Machine Learning Project -2

Question 1:

Reasons for difficulties in estimating the parameters by using the given model:

Given that X*i* is sampled from some distribution that depends on its position X*i* but in English it is possible to re-arrange the words in a sentence without changing the meaning. Therefore from this it is evident that the distribution values doesn't depend on the position of the word.

And given there are 1000 documents, each 1000 word long. If we estimate the parameters for 50,000 vocabulary set using limited number of document set we might end up with wrong values. Further using these estimated parameters for classification of documents results is classification errors. Therefore we need to estimate parameters on a larger dataset.

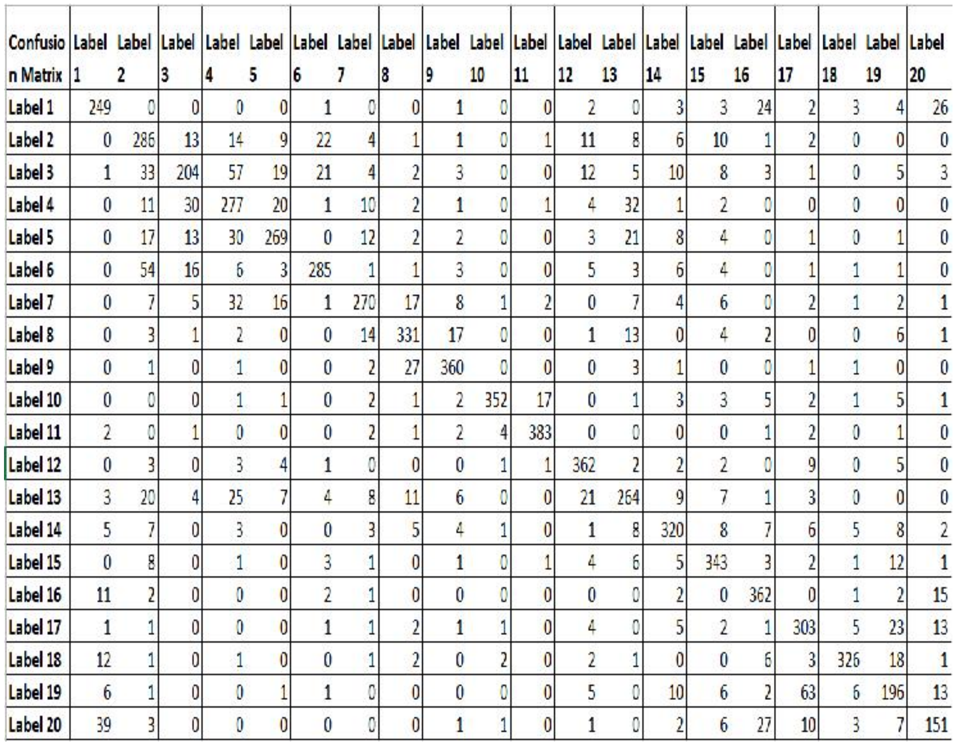
Question 2:

For Beta= (1/vocabulary size), the total number of correctly predicted documents are **5893** out of **7505** documents.

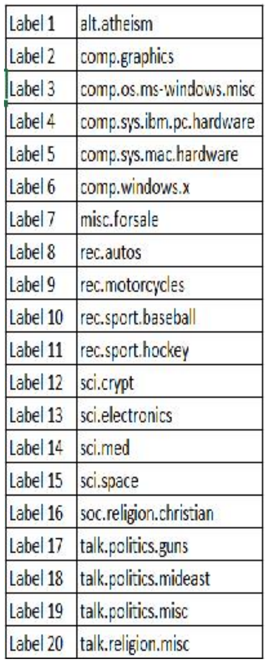
I have obtained an accuracy of **78.52098%.**

Confusion Matrix:

