

Assumptions :

- While removing the punctuations the letters are separated. For example u.s.a. will become u s a

Naïve Bayes

a).Pre-Processing :

- Tokenization : Longer strings are splits into tokens. In this the text is tokenized into sentences and then sentences are tokenized into words. During tokenization the punctuations are also removed.
- After tokenizing the documents the number is converted to words(like 100 is converted to one hundred).If size is greater than each digit of number is converted to word.(like 100 is converted to one zero zero)
- Case-Folding(Lower Case) : Converting all words to lower case to make searching easy.
- Stemming : It is process of eliminating affixes from the word to get the stem word. Generally it makes word running to run.

b).Methodology :

- First step is to load the files from directory and reading the files using csopen() function.
- Then we split the train and test data into x:y ratio where x is training data and y is test data.
- Once the file is read the all pre-processing is done like tokenization, conversion of number to words and stemming etc in train data.
- For each term in document in train data we calculate term frequency to make Vocabulary that is number of unique terms.
- Now we create document vector for each document in train data using Vocabulary and it stores the term along with its count in that document.
- Now we create class vector using document vector that is we have already known number of classes and number of document for each class.
- After class vector is created, now document test data is tested using Naïve bayes algorithm. The formula is stated as below where $P(c/x)$ is for given document x we have to find in which class it will belong. The probability is calculated for each word in document and multiplied. We calculate $P(c/x)$ of all the classes and whose probability is maximum the document will be assigned to that class.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

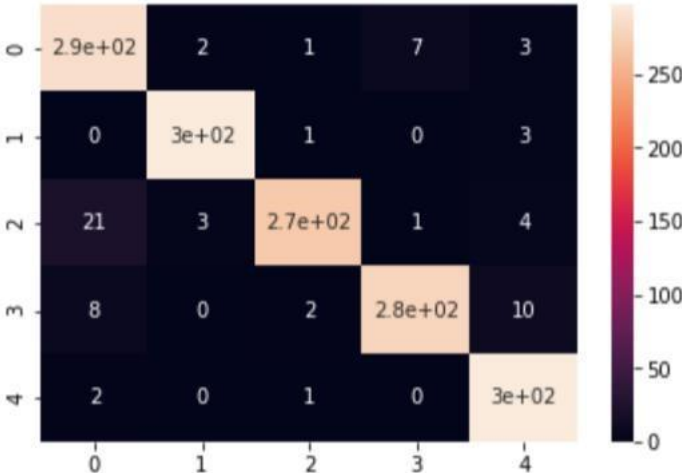
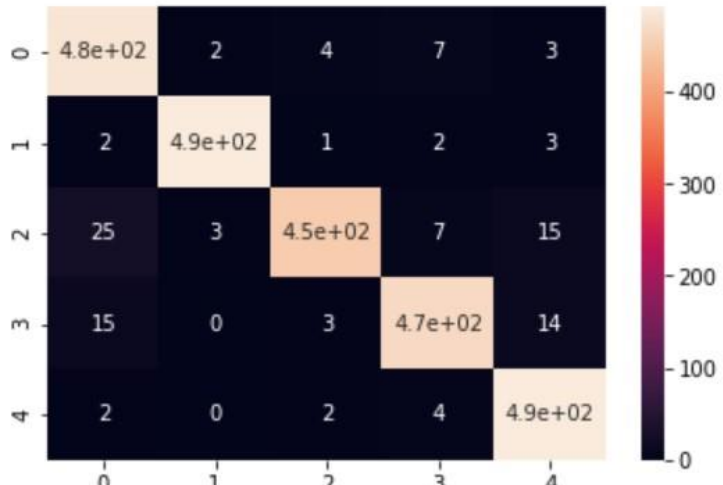
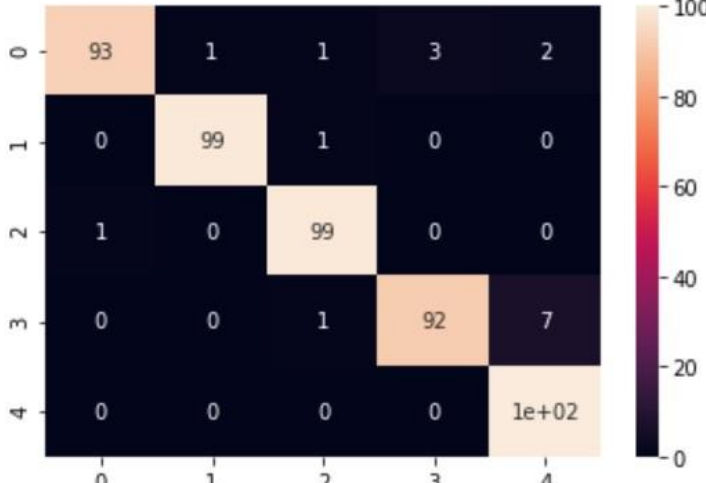
The diagram shows the formula with arrows pointing to its components: $P(c|x)$ is labeled 'Posterior Probability', $P(x|c)$ is labeled 'Likelihood', $P(c)$ is labeled 'Class Prior Probability', and $P(x)$ is labeled 'Predictor Prior Probability'.

