Graded Questions (Naïve Bayes):

1. Bayes Theorem and Its Building Blocks

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

1. Naïve Bayes for Categorical Data

**Graded Questions**

In this segment, you will solve questions on 'Naive Bayes' and **will be graded on your answers.**

**Comprehension - Spam and Ham E-Mails**

Bayesian classifiers are often used for document classification. The words in the documents are used as features for classification. For example, if you want to classify emails as spam or ham (genuine mail), you can use the ‘frequency of words in the text of an email’ as the features. The grammar is disregarded, which means that *unimportant*words like it, there, the, and etc.  are ignored.

For example, if the main text of an email is:

“Best offers on weight loss fitness bands! **Buy** this weekend to get a **free** protein supplement too!!**Limited** stock, **buy** now and get **free** stuff! Hurry up! For more **free** offers, subscribe on the link below.”

Then the **frequently** occurring words i.e. the**most important keywords (features)** can be counted and stored in a table as shown below. The email above (obviously spam) is shown in the first row of the table. **Freq 1** is the **most frequent word; Freq 2**is the **second most frequent word etc**. Also, note that the **order of features is important.** If the features are (free, report, buy, click), in that order, then ‘free’ is ‘Freq 1’, ‘report’ is Freq 2 and so on. Which means that  (report, free, buy, click) is a different observation from (free, report, buy, click).

The data set with features and class labels is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table 5: Most Occurring Words | | | | | |
| **S.No** | **Class** | **Freq 1** | **Freq 2** | **Freq 3** | **Freq 4** |
| 1. | **Spam** | free | buy | limited | hurry |
| 2. | Ham | reply | data | report | presentation |
| 3. | Ham | report | presentation | file | end of day |
| 4. | **Spam** | limited | file | buy | click |
| 5. | Ham | meeting | timelines | limited | documents |
| 6. | **Spam** | hurry | data | buy | stock |
| 7. | **Spam** | limited | sex | click | viagra |
| 8. | Ham | presentation | end of day | data | report |
| 9. | Ham | reply | data | presentation | click |
| 10. | **Spam** | free | reply | weekend | click |
| 11. | **Spam** | limited | click | free | hurry |
| 12. | Ham | meeting | end of day | weekend | data |
| 13. | **Spam** | hurry | weekend | stock | offer |
| 14. | Ham | report | presentation | file | end of day |
| 15. | Ham | free | timelines | reply | offer |

Let’s assume a simplified scenario where spammers use only the following important words in their emails:

**Spam Keywords:** buy, free, hurry, weekend, stock, offer, viagra, sex, limited, click

Also, assume that you are building a model for an organisation where the only important words in genuine (ham) emails are as follows:

**Ham Keywords:** reply, data, report, presentation, file, end of day, meeting, timelines, delay, documents

**Note:**Wherever you come across the word independent/independence in this module, conditional independence is implied as discussed in the previous segment "*Conditional Independence in Naive Bayes".*

**NOTE:**

1. **Use the above-given table to answer the following questions.**
2. **To Solve the question, you need to very careful with the order of the features, For example: If my given feature list is(free, data, weekend, click) then free is freq1, data is freq2 and so on. Hence the probability of P(free | spam) will be 2/7 and P(click | spam) will be 2/7.**

A screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

1. Naïve Bayes for Text Classification

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

**Graded Questions**

In this segment, you will use the [IMDB](http://www.imdb.com/) movie reviews dataset to classify reviews as 'positive' or 'negative'. We have divided the data into training and test sets. The training set contains 800 positive and 800 negative movie reviews whereas the test set contains 200 positive and 200  negative movie reviews.

This was one of the first widely-available sentiment analysis datasets compiled by Pang and Lee's. The data was first collected in 2002, however, the text is similar to movies reviews you find on IMDB today. The dataset is in a CSV format. It has two categories: Pos (reviews that express a positive or favourable sentiment) and Neg (reviews that express a negative or unfavourable sentiment). For this exercise, we will assume that all reviews are either positive or negative; there are no neutral reviews.

You will need to build a Multinomial Naive Bayes classification model in Python for solving the questions.

