

Home work (STAT. ML 57800): 1

Emmanuel Adebayo

August 2023

1 Problem 1

1. (a) $A = \begin{bmatrix} 2 & 1 & 4 \\ 0 & 3 & 2 \\ 0 & 0 & 5 \end{bmatrix}, B = \begin{bmatrix} 3 & -2 & 2 \\ 1 & 1 & -1 \\ 4 & -1 & 3 \end{bmatrix},$

Then $AB = \begin{bmatrix} (2*3) + (1*1) + (4*4), & (2*-2) + (1*1) + (4*-1), & (2*2) + (1*1) + (4*3) \\ (0*3) + (3*1) + (2*4) & (0*-2) + (3*1) + (2*-1), & (0*2) + (3*-1) + (2*3) \\ (0*3) + (0*1) + (5*4), & (0*-2) + (0*1) + (5*-1), & (0*2) + (0*-1) + (5*3) \end{bmatrix}$

Finally $AB = \begin{bmatrix} 23 & -7 & 15 \\ 11 & 1 & 3 \\ 20 & -5 & 15 \end{bmatrix},$

$BA = \begin{bmatrix} (3*2) + (-2*0) + (2*0), & (3*1) + (-2*3) + (2*0), & (3*4) + (-2*2) + (2*5) \\ (1*2) + (1*0) + (-1*0), & (1*1) + (1*3) + (-1*0), & (1*4) + (1*2) + (-1*5) \\ (4*2) + (-1*0) + (3*0), & (4*1) + (-1*3) + (3*0), & (4*4) + (-1*2) + (3*5) \end{bmatrix},$

which makes $BA = \begin{bmatrix} 6 & -3 & 18 \\ 2 & 4 & 1 \\ 8 & 1 & 29 \end{bmatrix},$ Conclusion: AB is not equal to

BA

$Bu = \begin{bmatrix} (3*1) + (-2*2) + (2*3) \\ (1*1) + (1*2) + (-1*3) \\ (4*1) + (-1*2) + (3*3) \end{bmatrix},$

Bu is $= \begin{bmatrix} 5 \\ 0 \\ 11 \end{bmatrix}$

(b) The original matrix $\begin{bmatrix} 2 & 1 & 4 \\ 0 & 3 & 2 \\ 0 & 0 & 5 \end{bmatrix}$, is already in row echelon form. Here the number of non zero rows is the rank $(A) = 3$

The rank of a matrix is the number of non zero rows in the row echelon form of the matrix. The reduced row echelon form of the A

$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, The rank of A is $\text{Rank}(A) = 3$

(c) The Transpose of A =

$$\begin{bmatrix} 2 & 0 & 0 \\ 1 & 3 & 0 \\ 4 & 2 & 5 \end{bmatrix}$$

(d)

$$uV^T$$

$$\begin{bmatrix} (1 * 3), & (1 * 2), & (1 * 1) \\ (2 * 3), & (2 * 2), & (2 * 1) \\ (3 * 3), & (3 * 2), & (3 * 1) \end{bmatrix}$$

Finally,

$$uV^T$$

$$= \begin{bmatrix} 3 & 2 & 1 \\ 6 & 4 & 2 \\ 9 & 6 & 3 \end{bmatrix},$$

(e) eigenvalues of A

$$\begin{bmatrix} 2-\lambda & 1 & 4 \\ 0 & 3-\lambda & 2 \\ 0 & 0 & 3-\lambda \end{bmatrix},$$

The determinant of the matrix =

$$(2-\lambda)[(3-\lambda) * (5-\lambda)] - 1 * [0 * 5 - \lambda - 0 * 2] + 4[0 * 0 - 3 - \lambda * 0],$$

$$2 - \lambda[(\lambda - 3) * (\lambda - 5)] = 0,$$

Therefore, The eigenvalues are $\lambda = 5, \lambda = 3, \lambda = 2$

(f) The inner product of U and V =
 $(1*3) + (2*2) + (3*1)$
 $= 10$

2 problem 2

1. (α, β) where α added to β is greater than 3 lies in the area above and to the right of the line.

