

CS 529 CLASS PROJECT 2: TOPIC CATEGORIZATION

Md Amanul Hasan¹
Farhan Asif Chowdhury²
September 19, 2018

Abstract

In this class project, we have implemented both Naive Bayes (NB) and Logistic Regression (LR) algorithms to classify the topic of a document based on its content (words). A total of 12,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups were used for training these algorithms, whereas a total of 6,774 documents were used to test the efficiencies of the two methods. The accuracies of our developed algorithms were evaluated by uploading the outputs for the testing dataset in Kaggle. The accuracies of the NB and the LR are 89.74% and 81.72% respectively without any preprocessing. We have also found that the NB is faster than the LR. In addition to using the whole dataset, we calculated the mutual information between topic (class) and each word (feature) to identify the important features in the large dataset by applying the information theory. As expected, the prepositions (“the”, “of”, “a”) score lower among the words in the documents.

Keywords: *Machine Learning, Topic Categorization, Naive Bayes, Logistic Regression, Mutual Information.*

¹Graduate student, Department of Civil Engineering, University of New Mexico, Albuquerque, email: amanulhasan@unm.edu

²Graduate student, Department of Computer Science, University of New Mexico, Albuquerque, email: fasifchowdhury@unm.edu

Both authors have contributed equally.

1 INTRODUCTION

Topic categorization is the task of identifying the topic of a document based on its content (words) [1]. The primary goal of this project is to implement two supervised machine learning methods: Naive Bayes (NB), and Logistic Regression (LR), to develop respective algorithms those can successfully categorize the documents into different topics based on their words. We are given a total of 12,000 training dataset that consists of (nearly) evenly distributed 20 known newsgroups and a total of 6,774 testing dataset of known topics. We are asked to implement the aforementioned algorithms to learn how to categorize the documents using this training dataset, and then, to employed the trained algorithms to identify the topic of each documents of the testing dataset. Finally, the efficiency of our developed algorithms are to be evaluated by uploading the outputs in Kaggle for different specified conditions. High-level descriptions of our developed algorithms, accuracies of results for the all specified conditions, and explanations of the results are given in following sections.

2 SHORT DESCRIPTIONS OF METHODS

2.1 Naive Bayes(NB)

The NB classifier is the simple "probabilistic classifier" based on applying Bayes' theorem with strong (naive) independence assumptions between the features [2]. It reduces the intractable sample complexity by making the assumption that the features are conditionally independence for a given class. If X represents the features and Y represents the classes of a dataset, the goal of the classifier is to find the $P(Y/X)$ from the Bayes theorem using Eq. (1).

$$P(Y|X_i) = P(Y) * \prod P(X_i|Y) \quad (1)$$

Where, $P(Y/X_i)$ is “posterior”, $P(X/Y)$ is the “Likelihood”) and $P(Y)$ is the “Prior”. The NB is also known as a generative classifier since it indirectly computes the $P(Y/X)$ from $P(X/Y)$ and the $P(Y)$. The Maximum Likelihood Estimation (MLE) is needed to estimate the $P(Y)$ using Eq. (2). On the other hand, Maximum A Posterior (MAP is required for calculating the $P(X/Y)$ using Eq. (3)

$$P(Y_k) = \frac{\text{no of docs labeled } Y_k}{\text{total of docs}} \quad (2)$$

$$P(X_i|Y_k) = \frac{\text{no of } X_i \text{ in } Y_k + (\alpha - 1)}{\text{total words in } Y_k + ((\alpha - 1) * \text{length of vocabulary list}, V)} \quad (3)$$

Where, $\alpha = 1 + \beta$ and $\beta = 1/|V|$. After training the algorithm, the class of a new document, Y^{new} can be classify using Eq. (4).

$$Y^{new} = \operatorname{argmax}[\log_2(P(Y)) + \sum_i (\text{no of } X_i^{new}) \log_2(P(X_i|Y_k))] \quad (4)$$

2.2 Logistic Regression (LR)

The LR is a statistical method that uses the logit function to relate the binary response variable to a set of explanatory variables, which can be discrete and/or continuous [2]. It is a discriminative classifier because it directly calculates the $P(Y/X)$ from the training data. The LR maximizes the conditional data likelihood using a weight variable, w as follows:

$$\begin{aligned} P(D_Y|D_x, w) &= \sum_{j=1}^N \ln(P(Y^j|X^j, w)) \\ &= \sum_j y^j (w_0 + \sum_i^n w_i x_i^j) - \ln(1 + \exp(w_0 + \sum_i^n w_i x_i^j)) \end{aligned} \quad (5)$$

There is no close form solution for the Eq. (5). However, it is a concave function and can be optimized using the gradient ascent/descent algorithm. The LR also experience the problem with overfitting training data. To prevent overfitting, regularization approach is used by penalize the weight term. The form of the W matrix is shown in Eq. (6).

$$W^{t+1} = W^t + \eta((\Delta - P(Y|W, X))X - \lambda W^t) \quad (6)$$

Where, Δ is a matrix of 1 (when $Y^l = y_j$) or 0 (when $Y^l \neq y_j$), η is the learning rate, and λ is the penalty term.

3 HIGH LEVEL DESCRIPTION OF THE CODES

3.1 Data Read and Handling

Handling the data is one of the major challenges of this project because both the training and the testing datasets are significantly large. The training file is about 1.40GB with a total of 12,000

sample data (rows), where each sample has a total of 611,190 elements (columns) of different information including a document ID (first element) and a class information (last element). The elements 2 to 61189 are word counts for a vectorized representation of words (refer to the vocabulary for a mapping). The testing file is about 0.80GB with a total of 6,774 sample data, where each sample has a total of 611,189 elements, similar to the testing data except the class information. Since these data are huge to read at a single time, a loop was used to upload the data by chunk-wise. Each chunk contains are maximum of 1,000 sample data (rows). After that, chunk data was stored in a total data matrix variable by appending them after the previous data. This was done to make the program faster, since we needed to make multiple iteration for the LR. The total However, we again used chunk by chunk data to calculate the probability or other matrices. In this case, a maximum of 100 data from the stored dataset matrix was extracted.