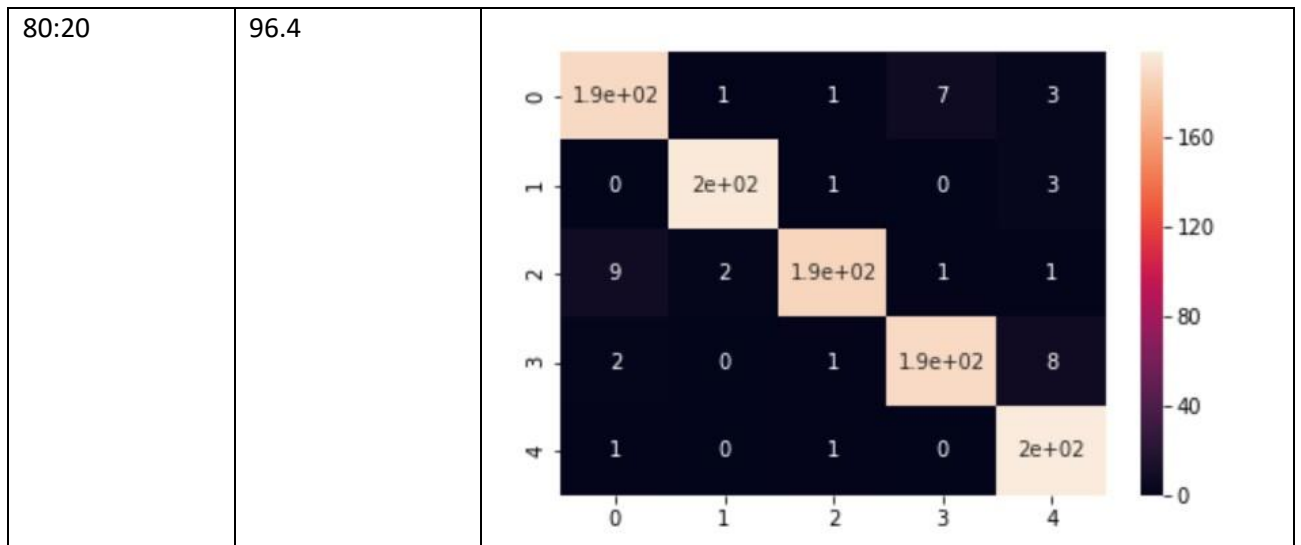
$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

- When all the documents are tested and now we have Actual and Predicted Class labels. Now we will measure accuracy and confusion matrix.

Accuracy= Correctly Labeled Documents/Total Documents **c)**

Test Cases :

Ratio – X : Y	Accuracy	Confusion Matrix																																				
70:30	95.4	 <table><tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th></tr><tr><th>0</th><td>2.9e+02</td><td>2</td><td>1</td><td>7</td><td>3</td></tr><tr><th>1</th><td>0</td><td>3e+02</td><td>1</td><td>0</td><td>3</td></tr><tr><th>2</th><td>21</td><td>3</td><td>2.7e+02</td><td>1</td><td>4</td></tr><tr><th>3</th><td>8</td><td>0</td><td>2</td><td>2.8e+02</td><td>10</td></tr><tr><th>4</th><td>2</td><td>0</td><td>1</td><td>0</td><td>3e+02</td></tr></table>		0	1	2	3	4	0	2.9e+02	2	1	7	3	1	0	3e+02	1	0	3	2	21	3	2.7e+02	1	4	3	8	0	2	2.8e+02	10	4	2	0	1	0	3e+02
	0	1	2	3	4																																	
0	2.9e+02	2	1	7	3																																	
1	0	3e+02	1	0	3																																	
2	21	3	2.7e+02	1	4																																	
3	8	0	2	2.8e+02	10																																	
4	2	0	1	0	3e+02																																	
50:50	95.44	 <table><tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th></tr><tr><th>0</th><td>4.8e+02</td><td>2</td><td>4</td><td>7</td><td>3</td></tr><tr><th>1</th><td>2</td><td>4.9e+02</td><td>1</td><td>2</td><td>3</td></tr><tr><th>2</th><td>25</td><td>3</td><td>4.5e+02</td><td>7</td><td>15</td></tr><tr><th>3</th><td>15</td><td>0</td><td>3</td><td>4.7e+02</td><td>14</td></tr><tr><th>4</th><td>2</td><td>0</td><td>2</td><td>4</td><td>4.9e+02</td></tr></table>		0	1	2	3	4	0	4.8e+02	2	4	7	3	1	2	4.9e+02	1	2	3	2	25	3	4.5e+02	7	15	3	15	0	3	4.7e+02	14	4	2	0	2	4	4.9e+02
	0	1	2	3	4																																	
0	4.8e+02	2	4	7	3																																	
1	2	4.9e+02	1	2	3																																	
2	25	3	4.5e+02	7	15																																	
3	15	0	3	4.7e+02	14																																	
4	2	0	2	4	4.9e+02																																	
90:10	96.8	 <table><tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th></tr><tr><th>0</th><td>93</td><td>1</td><td>1</td><td>3</td><td>2</td></tr><tr><th>1</th><td>0</td><td>99</td><td>1</td><td>0</td><td>0</td></tr><tr><th>2</th><td>1</td><td>0</td><td>99</td><td>0</td><td>0</td></tr><tr><th>3</th><td>0</td><td>0</td><td>1</td><td>92</td><td>7</td></tr><tr><th>4</th><td>0</td><td>0</td><td>0</td><td>0</td><td>1e+02</td></tr></table>		0	1	2	3	4	0	93	1	1	3	2	1	0	99	1	0	0	2	1	0	99	0	0	3	0	0	1	92	7	4	0	0	0	0	1e+02
	0	1	2	3	4																																	
0	93	1	1	3	2																																	
1	0	99	1	0	0																																	
2	1	0	99	0	0																																	
3	0	0	1	92	7																																	
4	0	0	0	0	1e+02																																	