2. The slope of the vector

$$[2, 1]$$

is =

$$\frac{1}{2}$$

while the slope of the function $f(x) = 3 - x$ is -1

the vector has 1 unit up and 2 units to the right slope and $y = 3 - x$ has -1 slope meaning they are not parallel. However, they are perpendicular either because they are not not reciprocal of each other. That is, if a line has a slope m , the perpendicular line will have a slope of

$$\frac{-1}{m}$$

3. The formula for line to origin is in the form:

$$\frac{|a * x_1 + b * y_1 + c|}{\sqrt{a^2 + b^2}}$$

where a = 1, b = 1, and c = -3 and the origin (x,y) = (0,0). so,

$$\frac{|1 * 0 + 1 * 0 - 3|}{\sqrt{1^2 + 1^2}}$$

=

$$\frac{3}{\sqrt{2}}$$

3 problem 3

1. (a) The value for x are respectively $x = 1, 2, 3, 4, 5, 6$ and the $P(X=x)$ for each x is

$$\frac{1}{6}$$

The expectation $E(x)$ is defined as $(1+2+3+4+5+6)$ multiplied by

$$\frac{1}{6}$$

. $E(x) = 3.5$. We can expect $E(x)$ when the dice is rolled
The variance $\text{var}(x)$ is defined as

$$E(x^2) - E(x)^2$$

$$E(x^2)$$

= (

$$1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2$$

) multiplied by

$$1/6$$

= 15.16666667 Thus, $\text{var}(x) = 15.16666667 -$

$$3.5^2$$

The variance = 2.91666667

(b) Now that the $x = 5, 6, 7, 8, 9, 10$, The expected value $E(x)$ is $1/6 * (5+6+7+8+9+10)$. $E(x) = 7.5$ The variance $\text{var}(x)$ is calculated as thus,

$$\begin{aligned}
 & E(x^2) \\
 &= 1/6 * (\\
 & \qquad 5^2 + 6^2 + 7^2 + 8^2 + 9^2 + 10^2 \\
 &) = 59.16666667 \text{ Thus, } \text{var}(x) = 59.16666667 - \\
 & \qquad 7.5^2 \\
 &= 2.91666667
 \end{aligned}$$

(c) 1

The probability of two mutually exclusive occurring simultaneously is zero.

c2

Now that they are not mutually exclusive (i.e mutually inclusive) but independent $P(X = x1, Y = y2) = P(X = x1) * P(Y = y2) = 0.5 * 0.5 = 0.25$

(d) 1

$P(\text{goodquizscore} = \text{yes}, \text{ given review} = \text{yes})$. Here we can use Bayes' theorem. it is of the form $P(A \text{ given } B) = [P(A \text{ and } B)] / P(B)$. Thus here, $P(\text{good quiz score and review} = \text{yes}) / P(\text{review} = \text{yes}) =$

$$\frac{(0.4)}{0.6}$$

$$= 0.4/0.6 = 0.6666667$$

(d)2

It is not equal because the events are not independent. It will hold true, i.e be equal if the occurrence of one event does not influence the other event.

(d)3

we can solve for the $P(\text{good sleep})$ if plug in the equations that is $P(\text{Goodsleep} = \text{yes and Review} = \text{yes, Good quizscore} = \text{yes}) / P(\text{Review} = \text{yes, Good quizscore} = \text{yes}) = 0.7$ $P(\text{Goodsleep} = \text{yes and Review} = \text{yes, Good quizscore} = \text{yes}) = 0.7 * 0.4 = 0.28$

4 problem 4

(a) Using Thread for split

Decision stump type 1: if thread is new, then reads, if thread is old then skips The confusion matrix looks as follows

.	skips	reads
skips	3	0
reads	1	2

The accuracy is $(3+2)/(3+0+1+2) = 5/6$

Decision stump type 2: if thread is old then reads. if thread is new then skips The confusion matrix is as follows,

.	skips	reads
skips	1	2
reads	3	0

The accuracy is $(1+0)/(1+2+3+0) = 1/6$

(b) using Author feature as splitting criteria

Decision stump type 1: If Author is known then reads. if author is unknown, then skips. The accuracy for this stump and the confusion matrix is shown below.

