Kadir Alat

21502389

Section : 2

HW1

Question 1.1

To have 1 turnover there should be only one winning or only one loosing.

W-W-W-W-W-W-W-L (and permutations of it)

L-L-L-L-L-L-L-W (and permutations of it)

P(Total) = P(only one winning) + P(only one loosing)

$= (8,7)*(0.6)^7*(0.4)+ (8,1)*(0.4)^7*(0.6)$

= 0.09744384

Question 1.2

If player A wants to win the match, s/he needs to win 2 match in the beggining of after loosing a match. In this regard there are two possible combination of series. First one starts with winning, Second one starts with loosing.

First Series : WW – WLWW- WLWLWW-WLWLWLWW- …….

Second Series : LWW-LWLWW-LWLWLWW-LWLWLWLWW-…….

Calculating the probabilities:

First Series:

$p^2 + p*(1-p)*p^2 + (p*(1-p))^2 *p^2 + (p*(1-p))^3 * p^2 + \ldots\ldots\ldots\ldots$

Second Series:

$(1-p)*p^2 + (1-p)^2*p^3 + (1-p)^3*p^4 + \ldots\ldots\ldots\ldots$

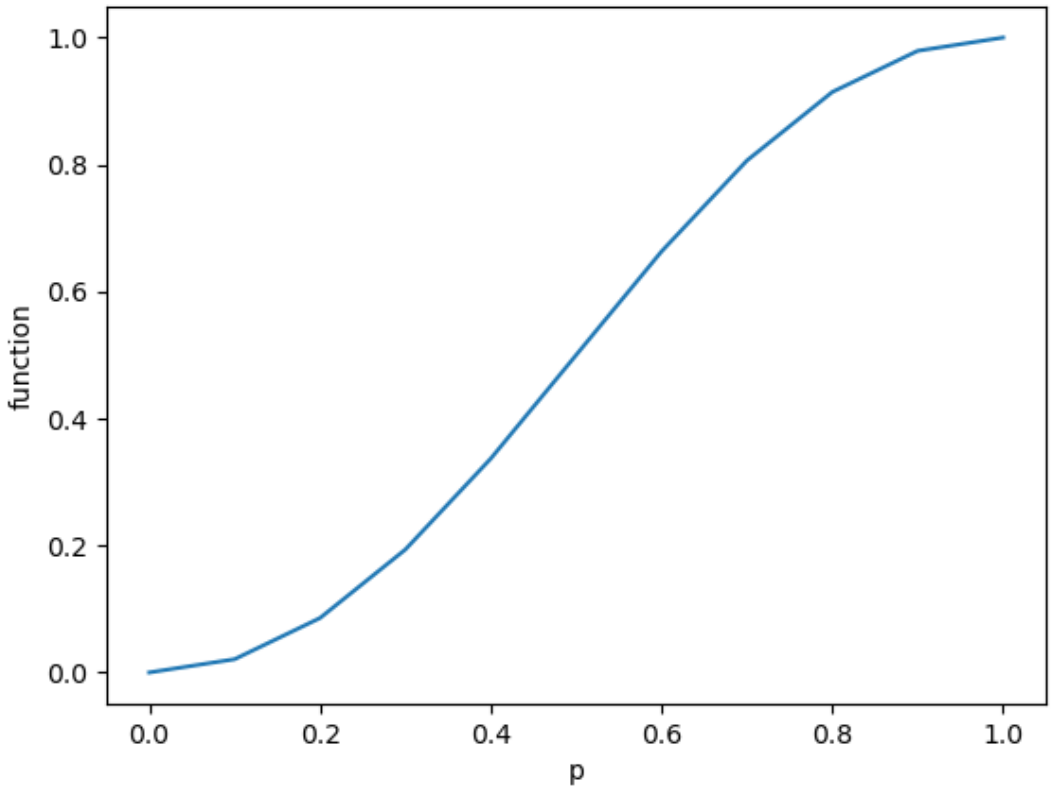
Total Probability = First Series + Second Series