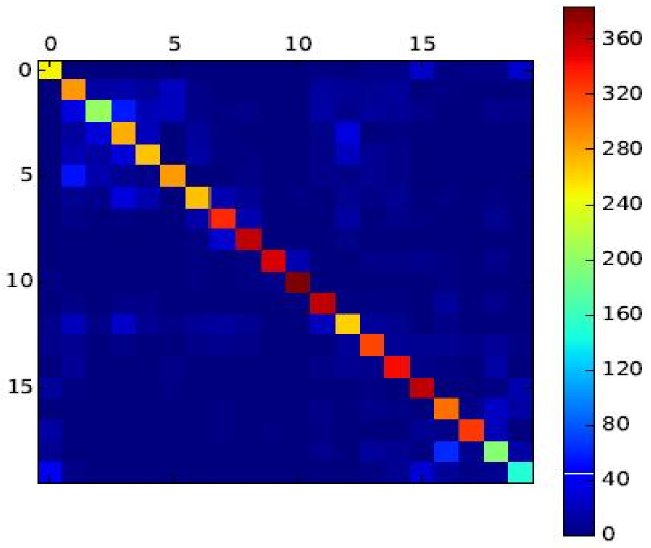
Given Output Vs Predicted values



Confusion Matrix	Label 1	Label 2	Label 3	Label 4	Label 5	Label 6	Label 7	Label 8	Label 9	Label 10	Label 11	Label 12	Label 13	Label 14	Label 15	Label 16	Label 17	Label 18	Label 19	Label 20
Label 1	249	0	0	0	0	1	0	0	1	0	0	2	0	3	3	24	2	3	4	26
Label 2	0	286	13	14	9	22	4	1	1	0	1	11	8	6	10	1	2	0	0	0
Label 3	1	33	204	57	19	21	4	2	3	0	0	12	5	10	8	3	1	0	5	3
Label 4	0	11	30	277	20	1	10	2	1	0	1	4	32	1	2	0	0	0	0	0
Label 5	0	17	13	30	269	0	12	2	2	0	0	3	21	8	4	0	1	0	1	0
Label 6	0	54	16	6	3	285	1	1	3	0	0	5	3	6	4	0	1	1	1	0
Label 7	0	7	5	32	16	1	270	17	8	1	2	0	7	4	6	0	2	1	2	1
Label 8	0	3	1	2	0	0	14	331	17	0	0	1	13	0	4	2	0	0	6	1
Label 9	0	1	0	1	0	0	2	27	360	0	0	0	3	1	0	0	1	1	0	0
Label 10	0	0	0	1	1	0	2	1	2	352	17	0	1	3	3	5	2	1	5	1
Label 11	2	0	1	0	0	0	2	1	2	4	383	0	0	0	0	1	2	0	1	0
Label 12	0	3	0	3	4	1	0	0	0	1	1	362	2	2	2	0	9	0	5	0
Label 13	3	20	4	25	7	4	8	11	6	0	0	21	264	9	7	1	3	0	0	0
Label 14	5	7	0	3	0	0	3	5	4	1	0	1	8	320	8	7	6	5	8	2
Label 15	0	8	0	1	0	3	1	0	1	0	1	4	6	5	343	3	2	1	12	1
Label 16	11	2	0	0	0	2	1	0	0	0	0	0	0	2	0	362	0	1	2	15
Label 17	1	1	0	0	0	1	1	2	1	1	0	4	0	5	2	1	303	5	23	13
Label 18	12	1	0	1	0	0	1	2	0	2	0	2	1	0	0	6	3	326	18	1
Label 19	6	1	0	0	1	1	0	0	0	0	0	5	0	10	6	2	63	6	196	13
Label 20	39	3	0	0	0	0	0	0	1	1	0	1	0	2	6	27	10	3	7	151