3.2 Naive Bayes (NB) Code

Fig. 1 shows the flow chart for the NB code used for this project. Since total data cannot be imported at a single time, our NB code imports the data by chunk and then develops the required frequency tables by class. Several iterations are required to import the total data, and after each iteration the code updates the frequency tables by adding new data. When all data are imported, it calculates the $P(Y)$ using MLE and $P(X/Y)$ using MAP using these frequency tables. Later, $P(Y)$ and $P(X/Y)$ matrices are used to calculate the posterior $P(Y/X)$. Finally, the classes of the new data can be predicted by choosing the maximum probability of result from the multiplication product of feature matrix of new data with the developed $P(Y/X)$ matrix.

3.2 Logistic Regression (LR) Code

Fig. 2 shows the flow chart for the LR code used for this project. Similar to the NB code, our LR code also imports the data by chunk and splits the data into Y and X matrices. The X matrix is then updated by standardizing each row with respect to their minimum and maximum values. A new column with unit values is added at the left of the X matrix. This code assumes an initial weight matrix, W with all zero values. Then, it calculates the $P(Y|W, X)$ matrix by multiplying the W and X matrices. On the other hand, Y matrix is used to develop the “delta” matrix filling 1, when $y_i = Y_k$ otherwise 0. Later, “delta” and $P(Y|W, X)$ are used to update the W matrix with help of a learning rate and a penalty variable. This code checks the improvement after each iteration and compares it with a stopping criterion to see whether stopping or further iteration. If improvement is greater than the stopping criterion, the code continues the iterations and updates the W matrix. When improvement is smaller than the stopping criterion, the code stops the iteration and choose the final W matrix. This W matrix is then used to determine the classes of the testing dataset.

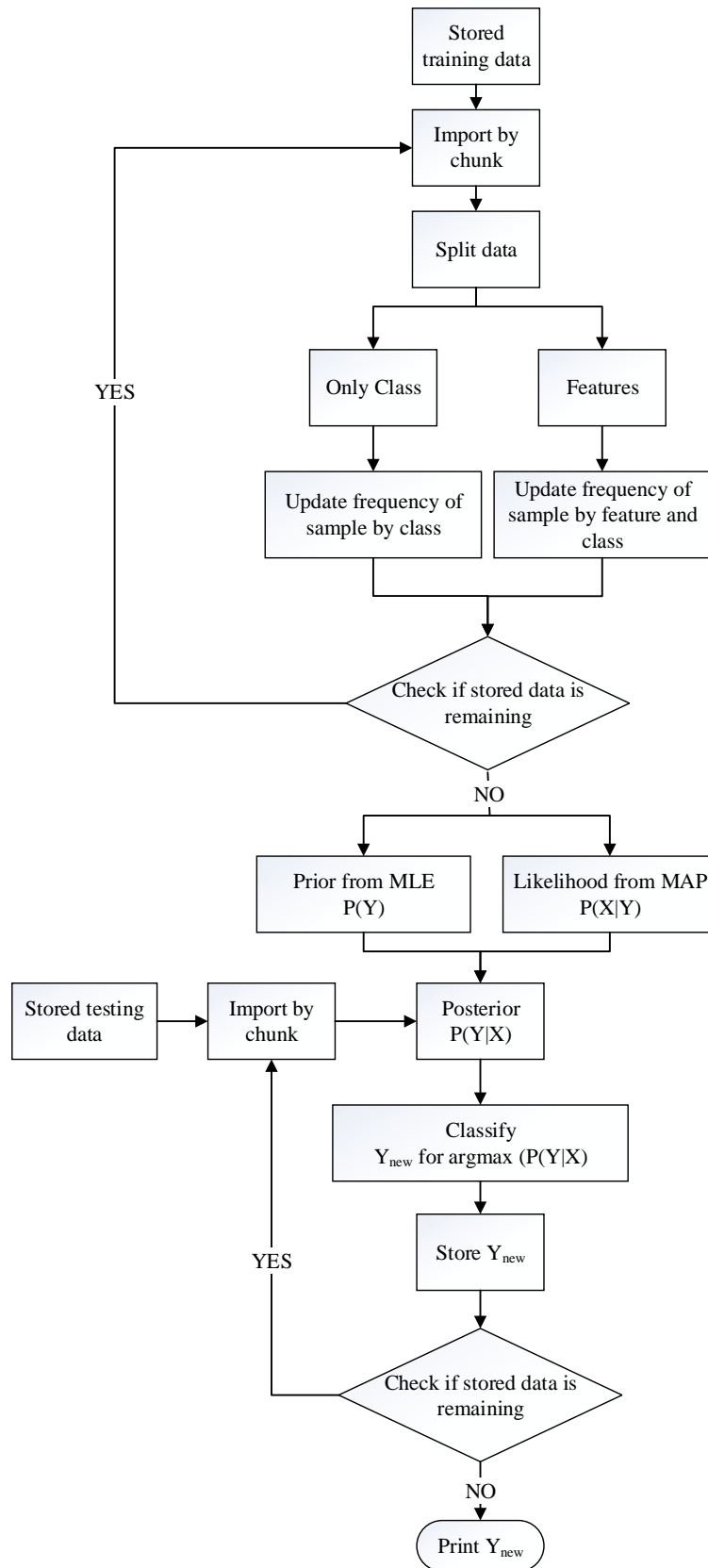


Fig. 1 Flow chart of NB code.

