<u>Practical-5</u>

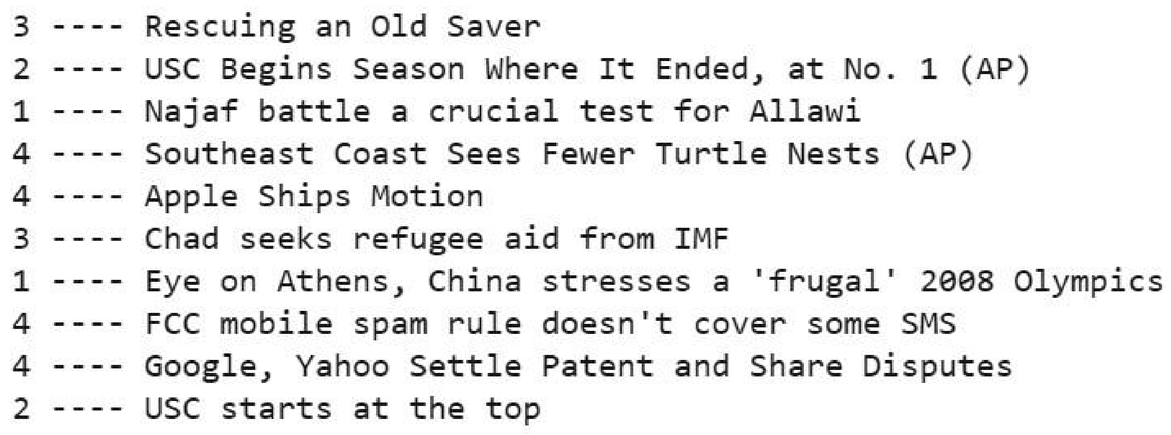
Objective: Implement Text classification using Bayesian rule.

Dataset:

1.Dataset contains information about the news articles classified into four classes i.e., World, Sports, Sci/Tech, Business.

2.It contains 200 articles with each of the category having 50 articles on an average

3. Below are the 10 random sets from this training set:



```
3 ---- Rescuing an Old Saver
2 ---- USC Begins Season Where It Ended, at No. 1 (AP)
1 ---- Najaf battle a crucial test for Allawi
4 ---- Southeast Coast Sees Fewer Turtle Nests (AP)
4 ---- Apple Ships Motion
3 ---- Chad seeks refugee aid from IMF
1 ---- Eye on Athens, China stresses a 'frugal' 2008 Olympics
4 ---- FCC mobile spam rule doesn't cover some SMS
4 ---- Google, Yahoo Settle Patent and Share Disputes
2 ---- USC starts at the top
```

4.Testing query:

[India vs Pakistan ODI match,India wins **;**Google launches new phone with jio]

Procedure:

1.Load the dataset train.csv and convert it into a list containing list of two elements, class name and article heading.

2.Do the pre-processing of the text present in article heading in following manner:

- Remove all the numeric, alpha-numeric and special characters from the text
- Converts the strings into tokens
- Remove all the stop words
- Apply stemming for getting the root word for all the text.

3.Combine all the article text and find unique words from it.

4.Create different space for each given class and count occurrence of each character in each class.

5.Calculate prior probabilities for each word in each space that is created. Formula for prior probability is:

P(wk/classi) = (nk + 1)/(n + |vocabulary|)

Where wk is the current word, nk is the number of times the word appears for class i, n is the total number of words in class i and |vocabulary| is the number of unique words in text corpus.