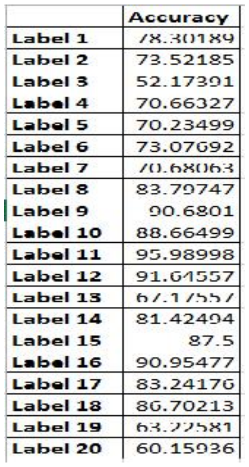


Label 1	alt.atheism
Label 2	comp.graphics
Label 3	comp.os.ms-windows.misc
Label 4	comp.sys.ibm.pc.hardware
Label 5	comp.sys.mac.hardware
Label 6	comp.windows.x
Label 7	misc.forsale
Label 8	rec.autos
Label 9	rec.motorcycles
Label 10	rec.sport.baseball
Label 11	rec.sport.hockey
Label 12	sci.crypt
Label 13	sci.electronics
Label 14	sci.med
Label 15	sci.space
Label 16	soc.religion.christian
Label 17	talk.politics.guns
Label 18	talk.politics.mideast
Label 19	talk.politics.misc
Label 20	talk.religion.misc



Question 3:

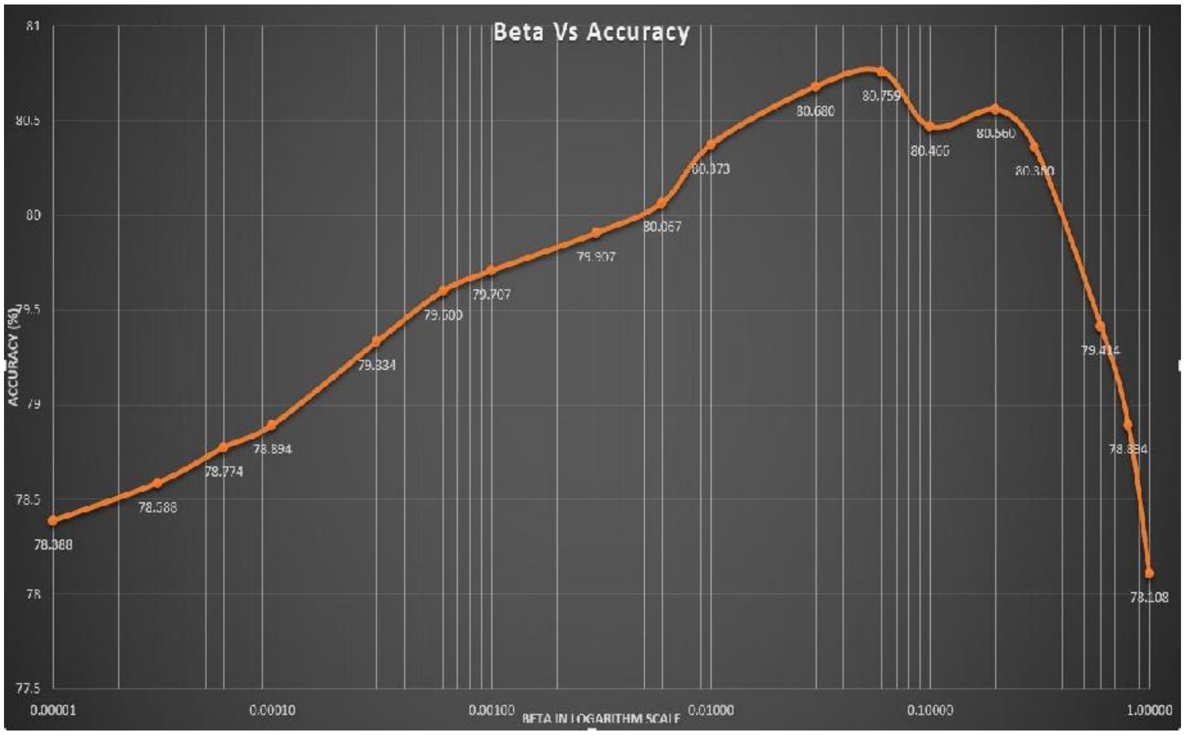
I have calculated the accuracy for every label as shown in the below table. And from the confusion matrix we can say that news groups related to similar topics are more confused than others. From the below table we can see that Label 3 has got a minimum accuracy. Label 3(comp.os.ms-windows.misc) is more confused with label 2,4,5 and 6 (comp.graphics, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x) as they all similar topics. The second least in below given table is Label 20 (talk.religion.misc) is more confused with Label 16 (soc.religion.christian) as they belong to similar topics.