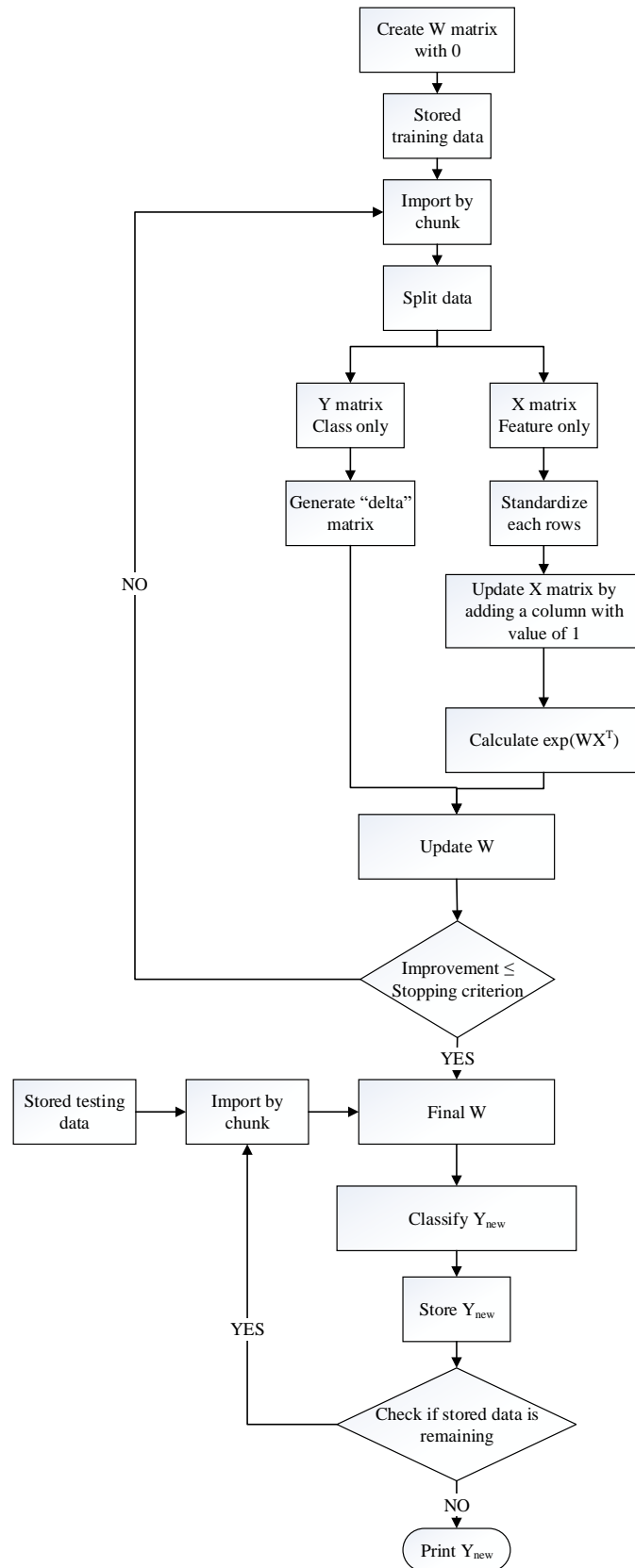


Fig. 2 Flow chart of LR code.

4 RESULTS: ANSWER TO QUESTIONS

Q.1 Answer

In this model, we don't consider that the probability of a feature given the class label is not independent of the other features. This basically implies that we are not considering the conditional independence of the features given the class. Estimating the feature-dependent probability parameters from a training set where the training set has a comparatively large amount of example data and number of features, would be a daunting task. Because we would have to count all the possible occurrence of the feature combination for each class label. For a two-class classification problem with N number of features, we will have to estimate $2 \cdot (2^N - 1)$ of parameters, where most of the parameter values would be zero as most of the feature combination won't be present in the training example data set. So, estimating all these conditional dependent probability parameters for a reasonably large data set is not feasible.

Q.2 Answer

We use beta value as a measure of the amount of hallucinated data example is added to the given original training data set. This extra padding of data is done to account for the missing features in the training dataset so that our estimated conditional probability parameters can handle those missing features in the new testing data set. Higher beta values correspond to the addition of the higher amount of extra example added for each feature to each class meaning higher smoothing. Fig. 3 shows the accuracy values for different beta values from the Kaggle. Fig. 4 shows the accuracy versus beta plot in semi-log scale. It shows that as we increase the amount of beta the smoothing increases to make each feature occurrence for each class towards similar values which in turn reduces rare features class distinguishing capability. So, when the beta value is higher, the test accuracy goes low. For lower feature values we also observe that the accuracy becomes low. This is due to that fact that lower beta values correspond to lower amount of smoothing or extra example padding, which makes our estimated conditional probability parameters to misfire to unseen features in new test data. For this reasons, mid-range beta values perform better in practice as they can account for the trade-off effect between extreme high vs extreme low beta values by doing an adequate amount of smoothing for all features, both present and missing. As expected, the maximum accuracy for our case when the beta is equal to 0.01.


Overview	Data	Kernels	Discussion	Leaderboard	Rules	Team	My Submissions	Submit Predictions
final_naive_bayes_1.6343073805321303e-05_result.csv 4 hours ago by Farhan add submission details							0.88367	<input type="checkbox"/>
final_naive_bayes_1_result.csv 4 hours ago by Farhan add submission details							0.85562	<input type="checkbox"/>
final_naive_bayes_0.1_result.csv 4 hours ago by Farhan add submission details							0.89193	<input type="checkbox"/>
final_naive_bayes_0.01_result.csv 4 hours ago by Farhan add submission details							0.89784	<input type="checkbox"/>
final_naive_bayes_0.001_result.csv 4 hours ago by Farhan add submission details							0.89371	<input type="checkbox"/>
final_naive_bayes_0.0001_result.csv 4 hours ago by Farhan add submission details							0.88957	<input type="checkbox"/>
final_naive_bayes_0.0001_result.csv 4 hours ago by Farhan add submission details							Error 	<input type="checkbox"/>
testing_naive_bayes_1e-05_result.csv 5 hours ago by Farhan							0.88249	<input type="checkbox"/>

Fig. 3 Kaggle screenshot for NB.

