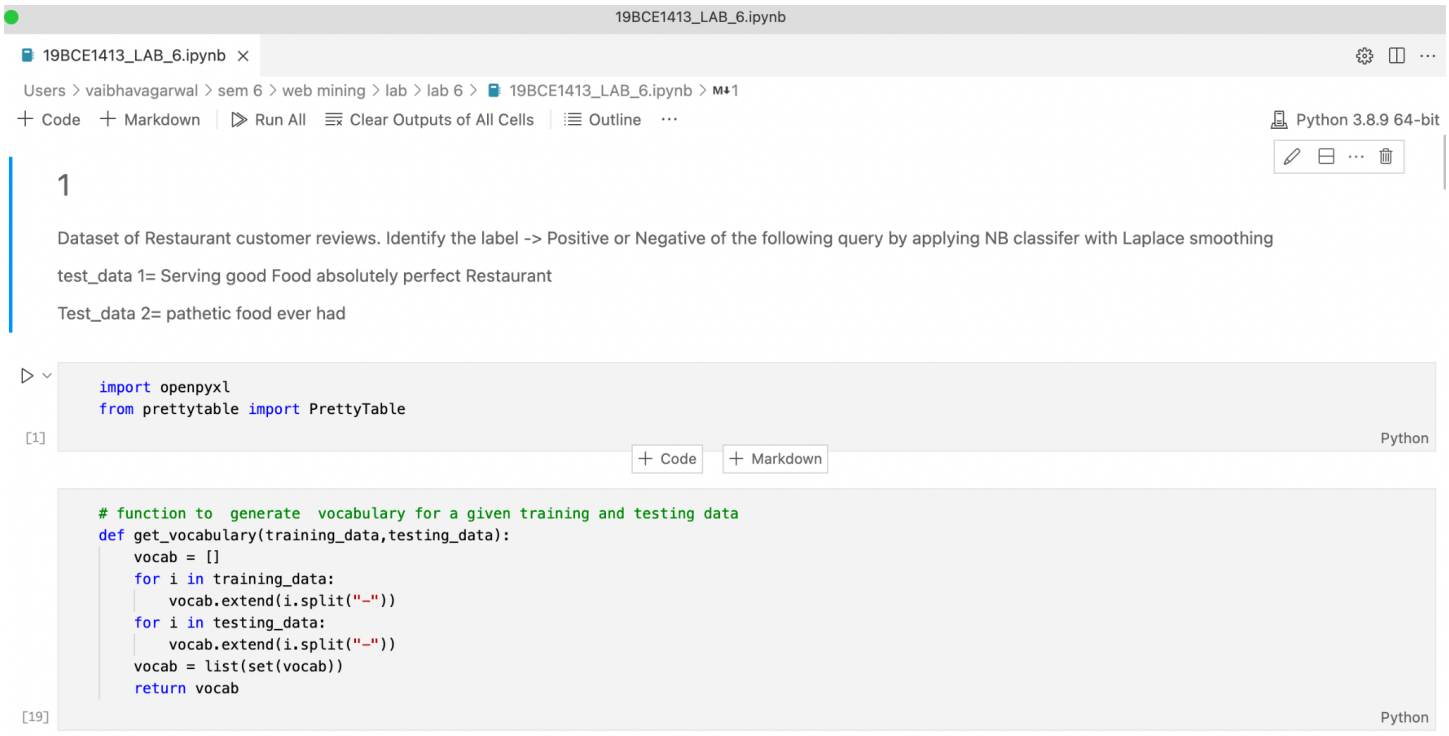
Programme	:	**B.Tech** Semester : **Win Sem 21-22**
Course	:	**Web Mining Lab** Code : **CSE3024**
Faculty	:	**Dr.Bhuvaneswari A** Slot : **L7+L8**
Date	:	**25-02--2022** Marks : **10 Points**