.	skips	reads
skips	1	1
reads	3	1

The Accuracy is $(1+1)/(1+1+3+1) = 2/6$

Decision stump type 2: if author is unknown then reads. if author is known then skips. The accuracy and the confusion matrix is shown below.

.	skips	reads
skips	3	1
reads	1	1

The Accuracy is $(3+1)/(3+1+1+1) = 5/6$

(c) Using Length as a splitting criteria feature

Decision stump type 1: if length is long then reads. if length is short then skips. The accuracy and confusion matrix is shown below.

.	skips	reads
skips	0	2
reads	4	0

The Accuracy is $(0+0)/(0+2+4+0) = 0/6$

Decision stump type 2: if length is short then read. if length is long. then skips. The accuracy and confusion matrix is shown below.

.	skips	reads
skips	4	0
reads	0	2

The Accuracy is $(4+2)/(4+0+0+2) = 6/6$

(d) When we used where as a splitting criteria feature

Decision stump type 1: if where is home then read. if where is work then skips. The confusion matrix and accuracy is shown below.

.	skips	reads
skips	2	1
reads	2	1

The Accuracy is $(2+1)/(2+1+1+2) = 3/6$

Decision stump type 2: if where is work the read. if where is home then skip. The confusion matrix and accuracy is shown below.

.	skips	reads
skips	2	1
reads	2	1

The Accuracy is $(2+1)/(2+1+1+2) = 3/6$

(i) It has a low accuracy rate.

(ii)

This model has failed to generalize and has underfitted.

(iii) both of the decision stump types has the same accuracy rate.

This could.

(iv) finally, it is obvious that because we are using a decision stump, it is too simplistic and it is failing to capture the complex underlying patterns

- (e) The feature with the based accuracy is based on length. Precisely if length is short then reads. if length is long then skips. This has the accuracy of 1. That is 100 percent accuracy rate

- (f) The entropy of tossing a fair coin is $H(V) = - \text{summation } [p(v_k) \log_2 p(v_k)]$. in this case the probability of heads is $P(\text{heads}) = 0.5$ and the probability of tails $p(\text{tails}) = 0.5$. so

$$-[0.5 \log_2 0.5 + 0.5 \log_2 0.5]$$

= 1. The entropy of fair coin is 1

- (g) The information when Thread is the decision feature. First we calculate of the Entropy before split which is

$$\frac{-4}{6} \log_2 \frac{4}{6} - \frac{2}{6} \log_2 \frac{2}{6}$$

= 0.91829. Next we calculate the entropy after split using the "new" attributes. The entropy of "new" is

$$\frac{-2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3}$$

= 0.91829. The entropy of "old" is

$$\frac{-3}{3} \log_2 \frac{3}{3} - \frac{0}{3} \log_2 \frac{0}{3}$$

= 0 Now the weighted average is calculated. as

$$\frac{3}{6} * 0 + \frac{3}{6} * 0.91829$$

= 0.459145 Finally the information gain is the entropy before split minus the weighted average $0.91829 - 0.459145 = 0.459145$

(h) using author as the decision feature for splitting. From (g). we already know the information gain before split which is 0.91829. now the entropy of known is calculated as thus.

$$\frac{-3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4}$$

= 0.81127. The entropy of unknown is calculated as thus,

$$\frac{-1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2}$$

= 1 The weighted average is

$$\frac{4}{6} * 0.81127 + \frac{2}{6} * 1$$

= 0.87418 The information gain = 0.91829 - 0.87418 = 0.04411

- (i) Using length as decision feature for splitting. From previous question we already know the entropy before splitting which is 0.91829.