Please download the datasets from below:

**Note:**

* **Tag negative(Neg) as 0 and positive(Pos) as 1.**

**[imdb\_train\_dataset](https://cdn.upgrad.com/UpGrad/temp/049d2f51-7903-4fc3-a7b4-5ae2f1eb6968/movie_review_train.csv" \o "movie_review_train.csv" \t "_blank)**

[file\_download](https://cdn.upgrad.com/UpGrad/temp/049d2f51-7903-4fc3-a7b4-5ae2f1eb6968/movie_review_train.csv" \o "movie_review_train.csv" \t "_blank)**[Download](https://cdn.upgrad.com/UpGrad/temp/049d2f51-7903-4fc3-a7b4-5ae2f1eb6968/movie_review_train.csv" \o "movie_review_train.csv" \t "_blank)**

**[imdb\_test\_dataset](https://cdn.upgrad.com/UpGrad/temp/93a8a3f6-b8c7-4e1a-8ce1-5734df27e875/movie_review_test.csv" \o "movie_review_test.csv" \t "_blank)**

[file\_download](https://cdn.upgrad.com/UpGrad/temp/93a8a3f6-b8c7-4e1a-8ce1-5734df27e875/movie_review_test.csv" \o "movie_review_test.csv" \t "_blank)**[Download](https://cdn.upgrad.com/UpGrad/temp/93a8a3f6-b8c7-4e1a-8ce1-5734df27e875/movie_review_test.csv" \o "movie_review_test.csv" \t "_blank)**

Please answer the following questions based on the model you make using above datasets:

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

1. Coding Graded:

Question 1

**Sorting list of tuples**

You are provided with a list of tuples having three elements, name, age and salary. You are required to arrange the tuples in order of any of the three elements. The variable (name/age/salary) has to be decided by the second input, which is an integer (1 : name, 2 : Age, 3 : Salary). Make sure that the ordering should be ascending.

**Example:**

**Input 1 : [('Ram', 23 , 3000) , ('Mohan' , 22 , 4000 ) , ( 'Suresh' , 19 , 8000)]**

**2**

**Output 1:[ ( 'Suresh' , 19 , 8000) , ('Mohan' , 22 , 4000 ) ,('Ram', 23 , 3000)]**

**Input 2 : [('Ram', 23 , 3000) , ('Mohan' , 22 , 4000 ) , ( 'Suresh' , 19 , 8000), ('Sita' , 28,2500)]**

**3**

**Output 2:[ ('Sita' , 28,2500), ('Ram', 23 , 3000), ('Mohan' , 22 , 4000 ) , ( 'Suresh' , 19 , 8000)]**

Solution:

import ast,sys

input\_str = input()

input\_list = ast.literal\_eval(input\_str)

sorting\_choice = input()

# Your code goes here

def sort\_tup(tup,n):

tup.sort(key = lambda x: x[n])

return tup

print(sort\_tup(input\_list,int(sorting\_choice)-1))

Question 2

**What's Missing?**

For a given list of integers in increasing order from n to m with no duplicates and common difference between two consecutive numbers being 1 except where the number is missing  with total  m-n integers in list, find out the missing number.  
  
Sample input : [ 8,9,10,12}  
Sample Output : 11

Solution:

# Import the relevant libraries

import ast,sys

# Read the input list

input\_str = sys.stdin.read()

input\_list = ast.literal\_eval(input\_str)

# Write your code here

def missing\_elements(L):

start, end = L[0], L[-1]

return sorted(set(range(start, end + 1)).difference(L))

print(''.join(str(x) for x in missing\_elements(input\_list)))

Question 3

**Finding common letters in string**

You are provided two strings. You have to determine the common letters in the two strings. Make sure that the case difference is kept in mind. 'A' is not the same as 'a'. Also, note that the common letters are ordered in their alphabetical order.  
  
**Examples:  
Input 1:  
    S1  = 'Hello'  
   S2  = 'World'  
Output 1:  
  The common letters are :  
   l  
  o  
  
Input 2:  
   S1 = 'abc'  
   S2 = 'def'  
Output 2:  
  There are no common letters between them**

Solution:

# Take the strings as input

s1=input()

s2=input()

# Write your code below

a=list(set(s1)&set(s2))

a.sort()

if len(a) :

print('The common letters are:')

for i in a:

print(i)

else:

print('There are no common letters between them')

Question 4

**Join String**

Given a list of strings, join them using the delimiter  '-'   
  
Sample Input : ['Accountability', 'and', 'Ownership']  
  
Sample Output : Accountability-and-Ownership

Solution:

# Import the relevant libraries

import ast,sys

# Read the input list

input\_str = sys.stdin.read()

input\_list = ast.literal\_eval(input\_str)

# Write your code here

print("-".join(input\_list))