Name : Sait Akturk

Section : 02

No : 21501734

For testing code implementation

- **Install numpy and pandas for python3**
- **run code with "python CS464_HW1_2_SAIT_AKTURK.py"**

QUESTION 1

For William Gates

P(X= "Win"| Y = "Bold") = 0.6, P(X = "Lose" | Y = "Bold") = 0.4, P(X = "Draw" | Y = "Timid") = 0.65,

P(X = "Lose" | Y = "Timid") = 0.35

1.1

Result => 2 * P(X= "Win"| Y = "Bold") * P(X = "Lose" | Y = "Bold") * P(X= "Win"| Y = "Bold") + P(X= "Win"| Y = "Bold") * P(X= "Win"| Y = "Bold") = 2 * 0.6 * 0.4 * 0.6 + 0.6 * 0.6 = 0.648

1.2

P(X = "Win" | Y = "Timid") = 0, therefore there is no possibility to win while playing timid style.(The question is not clear, since in the question states that William plays always bold in sudden death, however, in 1.2, the problem says that playing timid in first 3 match, which has sudden death play as well. I am assuming that 1.2 is valid)

1.3

Probability of the winning of Steve is equal to losing for William Gates. Therefore, the problem becomes

P(Y = "Bold" | X = "Lose") = (P(Y = "Bold") * P(X = "Lose" | Y = "Bold")) / (P(X = "Lose")

P(Y = "Bold") = 0.5

P(Y = "Timid") = 0.5

P(X = "Lose") = P(Y = "Bold") * P(X = "Lose" | Y = "Bold") + P(Y = "Timid")* P(X = "Lose" | Y = "Timid")

P(X = "Lose") = 0.5 * 0.4 + 0.5 * 0.35 = 0.375

P(Y = “Bold” | X = “Lose”) = (0.5 * 0.4) / 0.375 = 0.53

1.4

0.75 * 0.85 + 0.25 * (0.35 + 0.4) = 0.825

(I think this question is also ambigious that wrong prediction will cause both players will play same style. For example, if both players play bold, then two different results appear that first the probability of William lose with bold style is 0.4, and second the probability of Steve win with bold style is 0.6. Which one is correct ? I choose only William plays with bold and lose.)

QUESTION 2

2.1

P(S = disease) = 0.005

P(S = healthy) = 0.995

P(T = positive | S = disease) = 0.97

P(T = negative | S = disease) = 0.03

P(T = positive | S = healthy) = 0.02

P(T = negative | S = healthy) = 0.98

2.2

P(S = disease | T = positive) = (P (S = disease) * P (T = positive | S = disease)) / P(T = Positive)

P (T = positive) = P (S = healthy) * P (T = positive | S = healthy) + P (S = disease) * P (T = positive | S = disease)

P (T = positive) = 0.995 * 0.02 + 0.005 * 0.97 = 0.02475

P(S = disease | T = positive) = (0.005 * 0.97) / 0.02475= 0.19595959596

Since given test result is positive, the patient has probability of 19% has disease that means the patients higher probability and should be diagnosed healthy.

QUESTION 3

3.1

mle

→$L_\lambda(x_1, x_2 \ldots, x_n) = \prod_{i=1}^{n}(\ (\lambda^{xi} * e^{-\lambda})/ \ xi!)$

→$K_\lambda \ (x) \ = \ln(\ L_\lambda(x)\ ) = \sum_{i=1}^{n} \text{xi} * \ln(\lambda) - \ \lambda \ - \ln(xi!)$

→$d\ K_\lambda \ (x) \ / \ d\lambda = 0 \rightarrow \sum_{i=1}^{n} \left(\frac{\text{xi}}{\lambda} - \ 1\right) =$ -n + (1\$\lambda$) * $\sum_{i=1}^{n} xi$ → $\lambda \ = \ \sum_{i=1}^{n}(\frac{xi}{n})$