Analysis :

From above observations, we can infer that there is slight change in the accuracy along all the splits. This is because all the classes in the given dataset are independent of each other and distribution of training-data and testing data is same across each class so if our training data is less then it will perform well on test data.

Naïve Bayes using TF IDF

a). **Pre-Processing** : Same as in question1

b). **Methodology** :

- First step is to load the files from directory and reading the files using `copen()` function.
- Then we split the train and test data into 70:30 ratio where 70% is training data and 30% is test data.
- Once the file is read the all pre-processing is done like tokenization, conversion of number to words and stemming etc in train data.
- For each term in document in train data we calculate term frequency to make Vocabulary that is number of unique terms.
- Now for each term in Vocabulary we will calculate the IDF Inverse Document Frequency of the term.
- Now we have to perform classification on the basis of efficient features selection. Here the features are Vocabulary that is words so we will select top 60% of the words from the Vocabulary on the basis of TF-IDF Score of the word and call it as Vocab and perform the steps below.
- Now we create document vector for each document in train data using Vocab and it stores the term along with its TF-IDF scores.
- Now we create class vector using document vector that is we have already known number of classes and number of document for each class.
- After class vector is created, now document test data is tested using Naïve bayes algorithm. The formula is stated as below where $P(c/x)$ is for given document x we have to find in which class it will belong. The probability is calculated for each word in document and multiplied.

We calculate $P(c/x)$ of all the classes and whose probability is maximum the document will be assigned to that class.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Likelihood
Class Prior Probability
Posterior Probability
Predictor Prior Probability

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

- When all the documents are tested and now we have Actual and Predicted Class labels. Now we will measure accuracy and confusion matrix.

Accuracy= Correctly Labeled Documents/Total Documents **c).Test-Cases**

:

X:Y(training and testing)	Accuracy	Confusion Matrix																																				
70:30	96.2	<table><tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th></tr><tr><th>0</th><td>2.9e+02</td><td>2</td><td>1</td><td>7</td><td>3</td></tr><tr><th>1</th><td>0</td><td>3e+02</td><td>1</td><td>0</td><td>3</td></tr><tr><th>2</th><td>19</td><td>3</td><td>2.7e+02</td><td>1</td><td>3</td></tr><tr><th>3</th><td>8</td><td>0</td><td>2</td><td>2.8e+02</td><td>10</td></tr><tr><th>4</th><td>2</td><td>0</td><td>1</td><td>0</td><td>3e+02</td></tr></table>		0	1	2	3	4	0	2.9e+02	2	1	7	3	1	0	3e+02	1	0	3	2	19	3	2.7e+02	1	3	3	8	0	2	2.8e+02	10	4	2	0	1	0	3e+02
	0	1	2	3	4																																	
0	2.9e+02	2	1	7	3																																	
1	0	3e+02	1	0	3																																	
2	19	3	2.7e+02	1	3																																	
3	8	0	2	2.8e+02	10																																	
4	2	0	1	0	3e+02																																	

Analysis of 70:30 when we apply Normal TF and feature selection using TF-IDF Scores :

From above observation we can infer that when we apply feature selection the accuracy is improved and since feature selection will take unnecessary noise into consideration so overall computation is also improved. Since we are using TF-IDF measure as our feature selection component so it will consider only those word which are important in entire corpus while Normal TF will give information about words which are important to document.