris.csv","r") as f:
    f.readline()
    for line in f:
        line = line.strip().split(',')
        iris.append((line[-1],[float(x) for x in line[:-1]]))

random.shuffle(iris) # shuffling data as iris is ordered by label

AB_trainingData = iris[:74] # 70% for 70% of total
AB_validationData = iris[74:105] # 30% for 70% of total
AB_testData = iris[105:] # 30% of total

AB_AB_maxk = 0
AB_AB_maxcur = 0

print("Validation Data Accuracy for different K")
for AB_k in [1,2,3,4,5,6,7,8,9]:
    AB_cor = 0
    for AB_y,AB_x in AB_validationData:
        if knn(AB_trainingData,AB_x,AB_k) == AB_y:
            AB_cor += 1
    print(f"K = {AB_k} Accuracy = {round(AB_cor/len(AB_validationData),7)*100}%"
    if AB_cor > AB_AB_maxcur:
        AB_AB_maxcur = AB_cor
        AB_AB_maxk = AB_k

print("Best k = ",AB_AB_maxk)
```

```
correct = 0
for x in AB_testData:
    pred = knn(AB_trainingData,x[1],AB_AB_maxk)
    if(pred == x[0]):correct += 1


print("Test Accuracy for best K ",correct/len(AB_testData))
```

```
Validation Data Accuracy for different K
K = 1 Accuracy = 93.54839%
K = 2 Accuracy = 93.54839%
K = 3 Accuracy = 93.54839%
K = 4 Accuracy = 93.54839%
K = 5 Accuracy = 93.54839%
K = 6 Accuracy = 93.54839%
K = 7 Accuracy = 96.77419%
K = 8 Accuracy = 93.54839%
K = 9 Accuracy = 96.77419%
Best k =  7
Test Accuracy for best K  0.9777777777777777
```