Binary Classification using N-Gram Model on Text Data

CIS6930 – Introduction to Data Mining
Final Project Report (Group-8)

Swarabarna Sarkar (UFID: 7416-1498), Deep Chakraborty (UFID: 2115-1818)
Debarshi Mitra (UFID: 3381-3136), Avirup Chakraborty (UFID: 5291-4909)

***Abstract*— This report describes our implementation of a pre-rating system of the Amazon product review data which determines the helpfulness of a review given by a customer and whether it would be beneficial to show that on the product page or not. It is a binary classification problem which predicts the helpfulness score as 1 if a certain review is actually helpful and predicts 0 otherwise. This algorithm uses the N-Gram model to convert the review text into TF-IDF values and then presents a comparative analysis of four common classification algorithms that are implemented without the use of any existing machine learning libraries in Python.**

I. INTRODUCTION

The quality of the customer reviews given for any products or other services always affects the business of any company. Different companies have tried various methods to rate and filter the quality of the customer reviews. For example - Reddit uses an up-vote system, Quora fosters a community that values high-quality responses over low-quality ones, and Amazon allows for its users to rate the "helpfulness" of reviews left on their products. But, Amazon has both top reviews and the recent reviews shown on the product page. The top reviews are based on the usefulness of the reviews decided by the users. The recent reviews make a feature on the product page which may be of poor quality and may not always be helpful. Hence, such reviews displayed on the product page affects Amazon's business to some extent.

We propose a binary classification system which categorizes any new reviews into "helpful" or "not helpful" class before any such review makes a feature on the Amazon's product page. Our proposed system would help in filtering out poor quality reviews and would not allow them to be displayed at the top. It will use Amazon's product reviews dataset to train the models so that it can make a binary classification of any new review coming as a test case. We will be using NLP packages to pre-process the data so that we extract the features from the reviews and then we will be performing comparative analysis among standard classification algorithms like Gaussian Naive Bayes Classification, Decision Tree, Logistic Regression, and K-Nearest Neighbours.

II. LITERATURE SURVEY

Product reviews on online marketplaces are most helpful when they affect the sales and the online behaviour of users. Many researchers have worked on assessing the effect of product reviews on sales. Authors in [1] have shown that the relationship between numeric rating of reviews and sales, while the researchers in [2] demonstrated the association between review volumes and sales. They have also suggested that product sales can be affected by reviewer disclosure of identity [3]. This association is valid for all qualities of product. McAuley J., Pandey R., and Leskovec J. developed a network of substitutable and complimentary products using the text of product reviews [4]. Substitutes are the products that can be purchased instead of the given product, while compliments are the products that can be purchased in addition to the given product. The researchers in [5] modelled similar relationships between substitutes and compliments using images of the products. Hao et al. suggested techniques to determine whether a review will get a vote about helpfulness or not [6].

III. ALGORITHM DESCRIPTION & IMPLEMENTATION

In this project, we have implemented four major classification algorithms without the use of any python libraries. In all the implementations, we have calculated the interdependence of the variables with the class labels using the chi-square metric. Then, we only considered the top ten features that are mostly dependent on the class labels. Also, since the data class distribution of the two classes is highly unbalanced, so the training and the test data is split using stratified sampling method. We briefly describe each algorithm and our implementation in this section below:

Logistic Regression – Logistic regression is one of the most popular classification algorithm which is used in the prediction of probability of categorical and dependent variables. This dependent variable is of binary value containing data which is coded as either 0 representing failure/no or 1 representing success/yes. Logistic regression is a predictive analysis and describes relationship between a binary dependent variable and one or more independent variables. The logistic regression algorithm runs on the following basic assumptions:

a. Only the relevant variables are to be included from the dataset.
b. Logistic regression demands large sample datasets.

c. The dependent variable needs to be binary in case of binary logistic regression.
d. There is a linear relation between the independent variables and the log odds.
e. There must be minimal or zero multicollinearity in the model.

In our implementation of this algorithm, the score is generated for each of the selected feature using the sigmoid function as given below:

$$S(t) = \frac{1}{1+e^{-t}}.$$

The graph of the sigmoid function is given below:

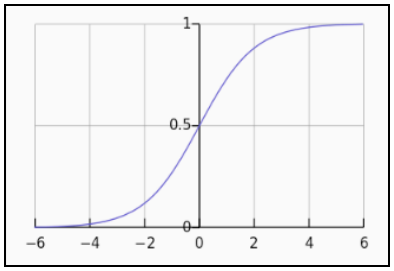

Fig 1 – Plot of the sigmoid function

The weightage of each of the feature is updated using the scores for each step of the gradient ascent. A log-likelihood function is used to check the convergence of the weights after each iteration to find the maximization point. Based on the weights which are computed, the class labels of the test instances are determined.