3.2

Map ~ prior * mle

$\theta\lambda$ = *Gamma*(λ) * $L_\lambda(x_1, x_2 \ldots, x_n)$

$\ln(\theta\lambda\ ) = \sum_{i=1}^{n} \text{xi} * \ln(\lambda) - \ \lambda \ - \ln(xi!) \ + \alpha \ln(\beta) + (\alpha - 1) * \ln(\lambda) - \ \beta\lambda \ - ln(\Gamma(\alpha))$

$(\ d \ln(\theta\lambda\ )/ \ d\ \lambda) = 0 \rightarrow \ \sum_{i=1}^{n} \left(\frac{\text{xi}}{\lambda} - \ 1\right) + \frac{(\alpha-1)}{\beta} - \beta = 0 \rightarrow \lambda \ = \frac{\sum_{i=1}^{n}(xi\ ) + \ (\alpha-1)}{\beta+\text{n}}$

3.3

Map ~ prior * mle

$\theta\lambda$ = *Uniform*(a, b) * $L_\lambda(x_1, x_2 \ldots, x_n)$

$\ln(\theta\lambda\ ) = \ln(\text{Uniform(a,b)}) + \ln(L_\lambda(x_1, x_2 \ldots, x_n))$

$d \ln(\theta\lambda\ ) / d\lambda = (d/d\lambda) * \ln(L_\lambda(x_1, x_2 \ldots, x_n))$

Since Uniform(a,b) does not contain variable λ, the derivative of uniform prior will not affect overall derivative result. therefore only variable will affect that is likelihood $\ln(L_\lambda(x_1, x_2 \ldots, x_n))$, therefore maximum likelihood itself.

QUESTION 4

4.1

The denominator of the map is a constant with respect to y, so it has no effect which argument make it maximum. Since, the denominator is same for all class y.

4.2

The ratio of negative tweets to whole classes is 60% in training set. Therefore, the dataset is a skewed dataset. The skewed dataset create biased decision towards to negative class, since calculation of map is required a prior that basically uses the ratio of each class in dataset. Because of this, the decision on test dataset will be biased towards to negative class.

4.3

5722 * 2 + 2 = 11446

4.4

```
import numpy as  np

import pandas as pd

train_features = pd.read_csv("question-4-train-features.csv", sep=',', header = None)

test_features = pd.read_csv("question-4-test-features.csv", sep=',', header = None )

train_labels = pd.read_csv("question-4-train-labels.csv",header=None, sep='\n')

test_labels = pd.read_csv("question-4-test-labels.csv", header = None, sep = '\n')

vocab = pd.read_csv("question-4-vocab.txt", sep = '\t', header = None)

map_labels = {'negative' : 0, 'neutral' : -1, 'positive' : 1}

train_X = train_features.values.astype(np.int32)

train_y = train_labels.values

train_y = np.vectorize(map_labels.get)(train_y).reshape(-1)

test_X = test_features.values

test_y = test_labels.values

test_y = np.vectorize(map_labels.get)(test_y).reshape(-1)

vocab = vocab.values

negative_tweets = train_X[:][ train_y == 0]

neutral_tweets  = train_X[:][ train_y == -1 ]

positive_tweets = train_X[:][ train_y == 1 ]
```

```
estimator = lambda arr : np.log(np.sum(arr, axis=0, keepdims=True) / np.sum(arr))


pi_neg = negative_tweets.shape[0] / train_X.shape[0]
pi_net = neutral_tweets.shape[0]  / train_X.shape[0]
pi_pos = positive_tweets.shape[0] / train_X.shape[0]


estimator_neg = estimator(negative_tweets)
estimator_net = estimator(neutral_tweets)
estimator_pos = estimator(positive_tweets)


def mle(arr,estimator,pi) :
   estimator = np.nan_to_num(estimator)
   mul = np.multiply(arr, estimator)
   mul[ mul < -1e200 ] = float('-inf')
   return np.log(pi) + np.sum(mul,axis=1,keepdims=True)


neg = mle(test_X, estimator_neg, pi_neg)
net = mle(test_X, estimator_net, pi_net)
pos = mle(test_X, estimator_pos, pi_pos)
predictions = np.hstack([net,neg,pos])
max_predictions =np.argmax(predictions, axis=1) -1


acc = max_predictions == test_y
len(acc[ acc == True])/len(acc)
```