The entropy for short is calculated as thus,

$$\frac{-2}{2} \log_2 \frac{2}{2} - \frac{0}{2} \log_2 \frac{0}{2}$$

= 0 The entropy for long is calculated as thus,

$$\frac{-4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4}$$

= 0 The weighted average is calculated as

$$\frac{4}{6} * 0 + \frac{2}{6} * 0$$

= 0 The information gain is 0.91829 - 0 = 0.91829.

- (j) Using where as decision feature for splitting The entropy before split is known to be 0.91829. The entropy for home is calculated as,

$$\frac{-2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3}$$

= 0.91829 The entropy for work is calculated as

$$\frac{-2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3}$$

= 0.91829. The weighted average is

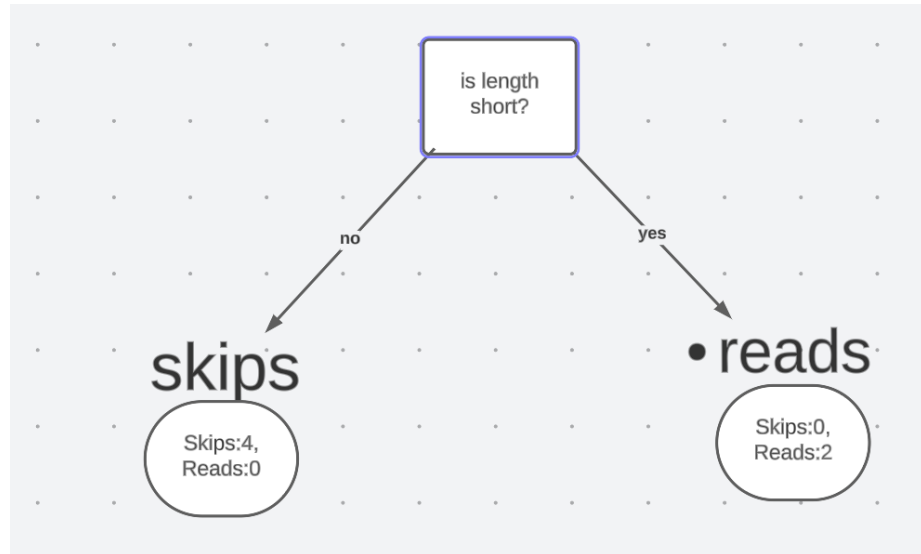
$$\frac{3}{6} * 0.91829 + \frac{3}{6} * 0.91829$$

= 0 The information is 0.91829 - 0.91829 = 0

- (k) the decision stump that gives the best information gain is based on length where an information of 0.91829. Precisely if length is short then read. if length is long then skips.

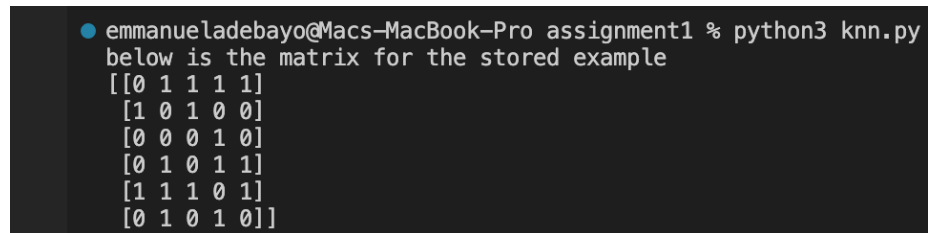
- (l) Both (e) and (k) both showed that they both are the best decision feature for splitting. They are the same. precisely, "length" as feature came out on top when using accuracy and information gain as a decision feature for splitting