The graph of the log-likelihood function is given below:

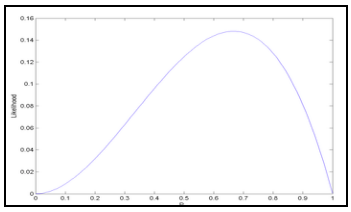

Fig 2 – Plot of the log-likelihood function

Gaussian Naïve Bayes – Gaussian Naïve Bayes deals with continuous data. The continuous values associated with each of the class are dispersed with respect to Gaussian distribution. In other words, the Naïve Bayes algorithm can get extended to real value attributes assuming Gaussian distribution. Gaussian Naïve Bayes is this extension of this Naïve Bayes. The formula expresses the Gaussian Naïve Bayes as below:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

The Gaussian distribution function plot is given below:

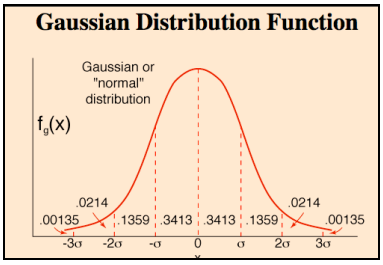

Fig 3 – Plot of the Gaussian distribution function

In our application of the Gaussian Naïve Bayes algorithm, the selected features are separated based on classes. The training data is summarized by computing the mean and standard deviation for each feature based on classes. A Gaussian Function has been used to estimate the probability of a given attribute value, given the known mean and standard deviation for the attribute estimated from the training data. Finally, the probabilities of all the attribute values for a data instance are considered to result in a map of class values to probabilities which is used to predict the labels on test dataset.

K-Nearest Neighbours – The K-nearest neighbour algorithm is used both for classification and regression and is a non-parametric method. The input for the K-nearest neighbour algorithm has the nearest training examples from feature set. The output for the K-nearest neighbour classification is a class membership. The object is selected taking the majority vote of the neighbours. The object is assigned the most common class among all the nearest k neighbours. For example, if k =1, the object gets assigned the same class in which that nearest neighbour belongs. The formula to find the distance in n-dimensional space is given as:

In general, for an *n*-dimensional space, the distance is

$$d(p,q) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + \cdots + (p_i-q_i)^2 + \cdots + (p_n-q_n)^2}.$$

The figure below shows the computation of the KNN algorithm when K=3. The green point is the test data and the Euclidean distance of that point is computed with all the training data points. The closest 3 training data neighbours of the test data is taken and the class of the test data is predicted based on the class labels of the 3 neighbours using a voting system. In this case, the green point will belong to the class of the red points.

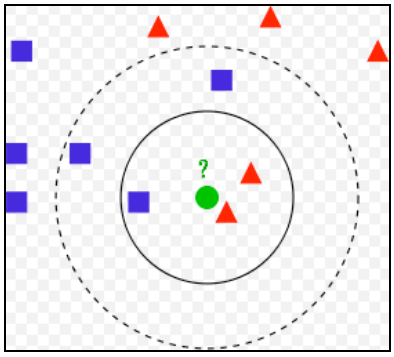

Fig 4 – KNN Algorithm representation

In our implementation, the Euclidean distance for each test instance is computed from the entire training set for the selected 10 features. The top k similar neighbours from the training set are taken and then the class of each test instance is determined by the voting of the class labels of the neighbours. The value of k is taken as odd in order to avoid ties while selecting class labels. We experimented with k=3 and 5 and got better results with k=3 which is only shown in this report.

Decision Tree (CART Algorithm) – The decision trees are a powerful and popular prediction method since it can easily explain the reason of a certain prediction choice of the test data instances. The CART stands for Classification and Regression Trees which are used to refer to the Decision Tree algorithms that can be used for classification or regression predictive modelling problems. The output tree of the CART algorithm is always a binary tree. It uses the GINI Index to measure the class impurity of data points in each node of the tree using the following formula:

$$Gini = \sum_{i \neq j} p(i)p(j)$$

A sample decision tree diagram generated using the CART algorithm is given below:

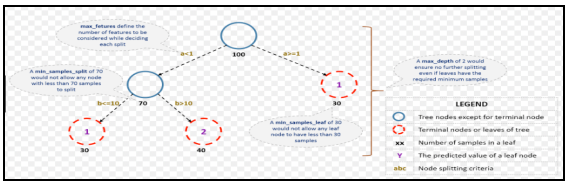

Fig 5 – Decision Tree representation

In our implementation, we compute the GINI Index of each of the selected features in order to select a certain feature value for the root node. At each subsequent level, we implemented two types of the decision tree – either all the features were again considered for computation of the GINI Index to detect the feature value in the lower levels which may lead to selection of same feature at every level, or, the features of the previous levels were not considered in the subsequent levels. After the construction of the decision tree, the class labels of the test instances are predicted based on the decision tree. The second implementation of the decision tree gave us better results and is only shown in this report.