	Accuracy
Label 1	78.30189
Label 2	73.52185
Label 3	52.17391
Label 4	70.66327
Label 5	70.23499
Label 6	73.07692
Label 7	70.68063
Label 8	83.79747
Label 9	90.6801
Label 10	88.66499
Label 11	95.98998
Label 12	91.64557
Label 13	67.17557
Label 14	81.42494
Label 15	87.5
Label 16	90.95477
Label 17	83.24176
Label 18	86.70213
Label 19	63.22581
Label 20	60.15936

Question 4:

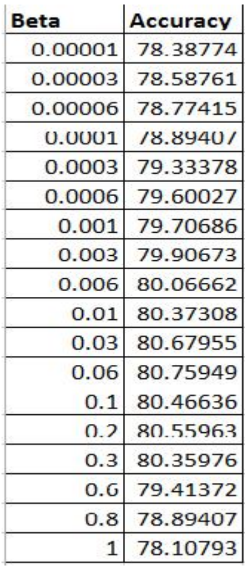
I have taken the Beta values in the range of (0.00001 - 1.0) ran the code and collected the Accuracy values. I have taken the values into Excel and plotted the semilog graph.



Seen from graph, as the Beta values increases the Accuracy also increases to certain level and tends to decrease. Highest Accuracy of **80.759%** is obtained for Beta= **0.06.**

As the Beta value decreases the probabilities of the new words which are not found during the training documents and the rare words becomes very less and they tend to decrease the total Posterior value for a Document leading to the misclassification of the document.

Similarly as the Beta value tends to 1 the probabilities of the rare words do not contribute much to Posterior calculation get neglected due to which misclassification of documents occur.



Beta	Accuracy
0.00001	78.38774
0.00003	78.58761
0.00006	78.77415
0.0001	78.89407
0.0003	79.33378
0.0006	79.60027
0.001	79.70686
0.003	79.90673
0.006	80.06662
0.01	80.37308
0.03	80.67955
0.06	80.75949
0.1	80.46636
0.2	80.55963
0.3	80.35976
0.6	79.41372
0.8	78.89407
1	78.10793

Question 5:

To rank the words in the dataset based on how much the classifier 'relies on' them when performing its classification Entropy or Mutual Information can be better used.

But as we are limited to use Priors and Likelihood matrix, I have used the **Likelihood Matrix** to find the significant words.

Method Proposed: Common words or stop words have higher P(X/Y) value in all the labels (group of documents) because they are found in every document. Topic Specific words or rare words have higher P(X/Y) value only in 1 or 2 labels and have a small value across all other labels.

So I found the difference between the Max P(X/Y)) value for a row & every P(X/Y) in that row and summed up all the 20 modified labels values for a Word ID and used it as an estimate to tell whether the word is significant for classification or not.

For the stop words or common words the sum of all 20 modified P(X/Y) values will be less when compared to the sum value of topic specific words. So I have taken the list of 100 Word ID's with highest sum value and found the corresponding words from the vocabulary list using the word id.

Process Followed:

Step 1:

I have taken the Likelihood Matrix P(X/Y) matrix of size [(vocabulary_size, no_of_labels)=(61188,20)] and found out the maximum P(X/Y) value for a Word ID across all the labels.

Max(P(X/Y)) value for Word ID among 20 values in a row.

Step 2:

I have taken the difference of Word ID's P(X/Y) value and the Maximum P(X/Y) value of that row at every position (i,j) in the Likelihood matrix

Likelihood Matrix [i , j]=Row's Max(P(X/Y))-P(X/Y) at every (i,j) position in a row in Likelihood Matrix.