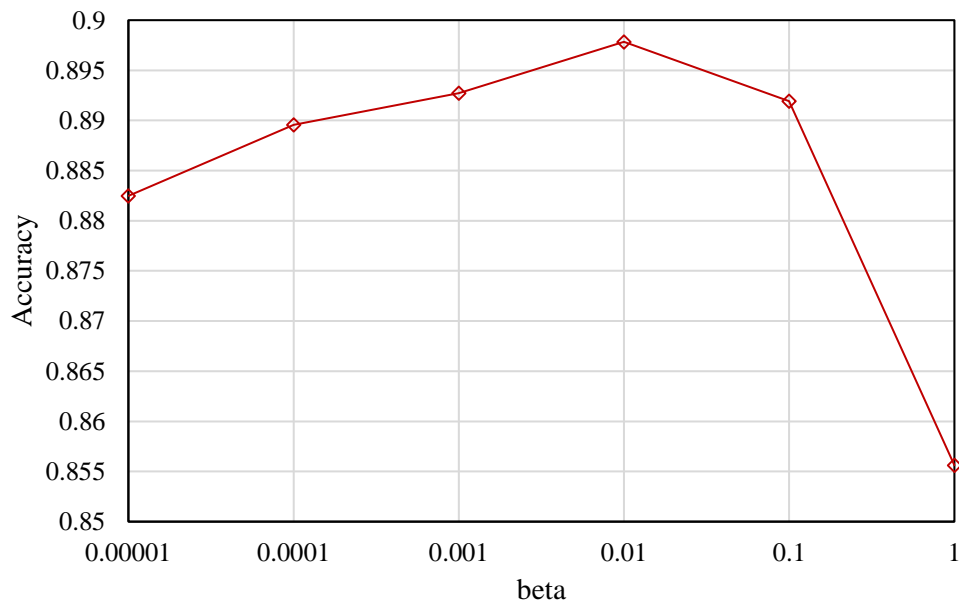


Fig. 4 Accuracy versus beta plot.

Q.3 Answer

We have used eta values (learning rate) of 0.001, 0.005 and 0.01 for each of the Lambda values (regularization parameter) of 0.001, 0.005 and 0.01. The found accuracies for different combination from the Kaggle are shown in Fig. 5. We have also plotted the accuracies for all different combination in Fig. 6. We observe that we achieve maximum test accuracy when $\eta=0.01$ and $\lambda=0.01$. Here, we achieved the best accuracy for the comparatively mid values of eta and Lambda. So, we can say that in our training process these values are the sweet spot for test set prediction as they give the best accuracy performance. Fig. 7 shows the changes of weight matrix over the iteration and the asymptote is observed after 1500 iterations.

Overview	Data	Kernels	Discussion	Leaderboard	Rules	Team	My Submissions	Submit Predictions
new_log_reg_eta_0.01_lambda_0.01_result.csv 14 hours ago by Farhan add submission details							0.81724	<input type="checkbox"/>
log_reg_eta_0.005_lambda_0.01_result.csv a day ago by Farhan add submission details							0.77561	<input type="checkbox"/>
log_reg_eta_0.005_lambda_0.005_result.csv a day ago by Farhan add submission details							0.76380	<input type="checkbox"/>
log_reg_eta_0.005_lambda_0.001_result.csv a day ago by Farhan add submission details							0.75642	<input type="checkbox"/>
log_reg_eta_0.01_lambda_0.01_result.csv a day ago by Farhan add submission details							0.81310	<input type="checkbox"/>
log_reg_eta_0.01_lambda_0.005_result.csv a day ago by Farhan add submission details							0.80011	<input type="checkbox"/>
log_reg_eta_0.01_lambda_0.001_result.csv a day ago by Farhan add submission details							0.79893	<input type="checkbox"/>
log_reg_eta_0.001_lambda_0.005_result.csv a day ago by Farhan add submission details							0.65544	<input type="checkbox"/>
log_reg_eta_0.001_lambda_0.01_result.csv a day ago by Farhan add submission details							0.65190	<input type="checkbox"/>
log_reg_eta_0.001_lambda_0.001_result.csv a day ago by Farhan add submission details							0.64186	<input type="checkbox"/>

Fig. 5 Kaggle screenshot for LR.

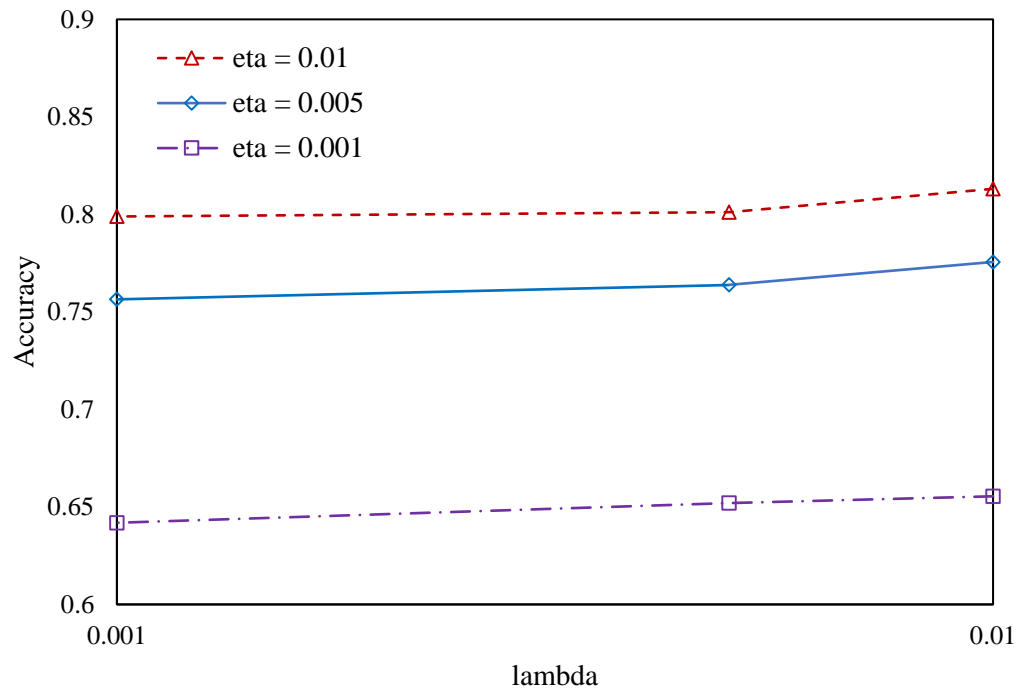


Fig. 6 Accuracy for different eta and lambda combinations.