(m) The decision stump using length as decision feature



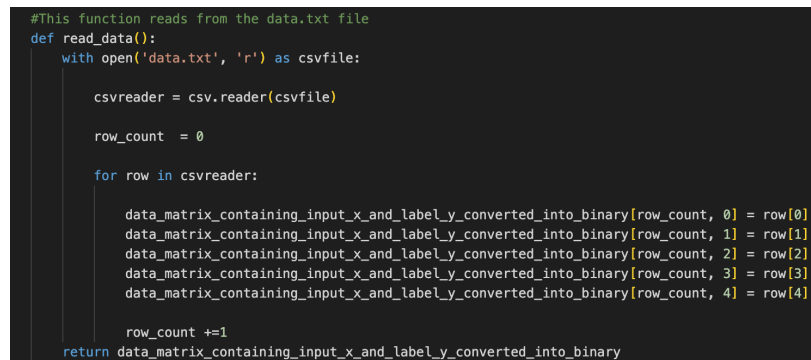
5 problem 5

- (a) I started with a file called "pre-processed-data.txt" which has the data unprocessed. That is, it is in the original form as given by the professor. I then follow the instruction and wrote the required function (read-data). The function reads from the processed data (data.txt), with each feature converted into binary values using an helper function, store it in a numpy matrix for later use and fast accessibility. The attribute reads, known, new, long, home are converted into 1s and the rest are converted into 0s The image for the converted matrix from data.txt is shown below



```
emmanueladebayo@Macs-MacBook-Pro assignment1 % python3 knn.py
below is the matrix for the stored example
[[0 1 1 1 1]
 [1 0 1 0 0]
 [0 0 0 1 0]
 [0 1 0 1 1]
 [1 1 1 0 1]
 [0 1 0 1 0]]
```

The code screenshot is also shown



```
#This function reads from the data.txt file
def read_data():
    with open('data.txt', 'r') as csvfile:
        csvreader = csv.reader(csvfile)

        row_count = 0

        for row in csvreader:
            data_matrix_containing_input_x_and_label_y_converted_into_binary[row_count, 0] = row[0]
            data_matrix_containing_input_x_and_label_y_converted_into_binary[row_count, 1] = row[1]
            data_matrix_containing_input_x_and_label_y_converted_into_binary[row_count, 2] = row[2]
            data_matrix_containing_input_x_and_label_y_converted_into_binary[row_count, 3] = row[3]
            data_matrix_containing_input_x_and_label_y_converted_into_binary[row_count, 4] = row[4]

            row_count +=1
        return data_matrix_containing_input_x_and_label_y_converted_into_binary
```

- (b) The hamming distance function is written and it is later used to calculate distance between the examples. The image for the hamming distance for all example versus the test sample (unseen sample) is shown

```
The distance of the unseen sample to each stored example is computed as thus using the hamming distance function
example: 3, Distance: 1
example: 4, Distance: 1
example: 1, Distance: 2
example: 6, Distance: 2
example: 2, Distance: 3
example: 5, Distance: 3
```

The code for the hamming distance is also shown

```
#This function calculates the distance between two vectors
def distance(vector_1, vector_2):
    #convert to numpy array
    vec1 = np.array(vector_1)
    vec2 = np.array(vector_2)

    dist = 0

    #checks if the vector are not of the same length. hamming distance cannot be calculated
    #else proceeds to calculate the hamming distance
    #How many points where they do not have the same value
    if (len(vec1) != len(vec2)):
        print("vectors are not of the same length!")
    else:
        for i in range(len(vec1)):
            if vec1[i] != vec2[i]:
                dist += 1
        return dist
```