$= p^2(1+ p*(1-p) + (p*(1-p))^2 + (p*(1-p))^3 + \ldots\ldots\ldots)$ +

$(1-p)*p^2(1 + p*(1-p) + (p*(1-p))^2 + \ldots\ldots\ldots)$

$= (p^2 + p^2 – p^3)* (1+ p*(1-p) + (p*(1-p))^2 + (p*(1-p))^3 + \ldots\ldots)$

$= (2p^2 – p^3) / (1 + p^2 - p)$



Question 2

```
import csv
import numpy as np
vocabulary = []

with open('tokenized_corpus.csv','r') as csv_file:
    csv_reader = csv.reader(csv_file)
    sms = []
    for line in csv_reader:
        sms.append(line)
        for token in line :
            if token in vocabulary:
                pass
            else :
                vocabulary.append(token)



temp =0
c=[]
```

```
d=[]
for sentence in sms:
    for words in vocabulary:
        for tokens in sentence:
            if words == tokens:
                temp = temp+1
        c.append(temp)
        temp =0
    d.append(c)
    c = []

with open ('feature_set.csv','w') as new_csv_file:
    csv_writer = csv.writer(new_csv_file,lineterminator = '\n')
    for line in d:
        csv_writer.writerow(line)

tempNum =0
training_set = []
testing_set = []
with open('feature_set.csv','r') as feature_set:
    csv_reader2 = csv.reader(feature_set)
    for line in csv_reader2:
```

```
        if tempNum >= 4460:
            testing_set.append(line)
            tempNum = tempNum + 1
        else:
            training_set.append(line)
            tempNum = tempNum +1


testing_set = np.array(testing_set,dtype=int)
training_set = np.array(training_set,dtype=int)


print(testing_set)


tempNum2 = 0
training_label = []
testing_label=[]
with open('labels.csv','r') as labels:
    csv_reader3 = csv.reader(labels)
    for line in csv_reader3:
        if tempNum2 >=4460:
```

```
            testing_label.append(line)
            tempNum2 = tempNum2 +1
        else:
            training_label.append(line)
            tempNum2 = tempNum2 +1

training_label = np.array(training_label,dtype=int)
testing_label = np.array(testing_label,dtype=int)



def estimate_t(dataset,label,code):
    temp = 0
    t_spam = np.zeros(len(dataset[1]),dtype=int)
    for i in label:
        if i == code:
            t_spam = np.add(t_spam ,
np.array(list(map(int,training_set[temp]))))
        temp = temp +1
    return t_spam
```

```
example = estimate_t(training_set,training_label,1)
example2 = estimate_t(training_set,training_label,0)

def n_spam(training_label):
    temp =0
    for i in training_label:
            if int(i) == 1:
                temp = temp +1
    return temp

num_of_spam = n_spam(training_label)
print("num of span",num_of_spam)

def n_ham(training_label):
    temp =0
    for i in training_label:
            if int(i) == 0:
                temp = temp + 1
    return temp

num_of_ham = n_ham(training_label)
print("num of ham",num_of_ham)
```

```
total_num_of_sms_train = len(training_set)
print("total num of sms in train" , total_num_of_sms_train)

def estimate_any_sms_spam(training,testing):
    total_sms = len(training) + len (testing)
    temp1=0
    temp2=0
    for a in training:
        if int(a) == 1:
            temp1 = temp1 + 1
    for b in testing:
        if int(b) == 1:
            temp2 = temp2 +1
    total_spam = temp1 + temp2
    probability = total_spam / total_sms
    return  probability

ex_pro = estimate_any_sms_spam(training_label,testing_label)
print("estimate any sms spam",ex_pro)
```

```
def theta_spam(training_label,training_set):
    total_spam = n_spam(training_label)
    spam_occurance = estimate_t(training_set,training_label,1)
    probabilities = spam_occurance / total_spam
    return probabilities

ex = theta_spam(training_label,training_set)
print("theta spam",ex)

def theta_ham(training_label,training_set):
    total_ham = n_ham(training_label)
    spam_occurance = estimate_t(training_set,training_label,0)
    probabilities = spam_occurance / total_ham
    return probabilities
ex2 = theta_ham(training_label,training_set)
print(" Theta ham ", ex2)

'''
with open ('deneme2.csv','w') as new_csv_file3:
    csv_writer3 = csv.writer(new_csv_file3,lineterminator = '\n')
    csv_writer3.writerow(ex)
```

```
    csv_writer3.writerow(ex2)
'''


def naive_bayes(set):
    prob_zero = 0
    prob_zero = np.log(n_ham(training_label)/(n_ham(training_label)+n_spam(training_label)))
    t_zero = np.zeros(len(set), dtype=int)
    t_zero = np.add(t_zero,np.array(list(map(int,set))))
    prob = theta_ham(training_label,training_set)
    result = np.multiply(prob,t_zero)
    prob_zero = prob_zero + np.sum(result)
    prob_one = 0
    prob_one = np.log(n_spam(training_label)/(n_ham(training_label)+n_spam(training_label)))
    t_one = np.zeros(len(set), dtype=int)
    t_one = np.add(t_one,np.array(list(map(int,set))))
    prob2 = theta_spam(training_label,training_set)
    result2 = np.multiply(prob2,t_one)
    prob_one = prob_one + np.sum(result2)
    if prob_zero > prob_one:
```

```
        return 1
    else:
        return 0

def accuracy(test_set, true_labels):
  accuracy_count = 0
  for e in range(len(test_set)):
    if naive_bayes(test_set[e]) == true_labels[e]:
      accuracy_count += 1
  accuracy = accuracy_count / len(true_labels)
  return accuracy

accuracy_ = accuracy(testing_set,testing_label)
print(accuracy_)

with open ('test_accuracy.csv','w') as new_csv_file4:
    csv_writer4 = csv.writer(new_csv_file4,lineterminator = '\n')
    csv_writer4.writerow([accuracy_])

accuracy_train = accuracy(training_set,training_label)

with open ('test_accuracy_laplace.csv','w') as new_csv_file5:
```

```
    csv_writer5 = csv.writer(new_csv_file5,lineterminator = '\n')
    csv_writer5.writerow([accuracy_train])
```