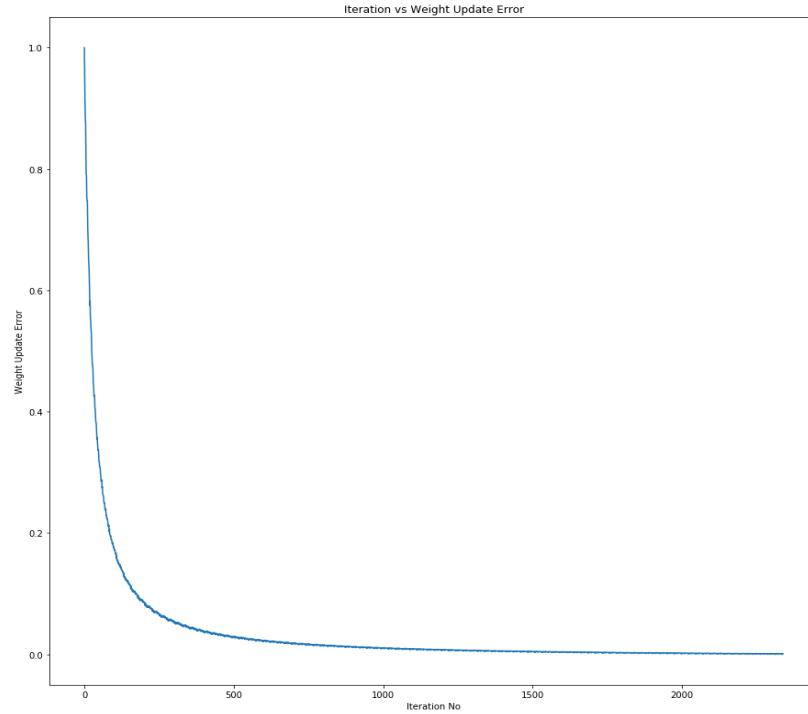


Fig. 7 Weight matrix update over iteration.

Q.4 Answer

The overall accuracy in testing data set was 81.72% for the optimized hyperparameter values.

To create the confusion matrix, we have used 25% of the training data as validation set and the rest of the 75% as the training data (as we did not have labelled data for the test data set to create the confusion matrix). Fig. 8 shows the confusion matrix. In this case our accuracy was 79.17%.

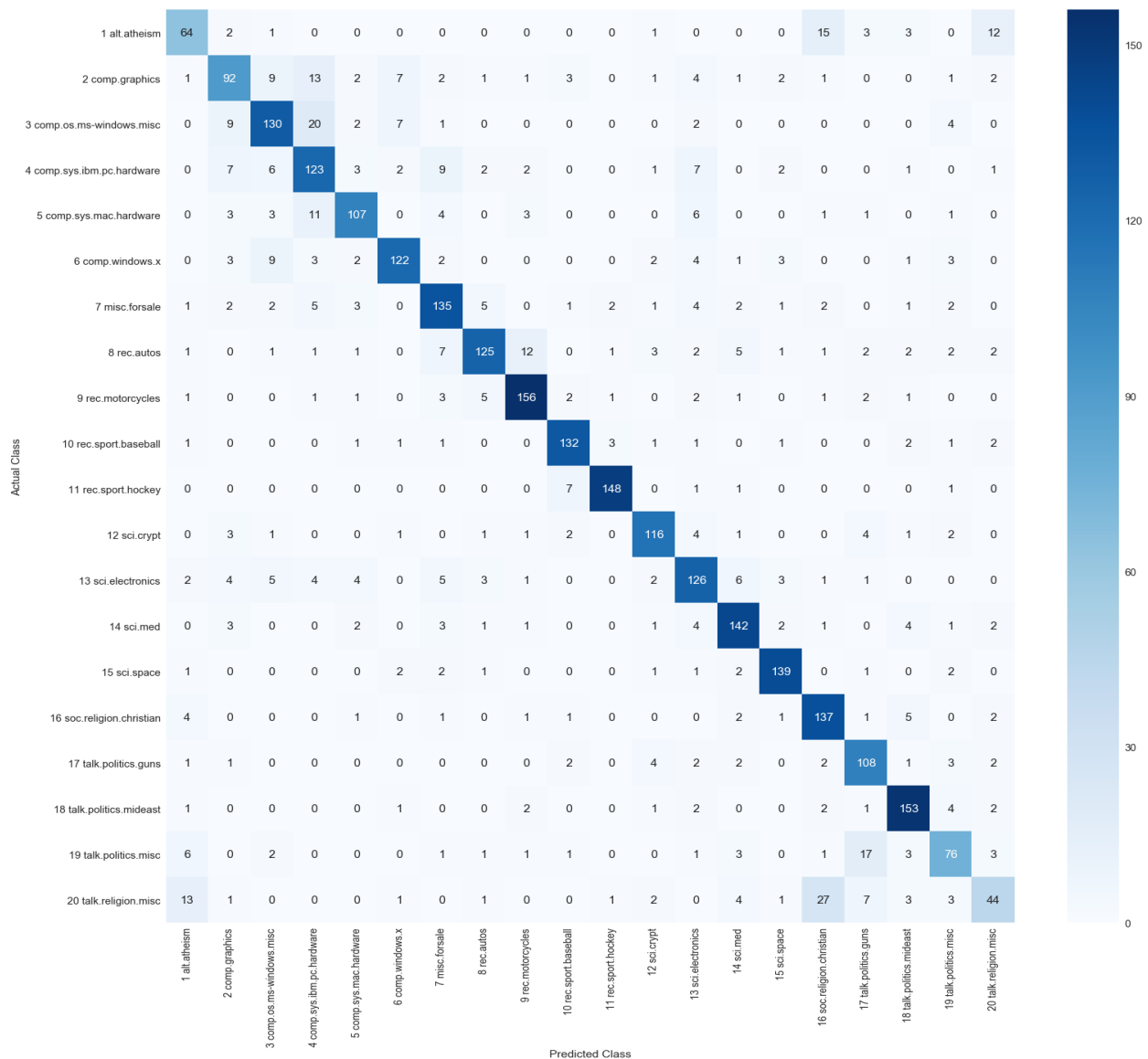


Fig. 8 Confusion matrix.