Vaibhav Agarwal

19BCE1413



```python
wb = openpyxl.load_workbook("data.xlsx")
sheet = wb.active #selecting the sheet

training_data = [] #to store the training data
training_data_labels = [] #to store the training data classes/labels

testing_data = [] #to store the test data
```

[20] Python

```python
#fetch  training data from the excel
for i in range(0,9):
    c1 = sheet.cell(i+3,2)
    c2 = sheet.cell(i+3,3)
    training_data.append(str(c1.value))
    training_data_labels.append(str(c2.value))
```

[4] Python

```python
#fetching the testing data from the excel
for i in range(0,2):
    c = sheet.cell(i+12,2)
    testing_data.append(str(c.value))
```

[6] Python

```python
#printing the training data
train_tb = PrettyTable(["training data","Labels"])
for i in range(0,len(training_data)):
    train_tb.add_row([training_data[i],training_data_labels[i]])
print(train_tb)
```

[7] Python

```
+-----------------------------------------+----------+
|               training data             |  Labels  |
+-----------------------------------------+----------+
|             Simply-loved-it             | Positive |
|    Most-disgusting-Food-I-have-ever-had | Negative |
|       Stay-away-very-disgusting-Food    | Negative |
|    menu-is-absolutely-perfect-loved-it  | Positive |
|       a-really-good-value-for-money     | Positive |
|       this is-a-very-good-Restaurant    | Positive |
|            terrible-experience          | Negative |
|          this-place-has-best-food       | Positive |
| this-place-has-most-pathetic-Serving-food | Negative |
+-----------------------------------------+----------+
```

+ Code + Markdown

```
    #printing the testing data
    test_tb = PrettyTable(["Testing Data","Label"])
    for i in range(0,len(testing_data)):
        test_tb.add_row([testing_data[i],"?"])
    print(test_tb)
```

[8] Python

```
+-------------------------------------------------+-------+
|                  Testing Data                   | Label |
+-------------------------------------------------+-------+
| Serving-good-Food-absolutely-perfect-Restaurant |   ?   |
|              pathetic-food-ever-had             |   ?   |
| Serving-good-Food-absolutely-perfect-Restaurant |   ?   |
|              pathetic-food-ever-had             |   ?   |
+-------------------------------------------------+-------+
```

```
    #generating the vocabulary from the given training and testing data
    vocabulary = get_vocabulary(training_data,testing_data)
    print("\nVocabulary Size = ",len(vocabulary))
    print("Vocabulary:\n",vocabulary)
```

[9] Python

```
Vocabulary Size =  35
Vocabulary:
 ['loved', 'had', 'Stay', 'terrible', 'has', 'it', 'Food', 'for', 'Serving', 'money', 'menu', 'this is', 'is', 'disgusting', 'Simply', 'very',
'Restaurant', 'most', 'experience', 'perfect', 'pathetic', 'absolutely', 'value', 'a', 'really', 'good', 'food', 'away', 'I', 'best', 'have', 'Most',
'place', 'this', 'ever']
```

```
    #computing the prior class probabilities
    prior_class_probs = []
    a = 0
    b = 0
    c = 0
    for i in range(0,len(training_data)):
        if training_data_labels[i] == "Positive":
            a = a + 1
        elif training_data_labels[i] == "Negative":
            b = b + 1
    prior_class_probs.append(a/len(training_data)) #Positive
    prior_class_probs.append(b/len(training_data)) #Negative
    print("\nPrior Class Probabilities: ",prior_class_probs)
```

[10] Python

```
Prior Class Probabilities:  [0.5555555555555556, 0.4444444444444444]
```

```
    #generating the dictionary for ever class containing all the words and their frequency
    Positive_dict = {}
    Negative_dict = {}
```

[11] Python

+ Code + Markdown

```
#generating dictionary for Positive class
Positive_all_words = []
for i in range(0,len(training_data)):
    if training_data_labels[i] == "Positive":
        Positive_all_words.extend(training_data[i].split("-"))

for i in list(set(Positive_all_words)):
    Positive_dict[i] = [Positive_all_words.count(i)]

print("\n'Positive' Class Dictionary:\n",Positive_dict)
```