Question 3.1

```
set = []
with open('feature_set.csv','r') as feature_set:
    csv_reader2 = csv.reader(feature_set)
    for line in csv_reader2:
        set.append(line)

def new_set(dataset):
    t_new = np.zeros(len(dataset[1]),dtype=int)
    for temp in range(len(dataset)):
        t_new = np.add(t_new , np.array(list(map(int,dataset[temp]))))
    return t_new

x = new_set(set)
```

```
def new_dataset(dataset):
    index=0
    indexes = []
    for element in dataset:
        if element >= 10:
            indexes.append(index)
        index = index + 1
    return  indexes

new_feature_indexes = new_dataset(x)

new_feature_set = []
temp = []
for x in set:
    for a in new_feature_indexes:
        temp.append(x[a])
    new_feature_set.append(temp)
    temp = []

train_new_feature_set = []
```

```
for i in range(4460):
    train_new_feature_set.append(new_feature_set[i])

train_new_feature_set = np.array(train_new_feature_set)

new_feature_set_by_features = train_new_feature_set.transpose()
print("once",train_new_feature_set)
print("sonra",new_feature_set_by_features)




accuracy_list = []

for i in new_feature_set_by_features:
    accuracy_list.append(accuracy(i,training_label))

temp = 0
indices1 = []
for x in accuracy_list:
```

```
    if x > 0.05:
        indices1.append(temp)
    temp = temp +1

with open ('forward_selection.csv','w') as new_csv_file6:
    csv_writer6 = csv.writer(new_csv_file6,lineterminator = '\n')
    csv_writer6.writerow(indices1)
```

Quesiton 3.2

```
trainin_set_frequencey = np.add(example,example2)
print(trainin_set_frequencey)
new_dict = dict(enumerate(trainin_set_frequencey))
print(new_dict)
sorted_frequency = -np.sort(-trainin_set_frequencey)
print(sorted_frequency)

lenght = len(indices1)

indices_required = []
for i in range(lenght):
    for x in new_dict:
```

```
        if new_dict[x] == sorted_frequency[i]:
            indices_required.append(x)


accuracy_list2 = []


for x in indices_required:


accuracy_list2.append(accuracy(new_feature_set_by_features[x],trainin
g_label))


with open ('frequency_selection.csv','w') as new_csv_file10:
    csv_writer10 = csv.writer(new_csv_file10,lineterminator = '\n')
    csv_writer6.writerow(accuracy_list2)
```

Question 4

First Principal Component is the direction of the greatest variability in the data. The reason is two data points that far away from each other might get too close when it comes to low variability. If two points are far away, we still would like to see them far away in reduced space. As an example:

cov(A) = $\begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix}$ Find Eigenvalue and EigenVector

| cov(A) – λ*I | = 0

cov(A) – λ*I = $\begin{bmatrix} 2-\lambda & -4 \\ -1 & -1-\lambda \end{bmatrix}$

| cov(A) – λ*I | = (2- λ)*(-1- λ) – (-1).(-4)

= λ^2- λ-6 = 0

λ = {3,-2} Eigen Values

for λ = 3 -> $\begin{bmatrix} -1 & -4 \\ -1 & -4 \end{bmatrix}$*v1 = $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ (assume v1 = $\begin{bmatrix} x1 \\ x2 \end{bmatrix}$)

-x1 – 4x2 =0 -> x2 =a x1 = -4a

||v1|| = 1 so,

v1 = $\begin{bmatrix} \frac{1}{\sqrt{17}} \\ -\frac{4}{\sqrt{17}} \end{bmatrix}$ eigenvector for eigenvalue λ = 3

for λ = -2 -> $\begin{bmatrix} 4 & -4 \\ -1 & 1 \end{bmatrix}$*v2 = $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ (assume v1 = $\begin{bmatrix} x1 \\ x2 \end{bmatrix}$)

4x1 – 4x2 = 0 -> x1=a x2=a

||v2|| = 1 so,

v2 = $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ eigenvector for eigenvalue λ = -2

2. Variance of v1 = Σ(v1-mean)^2/2

v1 = $\begin{bmatrix} \frac{1}{\sqrt{17}} \\ -\frac{4}{\sqrt{17}} \end{bmatrix}$ mean = -3/2$\sqrt{17}$

Σ(v1-mean)^2/2 = $(\frac{1}{\sqrt{17}} + \frac{3}{2\sqrt{17}})^2/2 + (\frac{-4}{\sqrt{17}} + \frac{3}{2\sqrt{17}})^2/2 = \frac{50}{136}$

Variance of v2 = 0

So,

v1 has bigger variance than v2 and also v1 has bigger eigen value than v2.

Question 4.1

First Principal component is the eigenvector v1 because it has the biggest variance with largest eigenvalue.

Question 4.2

Second Principal Component is the eigenvector v2 because it has the second biggest variance with second largest eigenvalue.