Q.5 Answer

There are some document classes which are more often being misclassified compared to other classes. When we observe the confusion matrix, we observe that; “2-comp.graphics”; “3-comp.os.ms-windows.misc”; “4-comp.sys.ibm.pc.hardware”, “5-comp.sys.mac.hardware” these three document classes are often misclassified as each other. The reason behind this is that these type of documents are of a common theme and share similar word occurrence. That’s why they are often misclassified among them Again, “1-alt.atheism”, “16-soc.religion.christian”, “20-talk.religion.misc” are also often misclassified among them due to the same reason as they have a common theme and share similar word count. Also, “17-talk.politics.guns”, and “19-talk.politics.misc” are misclassified among them being of similar document theme.

Q.6 Answer

In our training and test data set there are 61188 no of words which work as features for the document classification. But not all of them have same impact on the classification process. There are some very common words (for example: of, the, they) which occur almost similar no of time in each type of document, so their presence doesn’t have any significance for the classification process. On the other hand, there are some are words which mostly occur only in some certain type of document and their presence can play a crucial role in document classification process. These are the words on which the classifier relies more than the usual common word.

There are several approaches to rank the words (feature) based on how much they affect the classification; for example: Mutual Information, chi-square test. In our project we have used “Mutual Information” to rank the words based on their importance in the classification process by evaluating the mutual information between class and word. The mutual information between any two discrete variables such as X and Y can be compute using Eq. (7).

$$I(X, Y) = \sum_x \sum_y P(X, Y) \log \frac{P(X, Y)}{P(X)P(Y)} \quad (7)$$

Where, $P(X)$ and $P(Y)$ are the probability of X and Y respectively, $P(X, Y)$ is the joint probability of X and Y. We also know that when the two variables are mutually independent, the joint probability is only product of individual probabilities that means $P(X, Y) = P(X)P(Y)$. In such scenario, $\log(P(X, Y)/P(X)P(Y))$ would be zero. On the other hand, this parameter is greater than zero, when the variables are mutually dependent. We also assumed that more the dependency indicates higher values of $\log(P(X, Y)/P(X)P(Y))$. Since we have already computed some probabilities variables, we modified the mutual information score as shown in Eq. (8). We have applied the score on each word and class combinations to find out the rank of the words based on how much the classifier ‘relies on’ them when performing its classification.

$$I(X,Y) = \sum_y \log \frac{P(X|Y)P(Y)}{P(X)P(Y)} \quad (8)$$

In our ranking of the important words for classification, we observe that the ones with higher scores are not the common words, rather mostly rare words occurring mainly one or two classes heavily. For example, the word ranking top 3 are “wolverine”, “nhl” and “stephanopoulos”. Below we have plotted their frequency distribution among different classes In Figs. 9, 10, 11. In their frequency distribution plot, we observe that all three are mainly associated with one class, that’s why that word plays a vital role in classifying document of that class.

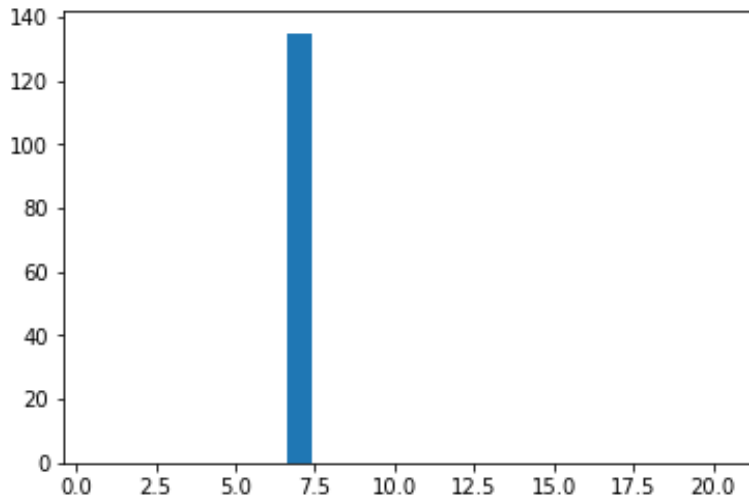


Fig. 9 Frequency distribution of “wolverine”, which occurs only in 7-misc.forsale

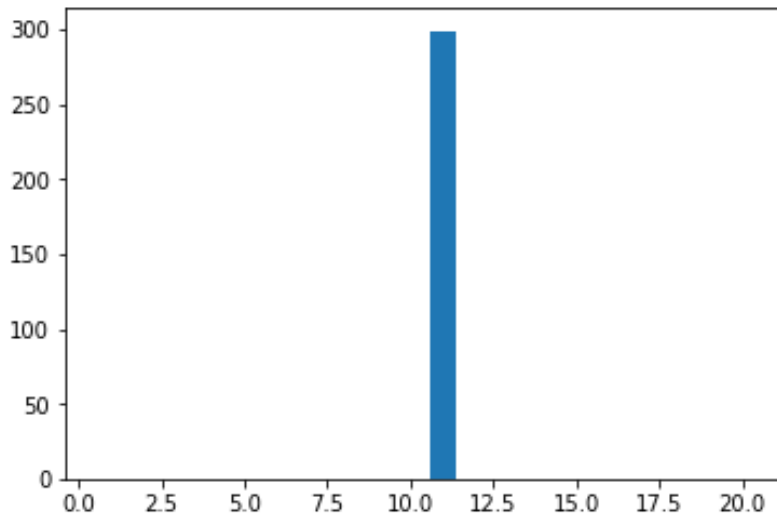


Fig. 10 Frequency distribution of “nhl”, which occurs only in 11-rec.sport.hockey

