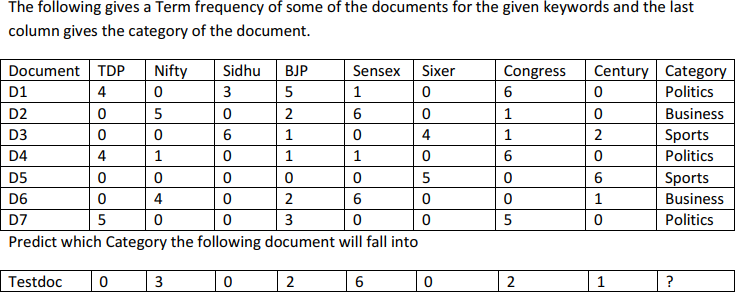
**CSE3024 WEB MINING**

**NAME:** Aryaman Mishra

**REGISTRATION NUMBER:** 19BCE1027

**LAB EXERCISE 5**

**Problem-1:** Write a Naïve Bayes Classifier in python without using any package for the following dataset.



**Proposed Algorithm/Pseudocode:**

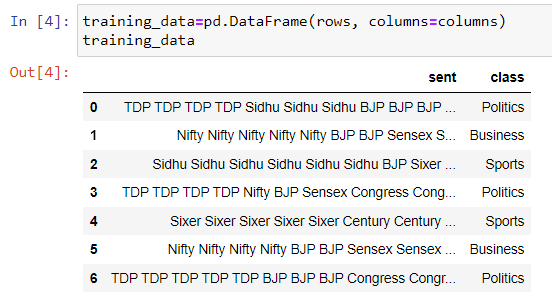
1. Compute the term document matrix for each category.
2. Find frequency of word in each category.
3. Calculate the probability of each word in every category.
4. Calculate the total count of each feature in the training set.
5. Find the probability of all the words in a text document.
6. Lowest priority is given to the document.

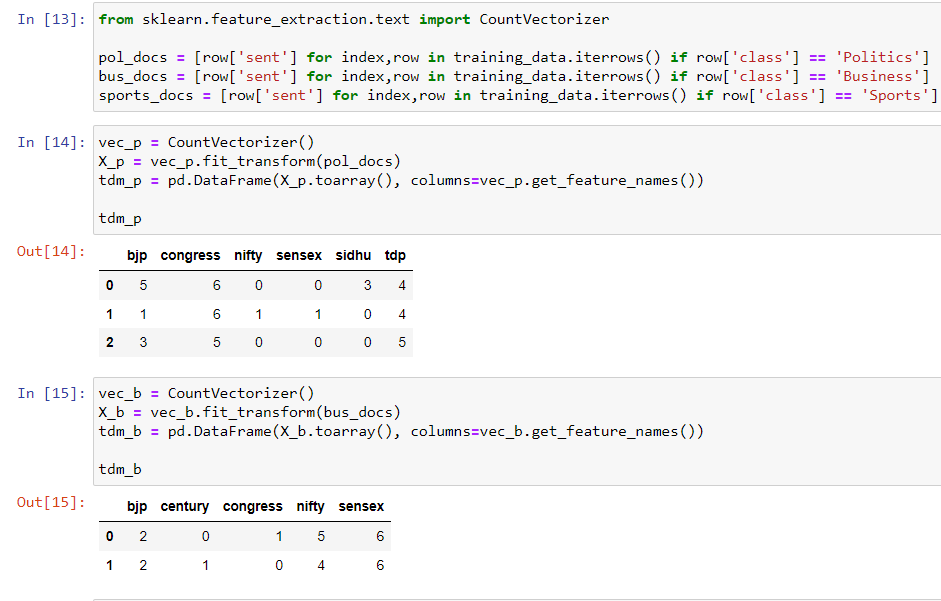
**Data Structure Proposed:** 2D-Arrays, Dictionaries.