Step 3:

I have taken the sum of all values across a row and obtained of resultant matrix of size (vocabulary_size, 1)=(61188,1)] and sorted the resultant matrix to find the Word ID's which have much importance

Sort (Sum (Likelihood Matrix [i,:]))

And found the 100 Word ID's with highest sum value and found the corresponding words in the vocabulary.

Improvements:

As the P(X/Y) values for all the Word ID's across a label is not normalized, there is a chance of having high sum values for some of the stop words. So we can normalize them to avoid getting the stop words from the method**.**

Question 6:

The list of top 100 words found using the above process are:

Nhl, apple, armenian, armenians, baseball, bike, bus, car, card, cars, chip, clipper, com, condition, controller, db, disk, dos, drive, drives, encryption, entry, files, or, god, government, graphics, game, gun, guns, he, his, hockey, image, israel, israeli, jews, jpeg, key, mac, mb, motif, my, nasa, of, offer, output, people, play, price, program, sale, scsi, season, server, shipping, space, system, team, that, the, they, turkish, was, we, were, who, widget, window, windows, year, and, president, and, are, article, as, asking, be, by, dod, edu, file, ide, in, is, it, jesus, keys, last, mail, me, mr, new, not, on, or, please, their, to, with, you

Question 7:

Training the machine learning algorithms based on biased data affects the future prediction. From the above list of words I could find the words Turkish, Armenian and Isreal and they could be mostly appearing in the age old news articles and may not appear in the future news articles. Therefore, the classification based on this ML algorithm developed on age old data may lead to some misclassification.

High Level Description of how the code works:

I have implemented the Naïve Bayes Classifier in Python.

The brief description of how the code flows:

- As we run the file the program enters the **Main function**
- In **Main Function** I have called various custom built functions to calculate Prior Matrix, Likelihood Matrix, Posterior Matrix, Accuracy and to find the top 100 words list.
- Firstly the **calc_prior()** function is called

  The Input train_label file is read and the count of number of documents per each label is extracted and is divided by total number of documents to get the Prior value for each label

  P(Yk) = # of docs labeled Yk/total # of docs

  The program returns to the Main function after calculating the Prior values P(Y) and prints the Prior Matrix generated by the **calc_prior()** function

- Next **doc_indexes(label_doc_count)** is called and label_doc_count matrix is passed as input to the function. This matrix contains the count of no.of.documents per label. This function is used to calculate the document id range in every label. After executing this function control returns to Main function
- Next **calc_likelihood()** function is called to calculate the Likelihood Matrix.

  In this function **train_data** file is read as input and Likelihood values are calculated using the below formula

P(Xi|Yk)=(count of Xi in Yk)+(α−1) / (total words in Yk)+((α−1)∗(length of vocab list)))

where α=1+β, β=1|V|

Count of X*i* is calculated label Y*k* and Total words in Y*k* is calculated by extracting the data for Y*k* label from the total data

- After calculating Likelihood matrix, the control returns to the Main function and prints the Likelihood Matrix and calls the **calc_posterior()** function.

  In this function **test_data** file is read and the posterior values are calculated using the below formula:

  Ynew=argmax[log2(P(Yk))+∑i(# of Xnewi)log2(P(Xi|Yk))]

- After calculating the Posterior Matrix, the control returns to the Main function and prints the Posterior Matrix and predicted label values, calls the **calc_accuracy()** function
- In this function Accuracy is calculated, Confusion Matrix are calculated and the control returns to the Main function
- In the Main function Accuracy value and Confusion Matrix are printed and the **calc_100words():** function is called.

  In this function the proposed method (Question 5 solution) is implemented to get the top 100 significant words on which the classifier relies on.

- After finding the top words the control returns to Main function and prints the list of 100 words and the program ends.