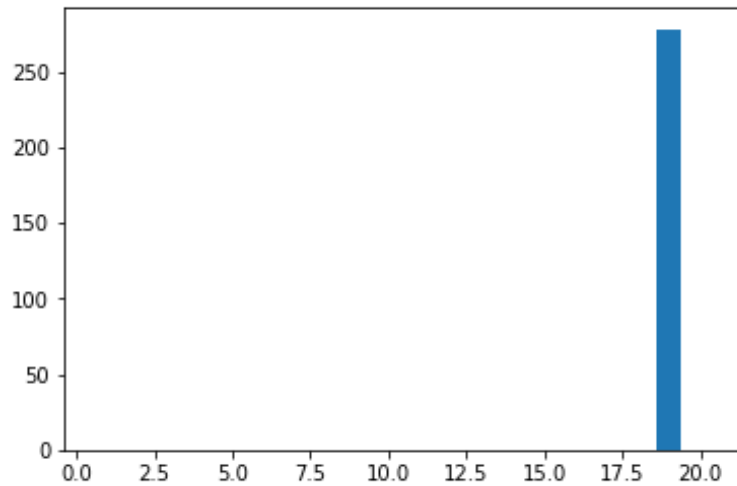


Fig. 11 Frequency distribution of “Stephanopoulos”, which occurs only in 19-talk.politics.misc

On the other hand, the words ranked lowest, are mostly common words occurring in almost similar frequency in all document class. The lowest three are “and”, “only” and “to”. Below we have plotted their frequency distribution for all classes in Figs. 12, 13, and 14 and we observe that they occur almost similar no of time in each document.

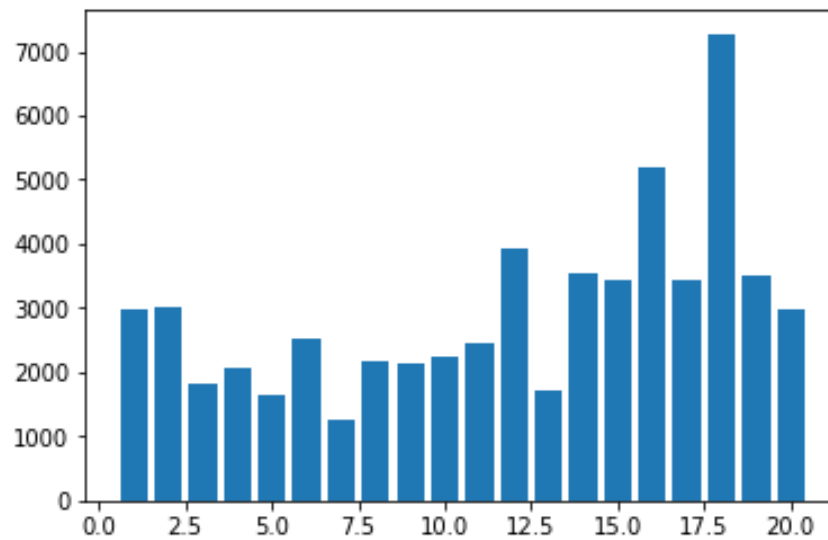


Fig. 12 Frequency distribution of “and” which occurs in similar distribution in all the classes

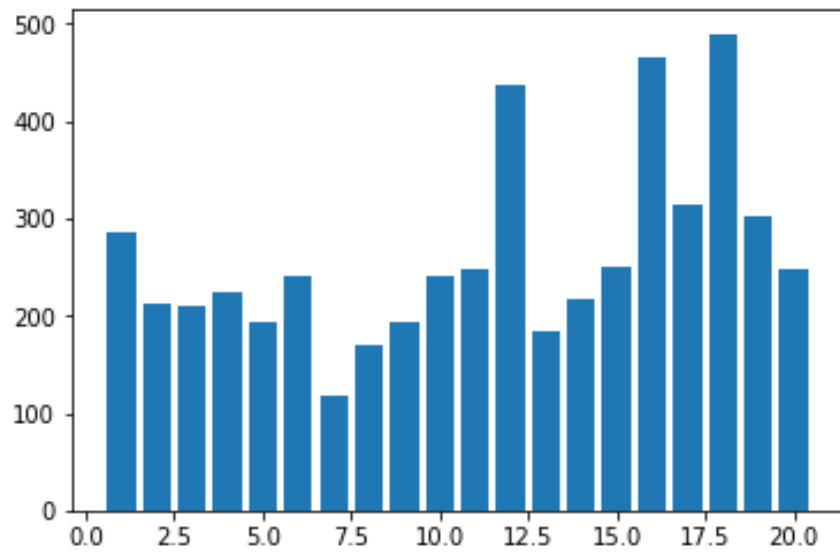


Fig. 13 Frequency distribution of “only”, which occurs in similar distribution in all the classes

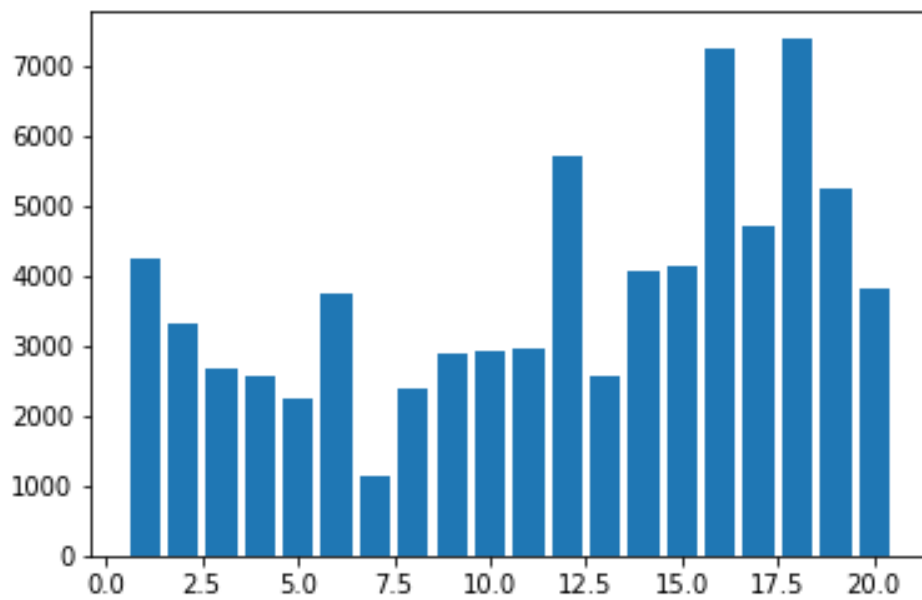


Fig. 14 Frequency distribution of “to”, which occurs in similar distribution in all the classes

We also used only 5000 highest scored words in NB algorithm for predicting classes for the testing data and got 62.68% accuracy, which is significant.