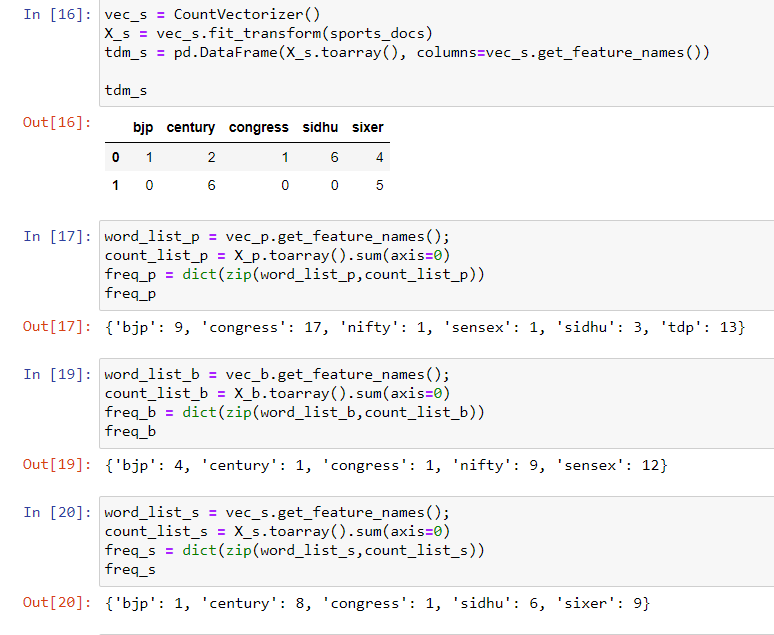
**Libraries Used:** Pandas, CountVectorizer

**IMPLEMENTATION CODE AND RESULTS:**

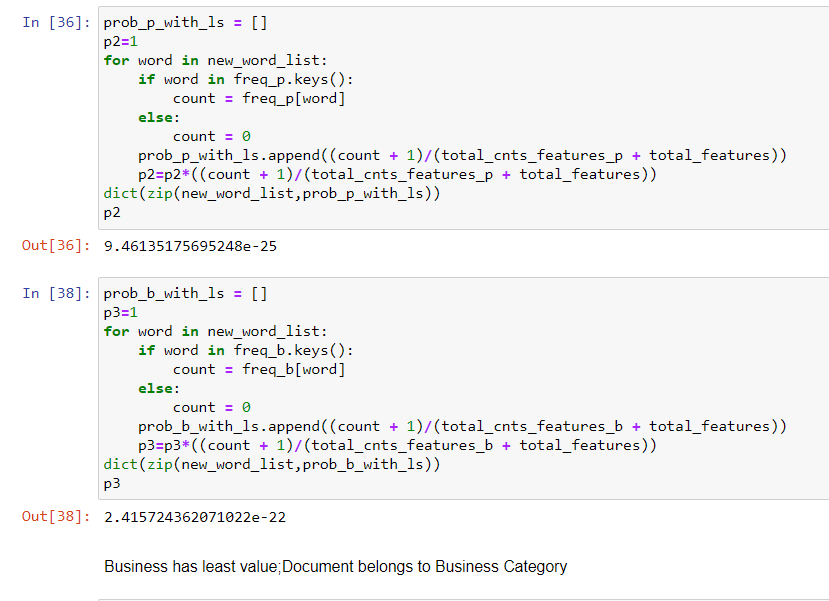












**RESULTS:**Test Doc belongs to ‘Business’ Category.

**Challenging Exercise 1-** Take any large text corpora, do the necessary preprocessing and built a Naïve Bayes Classifier and draw the interferences.

**Proposed Algorithm:**

1. First, we will**load all the necessary libraries** required for our classifier.
2. Next, **we load the data**(**training** and **test** sets):
3. We calculate the**total number of classes** and **samples** we have.
4. The next step consists of building the **Naive Bayes classifier** and finally **training**the**model. W**e will convert the collection of text documents (train and test sets) into a matrix of token counts.
5. To implement that text transformation we will use the **make\_pipeline**function. This will internally transform the text data and then the model will be fitted **using the transformed data.**
6. The last line of code **predicts the labels of the test set.**
7. Finally, we build the [**multi-class confusion matrix**](https://towardsdatascience.com/roc-curve-explained-using-a-covid-19-hypothetical-example-binary-multi-class-classification-bab188ea869c?source=friends_link&sk=08f3dba9c6415860f84f5195d9b0ff65)to see if the model is good or if the model predicts correctly only specific text categories.
8. Hence, we classify whatever sentence we choose.