- (c) The code for the knn predict is also included in the file The screenshot of the code and output are shown below

```
#The actual function
#Input: D is the data matrix (6 * 5 numpy array)
#input: K is the parameter for the number of neighbors to use
#input: x_hat is the unseen label to be predicted
def KNN_Predict(D, K, x_hat):
    print("below is the matrix for the stored example")
    print(D)
    if K > 6:
        print("K exceeds total number of stored data")
        return
    if K < 1:
        print("You must choose a k")
        return

    #The s stores the computed distance between each example and the unseen vector
    s = {}

    #These array extract each example
    #it is passed on the hamming distance function to calculate the distance
    extracted_vector_row_by_row = []

    #loop through the data D matrix
    for i in range(num_rows):
        for j in range(num_column):
            if j == 0:
                continue
            extracted_vector_row_by_row.append(D[i,j])

    #calculating the hamming distance for the ith example
    hamming_distance = distance(extracted_vector_row_by_row, x_hat)
    #store the hamming distance into the distance dictionary
    s[i] = hamming_distance
    #clear the row to allow space for the following example
    extracted_vector_row_by_row.clear()
```

```

#after the distances have been calculated
#sort the distances
s = sort_dictionary(s)

#beginning of calculating the prediction
#it is initialized to zero
y_hat = 0

#loop through the s distances map k times which is the number of specified neighbors
#use voting to get the predicted values

#count to check k
count = 0
#this is used to store the output label from the neighbors
#not necessary but for visualization
added_values = []
for key, value in s.items():
    #using count to make sure count is less than k
    #if it equal to k or more than, we have gone past
    #the number of allowed neighbors
    if count < K:

        #go to the Data D matrix
        #get the corresponding output label
        y_label = D[key, 0]

        #if it is zero, convert it to -1. because of voting
        if y_label == 0:
            y_label = -1

        added_values.append(y_label)
        #calculate the y_hat by voting
        y_hat = y_hat + y_label
        count +=1

```

```

        y_label = -1

        added_values.append(y_label)
        #calculate the y_hat by voting
        y_hat = y_hat + y_label
        count +=1
    else:
        break
#use the sign function to convert the outcome to a desired output
predicted_label = sign(y_hat)
#return the distance map and the predicted label
return s, predicted_label

```

The predicted value for $k = 3$ is -1

- (d) The test sample is read from a function and the knn predict function is called The code screen shot is also included

```
#The function reads the unseen sample
#from the file test_sample.txt
def read_test_sample():
    data_list = []
    with open('test_sample.txt', 'r') as file:
        csvreader = csv.reader(file)

        for line in file:
            # Split the line into an array using commas as the delimiter
            line_data = line.strip().split(",")

            # Append the line data to the list
            data_list.append(line_data)
    data_list = np.array(data_list).flatten()
    data_list = data_list.astype(int)
    return data_list
```

(e) when $k = 1$, the prediction is -1. which is skips

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● emmanueladebayo@Macs-MacBook-Pro assignment1 % python3 knn.py
below is the matrix for the stored example
[[0 1 1 1]
 [1 0 1 0]
 [0 0 1 0]
 [0 1 0 1]
 [1 1 0 1]
 [0 1 0 1]]
The distance of the unseen sample to each stored example is computed as thus using the hamming distance function
example: 3, Distance: 1
example: 4, Distance: 1
example: 1, Distance: 2
example: 6, Distance: 2
example: 2, Distance: 3
example: 5, Distance: 3
The predicted value for k = 1 is -1
It is predicted to be skips
○ emmanueladebayo@Macs-MacBook-Pro assignment1 %
```


(f) when $k = 3$, the prediction is -1. which is skips

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● emmanueladebayo@Macs-MacBook-Pro assignment1 % python3 knn.py
below is the matrix for the stored example
[[0 1 1 1 1]
 [1 0 1 0 0]
 [0 0 1 0 0]
 [0 1 0 1 1]
 [1 1 0 1 1]
 [0 1 0 1 0]]
The distance of the unseen sample to each stored example is computed as thus using the hamming distance function
example: 3, Distance: 1
example: 4, Distance: 1
example: 1, Distance: 2
example: 6, Distance: 2
example: 2, Distance: 3
example: 5, Distance: 3
The predicted value for k = 3 is -1
It is predicted to be skips
○ emmanueladebayo@Macs-MacBook-Pro assignment1 %
```