6.For given test data, multiply all the probabilities of all the words present in all elements of test data. Do this for each class

7.Find the maximum probability and the class that gives it. This is the required prediction

Code:

```
import nltk
nltk.download('all')

from nltk import word_tokenize

from csv import reader
import numpy as np
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import re

tokens = []
class_article = []
stop_words = set(stopwords.words('english'))
ps = PorterStemmer()


with open('/content/train.csv', 'r') as read_obj:
 csv_reader = reader(read_obj)
 for row in csv_reader:
 article_description = row[1]
 class_article.append([row[0],article_description])
 cleaned_text = re.sub('[^A-Za-z0-9]+|[0-9]', " ", article_description)
 token = [ps.stem(w) for w in word_tokenize(cleaned_text)]
 tokens = np.append(tokens, token)
unique_tokens = np.unique(tokens)


filtered_sentence = [w.lower() for w in unique_tokens if not w.lower() in stop_words]


import random
random.shuffle(class_article)
for ca in class_article[:10]:
 print(ca[0], "----", ca[1])

print(class_article[0])
print(filtered_sentence)
print(len(filtered_sentence))


freq_dict_1 = {}
freq_dict_2 = {}
freq_dict_3 = {}
freq_dict_4 = {}


for w in filtered_sentence:
 freq_dict_1[w] = 0 if w not in freq_dict_1 else freq_dict_1[w] + 1
 freq_dict_2[w] = 0 if w not in freq_dict_2 else freq_dict_2[w] + 1
 freq_dict_3[w] = 0 if w not in freq_dict_3 else freq_dict_3[w] + 1
 freq_dict_4[w] = 0 if w not in freq_dict_4 else freq_dict_4[w] + 1

for w in filtered_sentence:
 for item in class_article:
 if item[0] == '1' and w.lower() in item[1].lower():
 freq_dict_1[w] = freq_dict_1[w] + 1
 if item[0] == '2' and w.lower() in item[1].lower():
 freq_dict_2[w] = freq_dict_2[w] + 1
 if item[0] == '3' and w.lower() in item[1].lower():
 freq_dict_3[w] = freq_dict_3[w] + 1
 if item[0] == '4' and w.lower() in item[1].lower():
 freq_dict_4[w] = freq_dict_4[w] + 1


print(freq_dict_1)
print(freq_dict_2)
print(freq_dict_3)
print(freq_dict_4)


n_1 = sum(freq_dict_1.values())
```

```
n_2 = sum(freq_dict_2.values())
n_3 = sum(freq_dict_3.values())
n_4 = sum(freq_dict_4.values())
vocab = len(filtered_sentence)


for w in filtered_sentence:
 freq_dict_1[w] = (freq_dict_1[w] + 1)/(n_1 + vocab)
 freq_dict_2[w] = (freq_dict_2[w] + 1)/(n_2 + vocab)
 freq_dict_3[w] = (freq_dict_3[w] + 1)/(n_3 + vocab)
 freq_dict_4[w] = (freq_dict_4[w] + 1)/(n_4 + vocab)


test_data = []
reply = 'y'
while (reply.lower() == 'y'):
 test_string = input("Input article heading: ")
 test_data.append(test_string)
 reply = input("Wish to enter more? (y/n) ")
print(test_data)


def evaluate(test_data):
 result = []
 for test in test_data:
 probabilities = [1,1,1,1]
 test = test.lower()
 if test_word in freq_dict_1:
 probabilities[0] = probabilities[0] * freq_dict_1[test_word]
 if test_word in freq_dict_2:
 probabilities[1] = probabilities[1] * freq_dict_2[test_word]
 if test_word in freq_dict_3:
 probabilities[2] = probabilities[2] * freq_dict_3[test_word]
 if test_word in freq_dict_4:
 probabilities[3] = probabilities[3] * freq_dict_4[test_word]
 max_prob = max(probabilities)
 result.append([test, probabilities.index(max_prob)+1])
 return result


def predict(values):
 for value in values:
 if value[1] == 1:
 print(value[0], " belongs to class World")
 elif value[1] == 2:
 print(value[0], " belongs to class Sports")
 elif value[1] == 3:
 print(value[0], " belongs to class Business")
```

```
        elif value[1] == 4:
        print(value[0], " belongs to class Sci/Tech")


        predict(evaluate(test_data))
```

Output:

```
India vs Pakistan odi match, India wins belongs to class Sports
google launches new phone with jio belongs to class Sci/Tech
```