Q.7 Answer

The top hundred words with the highest measure using $\beta = 1/|V|$, where $|V|$ =no of words are given below (in descending order and from left to right):

"wolverine"	"alomar"	"istanbul"	"soderstrom"
"nhl"	"bikes"	"canucks"	"iivx"
"stephanopoulos"	"nsa"	"optilink"	"arabs"
"armenian"	"goalie"	"crypt"	"baalke"
"sabretooth"	"espn"	"mets"	"recchi"
"pitcher"	"fuhr"	"oname"	"xlib"
"liefeld"	"oilers"	"mattingly"	"rlk"
"leafs"	"widget"	"hirschbeck"	"argv"
"armenians"	"hobgoblin"	"potvin"	"gainey"
"athos"	"lunar"	"countersteering"	"dodgers"
"nyr"	"gilmour"	"pitches"	"batting"
"candida"	"hitter"	"winfield"	"sabres"
"bruins"	"args"	"bagged"	"siggraph"
"ripem"	"xdm"	"denning"	"azerbaijani"
"cramer"	"rbi"	"gant"	"gaza"
"ahl"	"keown"	"clemens"	"quadra"
"hulk"	"pitching"	"bmug"	"marvel"
"islanders"	"crypto"	"powerbook"	"sdpa"
"punisher"	"nyi"	"playoff"	"lemieux"
"ei"	"rayshade"	"argic"	"xmu"
"iisi"	"biker"	"israelis"	"pitchers"
"comics"	"ndet"	"serdar"	"azerbaijan"
"armenia"	"stl"	"orbiter"	"adb"
"lindros"	"mcfarlane"	"mustang"	"sumgait"
"lciiii"	"clh"	"pov"	"radiosity"

A few lowest ranked words are given below:

"and"	"from"	"as"
"only"	"at"	"another"
"to"	"like"	"all"
"if"	"are"	"some"
"have"	"either"	"just"
"has"	"other"	"know"
"time"	"on"	"is"
"this"	"will"	"you"
"so"	"with"	"give"
"one"	"make"	"way"
"the"	"since"	"once"
"an"	"but"	"both"
"now"	"or"	"well"
"in"	"most"	"be"
"it"	"into"	"would"
"here"	"more"	

Q.8 Answer

In this project our training dataset was selected independently and randomly from the same distribution of our testing dataset which makes our training dataset free from being biased.

Below we have plotted the class distribution of the training dataset.

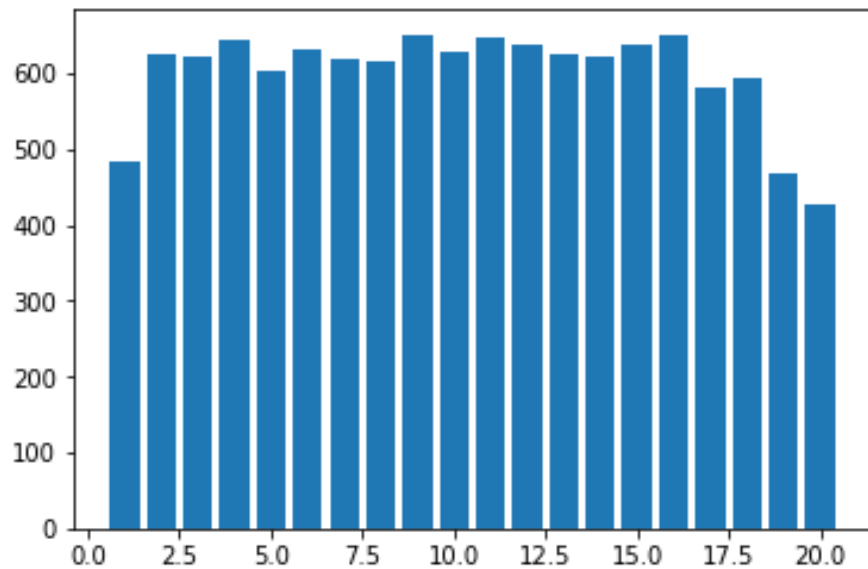


Fig. 15 Frequency distribution of all the 20 classes present in the training data

We can observe that our training dataset comprises of almost equal number of training example for each class. So, we don't have any bias towards any particular class.

But, although we don't have any bias in our training set example class distribution, when we look at our top ranked words with high impact on classification, (from the frequency distribution of top three ranked words shown in answer to the Q3) we notice that most of them are unique to one class and thus plays a crucial role in classifying that corresponding class. It means that, those words are biasing our parameters heavily towards the corresponding class. In case, our future training dataset is missing those class-biased top ranked words in those particular class of documents, our learned parameters might not perform well to identify the corresponding class.

5 DISCUSSIONS

In this project, we implemented the NB and LR classifiers to categorize the documents based on their contents. Findings are given as follows:

- We got higher accuracy for the NB (89.74%) code compare to LR code (81.72).
- Training of Naive Bayes method is faster than Logistic Regression method.
- Training using 5000 top words selected by Mutual Information gave test accuracy of 62.68%.

REFERENCES

- [1] T. M. Mitchell et al., Machine Learning. wcb, 1997.
- [2] www.wikipedia.org