[12] Python

```
'Positive' Class Dictionary:
 {'loved': [2], 'has': [1], 'it': [2], 'for': [1], 'money': [1], 'menu': [1], 'this is': [1], 'is': [1], 'Simply': [1], 'very': [1], 'Restaurant': [1],
'perfect': [1], 'absolutely': [1], 'value': [1], 'a': [2], 'really': [1], 'good': [2], 'food': [1], 'best': [1], 'place': [1], 'this': [1]}
```

```
#generating dictionary for Negative class
Negative_all_words = []
for i in range(0,len(training_data)):
    if training_data_labels[i] == "Negative":
        Negative_all_words.extend(training_data[i].split("-"))

for i in list(set(Negative_all_words)):
    Negative_dict[i] = [Negative_all_words.count(i)]

print("\n'Negative' Class Dictionary:\n",Negative_dict)
```

[13] Python

```
'Negative' Class Dictionary:
 {'had': [1], 'Stay': [1], 'terrible': [1], 'has': [1], 'Food': [2], 'Serving': [1], 'disgusting': [2], 'very': [1], 'most': [1], 'experience': [1],
'pathetic': [1], 'food': [1], 'away': [1], 'I': [1], 'have': [1], 'Most': [1], 'place': [1], 'this': [1], 'ever': [1]}
```

```
#calculating conditional probabilities of each word belonging in Positive class
for i in list(Positive_dict.keys()):
    p = (Positive_dict[i][0] + 1)/(len(Positive_all_words) + len(vocabulary))
    Positive_dict[i].append(round(p,4)) #storing the proabality in the dictionary itself
```

[14] Python

```
#calculating conditional probabilities of each word belonging in Negative class
for i in list(Negative_dict.keys()):
    p = (Negative_dict[i][0] + 1)/(len(Negative_all_words) + len(vocabulary))
    Negative_dict[i].append(round(p,4)) #storing the proabality in the dictionary itself
```

[15] Python

```python
testing_data_labels = [] #to store the test data documents classes/labels
#calculating the probablitiy of the choosen test document
for i in range(0,len(testing_data)):
    #tokenising the document
    test_doc_words = []
    test_doc_words.extend(testing_data[i].split("-"))
    p_Positive = prior_class_probs[0] #assigning the prior probab of class positive
    for j in range(0,len(test_doc_words)):
        #if word is in the dictionary then directly fetch the probability
        if test_doc_words[j] in Positive_dict.keys():
            p_Positive = p_Positive * Positive_dict[test_doc_words[j]][1]
        #if word is not there in the dictionary compute the probability
        else:
            #print("In else positive class,"+test_doc_words[j]+" not there in dict")
            p_Positive = p_Positive * (1/(len(Positive_all_words)+len(vocabulary)))

    #calculating the probab if the doc comes under Negative class
    p_Negative = prior_class_probs[1] #assigning the prior probab of class Negative
    for j in range(0,len(test_doc_words)):
        #if word is in the dictionary then directly fetch the probability
        if test_doc_words[j] in Negative_dict.keys():
            p_Negative = p_Negative * Negative_dict[test_doc_words[j]][1]
        #if word is not there in the dictionary compute the probability
        else:
            #print("In else Negative class,"+test_doc_words[j]+" not there in dict")
            p_Negative = p_Negative * (1/(len(Negative_all_words)+len(vocabulary)))


    if p_Positive > p_Negative:
        #print("Assigned to Positive")
        testing_data_labels.append("Positive")
    elif p_Positive < p_Negative:
        #print("Assigned to Negative")
        testing_data_labels.append("Negative")
```