**Pseudocode:**

from sklearn.feature\_extraction.text import CountVectorizer

corpus = [

'This is the first document.',

'This document is the second document.',

'And this is the third one.',

'Is this the first document?',

]

vectorizer = CountVectorizer()

X = vectorizer.fit\_transform(corpus)

print(vectorizer.get\_feature\_names())

['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']

print(X.toarray())

[[0 1 1 1 0 0 1 0 1]

[0 2 0 1 0 1 1 0 1]

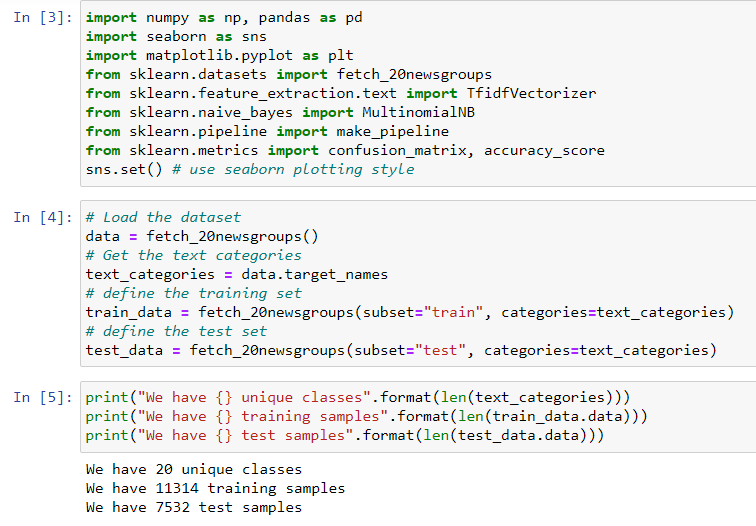
[1 0 0 1 1 0 1 1 1]

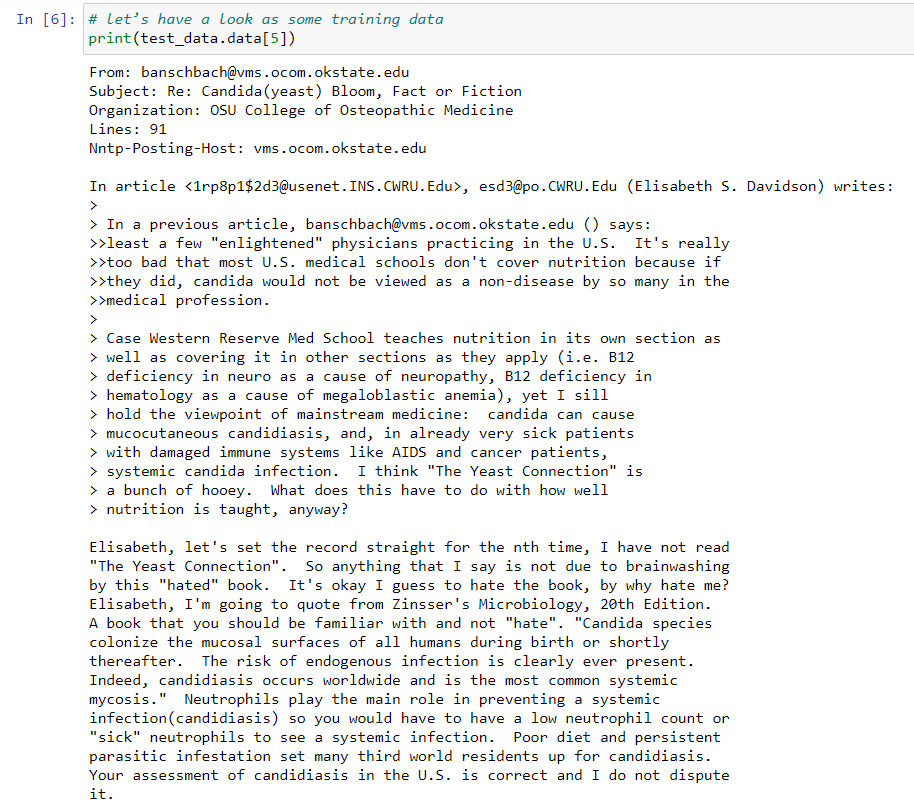
[0 1 1 1 0 0 1 0 1]]