4.5

```
estimator_with_prior = lambda arr : np.log((np.sum(arr, axis=0, keepdims=True)+1) / (np.sum(arr)+ 1 *
arr.shape[1]))
```

```
p_est_neg = estimator_with_prior(negative_tweets)
p_est_net = estimator_with_prior(neutral_tweets)
p_est_pos = estimator_with_prior(positive_tweets)

p_neg = mle(test_X, p_est_neg, pi_neg)
p_net = mle(test_X, p_est_net, pi_net)
p_pos = mle(test_X, p_est_pos, pi_pos)
predictions = np.hstack([p_net,p_neg,p_pos])
max_predictions =np.argmax(predictions, axis=1) -1
acc = max_predictions == test_y
len(acc[ acc == True])/len(acc)
```

4.6

```
test_X_ber = test_X.copy()
test_X_ber[ test_X_ber != 0 ] = 1
negative_tweets[ negative_tweets != 0 ] = 1
neutral_tweets[ neutral_tweets != 0 ] = 1
positive_tweets[ positive_tweets != 0 ] = 1

ber_estimator = lambda arr : np.sum(arr, axis = 0, keepdims = True) / arr.shape[0]

neg_ber_est = ber_estimator(negative_tweets)
net_ber_est = ber_estimator(neutral_tweets)
pos_ber_est = ber_estimator(positive_tweets)
```

```
def mle_bernoulli(arr,estimator,pi) : return  np.log(pi) +np.log( np.prod((arr * estimator + (1 - arr) * ( 1 - estimator)), axis=1, keepdims=True))


ber_neg = mle_bernoulli(test_X_ber, neg_ber_est, pi_neg)

ber_net = mle_bernoulli(test_X_ber, net_ber_est, pi_net)

ber_pos = mle_bernoulli(test_X_ber, pos_ber_est, pi_pos)


predictions = np.hstack([ber_net,ber_neg,ber_pos])

max_predictions =np.argmax(predictions, axis=1) -1


acc = max_predictions == test_y

len(acc[ acc == True])/len(acc)
```

4.7

Code:

```
top_words= lambda tweet_class : vocab[np.argsort(np.sum(tweet_class, axis=0))[-20:]][:,0]

negative_tweets_top_words = top_words(negative_tweets)

neutral_tweets_top_words = top_words(neutral_tweets)

positive_tweets_top_words = top_words(positive_tweets)
```

Output :

```
Negative -> ['flightled', 'late', 'gate', 'bag', 'flights', 'hour', 'plane,
       'delayed', 'help', 'customer', 'time', 'hold', 'hours', 'service',
       'cancelled', '@jetblue', '@southwestair', '@usairways', 'flight',
       '@united']
Neutral->['check', 'today', 'travel', 'change', 'fly', 'cancelled', 'flying
','time', 'tomorrow', 'dm', "fleet's", 'fleek', 'help', 'flights',
'@virginamerica', '@usairways', 'flight', '@southwestair','@united', '@jet
blue']
Positive -> ['flying', 'fly', 'today', 'amazing', 'airline', 'help', 'aweso
me','time', 'customer', 'guys', 'best', 'love', 'service','@virginamerica'
, 'great', '@usairways', 'flight', '@united','@jetblue', '@southwestair']
```

Discussion :

The most common words for classes is interpretable since there are some words like ‘late’, ‘delayed’, ‘cancelled’ in negative class; ‘best’, ‘love’, ‘awesome’, ‘amazing in positive class. It is expected to some adjectives for that classes. The models are interpretable because of the independence assumption. It is clear that each feature contribution to model decisions.