Python

```python
final_tb = PrettyTable(["Testing Data","Label"])
for i in range(0,len(testing_data)):
    final_tb.add_row([testing_data[i],testing_data_labels[i]])
print(final_tb)
```

[17] Pytho

```
+-------------------------------------------------+----------+
|                  Testing Data                   |  Label   |
+-------------------------------------------------+----------+
| Serving-good-Food-absolutely-perfect-Restaurant | Positive |
|              pathetic-food-ever-had             | Negative |
| Serving-good-Food-absolutely-perfect-Restaurant | Positive |
|              pathetic-food-ever-had             | Negative |
+-------------------------------------------------+----------+
```

2

The dataset . Create your own dataset as data.csv file Identify the class/category è Politics or Business or Sports - of the following query by applying NB classifer with Laplace smoothing

(i) query_data = [4,0,2,0,1,0,6,0]

(ii) query_data = [0,0,2,0,0 ,9,0,9]

(iii) query_data = [5,0,2,5,0 ,9,0,9]

```python
import numpy as np
import pandas as pd
from csv import reader
```

[17] Python

```python
data = pd.read_csv(r'lab6.csv')
```

[18] Python

```
data.info()
```

[19] Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   TDP       7 non-null      int64
 1   Nifty     7 non-null      int64
 2   Sidhu     7 non-null      int64
 3   BJP       7 non-null      int64
 4   Sensex    7 non-null      int64
 5   Sixer     7 non-null      int64
 6   Congress  7 non-null      int64
 7   Century   7 non-null      int64
 8   Category  7 non-null      object
dtypes: int64(8), object(1)
memory usage: 632.0+ bytes
```

```
data.columns
```

[20] Python

```
Index(['TDP', 'Nifty', 'Sidhu', 'BJP', 'Sensex', 'Sixer', 'Congress',
       'Century', 'Category'],
      dtype='object')
```

```
data.head(10)
```

Python

	TDP	Nifty	Sidhu	BJP	Sensex	Sixer	Congress	Century	Category
0	4	0	3	5	1	0	6	0	Politics
1	0	5	0	2	6	0	1	0	Business
2	0	0	6	1	0	4	1	2	Sports
3	4	1	0	1	1	0	6	0	Politics
4	0	0	0	0	0	5	0	6	Sports
5	0	4	0	2	6	0	0	1	Business
6	5	0	0	3	0	0	5	0	Politics

```
query_data= [[4,0,2,0,1,0,6,0],[0,0,2,0,0,9,0,9],[5,0,2,5,0,9,0,9]]
```

Python

```
outputlabels = data['Category'].unique()
words = list(data.columns)[:-1]
numtraindocuments = data.shape[0]
```

Python

```
print(outputlabels)
print(words)
print(numtraindocuments)
```

Python

```
['Politics' 'Business' 'Sports']
['TDP', 'Nifty', 'Sidhu', 'BJP', 'Sensex', 'Sixer', 'Congress', 'Century']
7
```

```python
conditional_probability = {}
probability = {}
```

Python

```python
for outputClass in outputlabels:
    temp_dataframe = data.loc[data['Category']==outputClass]
    probability[outputClass]=(temp_dataframe.shape[0]/numtraindocuments)
```

Python

```python
print(probability)
```

Python

```
{'Politics': 0.42857142857142855, 'Business': 0.2857142857142857, 'Sports': 0.2857142857142857}
```

```python
ALPHA = 1
```

Python

```python
for outputClass in outputlabels:
    temp_dataframe = data.loc[data['Category']==outputClass]
    total_word_count_in_category =0
    for i in range(temp_dataframe.shape[0]):
        for word in words:
            total_word_count_in_category += temp_dataframe.iloc[i][word]
    for word in words:
        current_word_count_in_category =0
        for i in range(temp_dataframe.shape[0]):
            current_word_count_in_category += temp_dataframe.iloc[i][word]
            cur_prob = (current_word_count_in_category + ALPHA) / (total_word_count_in_category)
            conditional_probability[(word, outputClass)] = cur_prob
```

[29] Python

