Barış Can 24/03/2019

21501886

HW1

1-)

a-)

b-)

c-)

d-)

2-)

a-)

P(S = disease) = 0.005

P(S = healthy) = 0.995

P(T = P | S = D) = 0.97

P(T = N | S = D) = 0.03

P(T = P | S = H) = 0.02

P(T = N | S = H) = 0.98

b-)

3-)

a-)

b-)

c-)

4-)

a-)

Since we are comparing the vectors and the mean that we have, we do not need to compare the denominator because they will cancel each other.

b-)

The negative tweets that we have in the dataset is . That is why the dataset is skewed towards one class. The percentage of the classes inside the dataset affects the estimation since the estimation might be biased. In addition, because the number of words inside the negative tweets will be more than the other classes. The estimation tends to be on the negative tweets side. We can prevent this by balancing the dataset or guide the solution towards another like we did in the below question (making equal solutions towards neutral class.)

c-)

d-)

I had 57.78% accuracy for multinomial estimation and 1236 wrong answer. Because we have many vectors, estimating via MLE is difficult. If we used gaussian estimation, it should have been better.

e-)

f-)

The accuracy was %57.71 and I had 1238 wrong answer. It is worse than the multinomial since multinomial uses smoothing, and its weights are bigger.

h-)

Codes:

import csv

with open('question-4-train-labels.csv') as csv\_file:

csv\_reader = csv.reader(csv\_file, delimiter=' ')

line\_count = 0

negative\_count = 0

positive\_count = 0

neutral\_count = 0

negative\_list = []

potisive\_list = []

neutral\_list = []

with open('question-4-train-features.csv') as csv\_hop:

hop = csv.reader(csv\_hop, delimiter=' ')

for row in csv\_reader:

for vector in hop:

if row[0] == "negative":

negative\_list.append(vector[0])

negative\_count += 1

break

elif row[0] == "positive":

potisive\_list.append(vector[0])

positive\_count += 1

break

else:

neutral\_list.append(vector[0])

neutral\_count += 1

break

print(f'Negative Count is {negative\_count}.')

print(f'Positive Count is {positive\_count}.')

print(f'Neuteal Count is {neutral\_count}.')

len(potisive\_list[1])

def mutinomial(array\_list, count):

j = 0

p = 11443

n = count

hop = []

while(j < p):

i = 0

sum = 0

while(i < n):

sum = sum + int(array\_list[i][j])

i = i + 1

hop.append(sum)

j = j + 2

print(hop)

return hop

def bernoulli(array\_list, count):

j = 0

p = 11443

n = count

hop = []

while(j < p):

i = 0

sum = 0

while(i < n):

if(int(array\_list[i][j]) > 0):

sum = sum + 1

i = i + 1

hop.append(sum)

j = j + 2

print(hop)

return hop

negative\_multinomial = mutinomial(negative\_list, negative\_count)

positive\_multinomial = mutinomial(potisive\_list, positive\_count)

neutral\_multinomial = mutinomial(neutral\_list, neutral\_count)

negative\_bernoulli = bernoulli(negative\_list, negative\_count)

positive\_bernoulli = bernoulli(potisive\_list, positive\_count)

neutral\_bernoulli = bernoulli(neutral\_list, neutral\_count)

def find\_mean(array\_list):

myInt = 11444

newList = [x / myInt for x in array\_list]

return newList

negativeM\_mean = find\_mean(negative\_multinomial)

positiveM\_mean = find\_mean(positive\_multinomial)

neutralM\_mean = find\_mean(neutral\_multinomial)

neutralM\_mean[0]

negativeB\_mean = find\_mean(negative\_bernoulli)

positiveB\_mean = find\_mean(positive\_bernoulli)

neutralB\_mean = find\_mean(neutral\_bernoulli)

neutralB\_mean[0]

def classify(meanArray, sample, size):

yMean = size / 11712

classification = 1

i = 0

j = 0

hop = 1

while(i < 5722):

classification = classification \* (float(sample[j]) \* float(meanArray[i]) + (1 - float(sample[j])) \* (1 - float(meanArray[i])))

#classification = classification \* (float(sample[j]) \* float(meanArray[i]))

i = i + 1

j = j + 2

hop = classification \* yMean

return hop

sample = []

with open('question-4-test-features.csv') as csv\_file:

csv\_reader = csv.reader(csv\_file, delimiter = ' ')

i = 0

while (i < 2928):

for row in csv\_reader:

sample.append(row[0])

i = i + 1

labels = []

with open('question-4-test-labels.csv') as csv\_file:

csv\_reader = csv.reader(csv\_file, delimiter = ' ')

i = 0

while (i < 2928):

for row in csv\_reader:

labels.append(row[0])

i = i + 1

accurancyM = 0

i = 0

while( i < 2928):

negative = classify(negativeM\_mean, sample[i], negative\_count)

positive = classify(positiveM\_mean, sample[i], positive\_count)

neutral = classify(neutralM\_mean, sample[i], neutral\_count)

if(labels[i] == "negative" and (negative > neutral and negative > positive)):

accurancyM = accurancyM + 1

elif(labels[i] == "positive" and (positive > neutral and positive > negative)):

accurancyM = accurancyM + 1

elif(labels[i] == "neutral" and (neutral > negative and neutral> positive)):

accurancyM = accurancyM + 1

elif(labels[i] == "neutral" and (neutral == negative or neutral == positive or positive == negative)):

accurancyM = accurancyM + 1

i = i + 1

accurancyB = 0

i = 0

while( i < 2928):

negative = classify(negativeB\_mean, sample[i], negative\_count)

positive = classify(positiveB\_mean, sample[i], positive\_count)

neutral = classify(neutralB\_mean, sample[i], neutral\_count)

if(labels[i] == "negative" and (negative > neutral and negative > positive)):

accurancyB = accurancyB + 1

elif(labels[i] == "positive" and (positive > neutral and positive > negative)):

accurancyB = accurancyB + 1

elif(labels[i] == "neutral" and (neutral > negative and neutral> positive)):

accurancyB = accurancyB + 1

elif(labels[i] == "neutral" and (neutral == negative or neutral == positive or positive == negative)):

accurancyB = accurancyB + 1

i = i + 1

print("Multi Accurancy is ")

print(accurancyM/2928)

print(2928 - accurancyM)

print("Bernoulli Accurancy is ")

print(accurancyB/2928)

print(2928 - accurancyB)