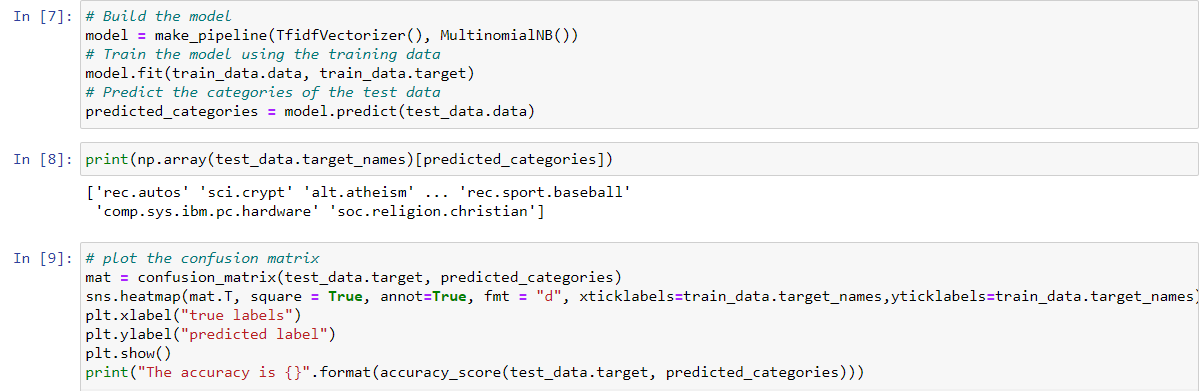
**Data Structure Proposed: Arrays**

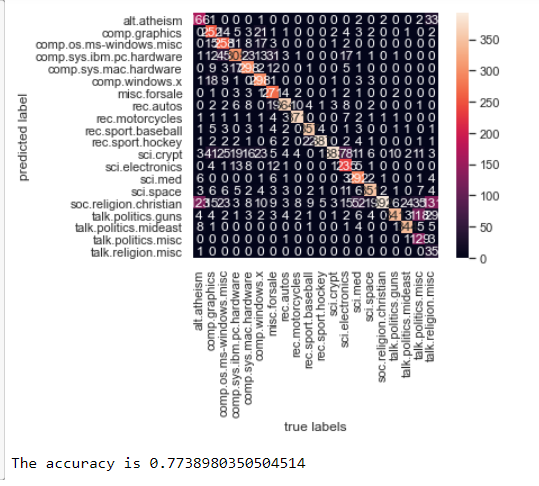
**Libraries Used:** Pandas, Seaborn, Matplotlib, pyplot, sklearn.

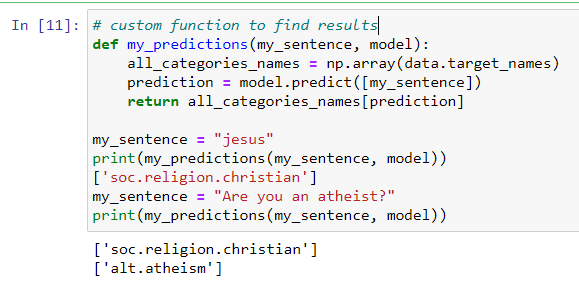
**IMPLEMENTATION CODE AND RESULTS:**











**RESULTS:** Hence we are able to infer our results and find out the confusion matrix and the value of accuracy which is around 0.77 or 77%.