```python
print("Conditional probability after applying smoothing\n")
conditional_probability
```

[30] Python

```
... Conditional probability after applying smoothing


{('TDP', 'Politics'): 0.3111111111111111,
 ('Nifty', 'Politics'): 0.044444444444444446,
 ('Sidhu', 'Politics'): 0.08888888888888889,
 ('BJP', 'Politics'): 0.2222222222222222,
 ('Sensex', 'Politics'): 0.06666666666666667,
 ('Sixer', 'Politics'): 0.022222222222222223,
 ('Congress', 'Politics'): 0.4,
 ('Century', 'Politics'): 0.022222222222222223,
 ('TDP', 'Business'): 0.037037037037037035,
 ('Nifty', 'Business'): 0.3703703703703703,
 ('Sidhu', 'Business'): 0.037037037037037035,
 ('BJP', 'Business'): 0.18518518518518517,
 ('Sensex', 'Business'): 0.48148148148148145,
 ('Sixer', 'Business'): 0.037037037037037035,
 ('Congress', 'Business'): 0.07407407407407407,
 ('Century', 'Business'): 0.07407407407407407,
 ('TDP', 'Sports'): 0.04,
 ('Nifty', 'Sports'): 0.04,
 ('Sidhu', 'Sports'): 0.28,
 ('BJP', 'Sports'): 0.08,
 ('Sensex', 'Sports'): 0.04,
 ('Sixer', 'Sports'): 0.4,
 ('Congress', 'Sports'): 0.08,
 ('Century', 'Sports'): 0.36}
```

```
query_dict = {}
list_query_dict = []
for data in query_data:
    for i, word in enumerate(words) :
        query_dict[word] = data[i]
    list_query_dict.append(query_dict)
    query_dict = {}
```

[31] Python

```
list_query_dict
```

[32] Pyth

```
[{'TDP': 4,
  'Nifty': 0,
  'Sidhu': 2,
  'BJP': 0,
  'Sensex': 1,
  'Sixer': 0,
  'Congress': 6,
  'Century': 0},
 {'TDP': 0,
  'Nifty': 0,
  'Sidhu': 2,
  'BJP': 0,
  'Sensex': 0,
  'Sixer': 9,
  'Congress': 0,
  'Century': 9},
 {'TDP': 5,
  'Nifty': 0,
  'Sidhu': 2,
  'BJP': 5,
  'Sensex': 0,
  'Sixer': 9,
  'Congress': 0,
  'Century': 9}]
```

```
categorical_result_probability = {}
result_probability =[]
for query_dict in list_query_dict:
    for output_class in outputlabels :
        cur_prob = 1
        for word in words :
            cur_prob *= (conditional_probability[(word, output_class)] ** query_dict[word])
            categorical_result_probability[output_class] = cur_prob
    result_probability.append(categorical_result_probability)
    categorical_result_probability = {}
```

[33] Python

```
result_probability
```

[34] Python

```
[{'Politics': 2.0212765225616147e-08,
  'Business': 2.0530257296565003e-16,
  'Sports': 2.1045339750400004e-15},
 {'Politics': 1.3799701971155752e-32,
  'Business': 1.2077990336473961e-26,
  'Sports': 2.0872693292214035e-09},
 {'Politics': 2.1796398918917e-38,
  'Business': 1.8331882202741092e-37,
  'Sports': 7.003713677304523e-22}]
```

```python
i=1
for categorical_result_probability in result_probability:
    result_category = max(categorical_result_probability, key=categorical_result_probability.get)
    result_score = categorical_result_probability[result_category]
    print(f"The query {i} entered belongs to the category : {result_category}")
    i=i+1
```

[35] Python

```
The query 1 entered belongs to the category : Politics
The query 2 entered belongs to the category : Sports
The query 3 entered belongs to the category : Sports
```