IV. DATASET DESCRIPTION

The dataset is taken from the repository provided by UCSD at Dr. Julian Mc Auley's research page as given below:

http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/

The data consists of all the product reviews present in the Amazon Marketplace divided in terms of categories like Books, Electronics, Clothes etc. and the dataset size ranges from few kilobytes to around 25 GB. All the data is given in JSON format in respective zip files. We narrowed our experiments to two the following datasets –

- Reviews of Clothes, Shoes and Jewellery which is around **2.92 GB** and we refer to it as our small dataset. The filename for this dataset is ***reviews_Clothing_Shoes_and_Jewelry.json.gz.***
- Reviews on Electronics which is around **5.47 GB** and we refer to it as our large dataset. The filename for this dataset is ***reviews_Electronics.json.gz.***

One example of the sample data is given below:


```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the
piano.  He is having a wonderful time playing these old hymns.
The music  is at times hard to read because we think the book
was published for singing from more than playing from.  Great
purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```


Fig 6 – Sample JSON of input data

Each record represents each user who bought and reviewed products of that category. There are nine columns out of which and is summarized below:

- **reviewerID** – ID of the Customer.
- **asin** – ID of the Product.
- **reviewerName** – Name of the Customer.
- **helpful** – The helpfulness of a review. It is an array which has two numbers like [2,3]. The first number represents the number of positive reviews about the product that the particular customer has given. The second number represents the total number of product reviews given by the customer.
- **reviewText** – The detailed text of the product review.
- **overall** – The rating of the product as given by the customer.
- **summary** – The summary of the product review.
- **unixReviewTime** – time of the review (Unix time).
- **reviewTime** – time of the review (raw).

The columns which we used for our experiments are – **helpful, overall and reviewText.** We extracted the two numbers from the helpful column into two columns – **helpful_num** and **helpful_denom** which has been used to generate the binary class labels**.**

The summary of the small dataset and the large dataset used in our experiments is given below:

	overall	helpful_num	helpful_denom	Helpful
count	89664.000000	89664.000000	89664.000000	89664.000000
mean	3.666577	23.133733	26.763249	0.909663
std	1.564947	150.920413	154.329765	0.286666
min	1.000000	0.000000	11.000000	0.000000
25%	2.000000	11.000000	13.000000	1.000000
50%	4.000000	14.000000	16.000000	1.000000
75%	5.000000	22.000000	25.000000	1.000000
max	5.000000	37539.000000	38004.000000	1.000000

Fig 7 – Summary of the small input dataset

	overall	helpful_num	helpful_denom	Helpful
count	361248.000000	361248.000000	361248.000000	361248.000000
mean	3.385253	29.915673	36.636123	0.817591
std	1.642396	111.034865	115.316165	0.386182
min	1.000000	0.000000	11.000000	0.000000
25%	2.000000	10.000000	14.000000	1.000000
50%	4.000000	14.000000	19.000000	1.000000
75%	5.000000	26.000000	32.000000	1.000000
max	5.000000	30735.000000	31453.000000	1.000000

Fig 8 – Summary of the large input dataset

The next section describes how the input data is pre-processed and converted to the format for applying the machine learning algorithms for our binary classification problem.

V. DATASET CLEANING & PRE-PROCESSING

All the input JSON data is pre-processed to the format for applying the machine learning algorithms. The pre-processing steps are as follows:

- The JSON data is converted to Data Frame in order to view the data in a tabular format. The sample data frame rows are given below:

reviewerID	asin	reviewerName	helpful	unixReviewTime	reviewText	overall	reviewTime	summary
A3H1W0DWLW5…	0000031887	ashley	[26, 28]	1286668800	I bought this as par…	5	10 10, 2010	Very happy customer
A12OFS8WQP8…	0000031887	crelieu	[20, 23]	1297123200	I looked all over amazon…	5	02 8, 2011	Better than i expected
A1M7MWBQJEJ…	0000031887	Rachel A. Reinhart	[9, 12]	1260662400	I'm really happy with …	5	12 13, 2009	Great Tutu for a Great…
AQ2NZ5X891V…	0000031887	SA Librarian	[14, 14]	1295308800	I ordered this tutu f…	5	01 18, 2011	Tutu Cute!
A1BRPZYSTVC…	0000031887	shellsbells…	[12, 17]	1323993600	The product I received …	1	12 16, 2011	Cheap Waste of Money!

Fig 9 – Sample DataFrame of input data

- The relevant columns **helpful, overall and reviewText** are only retained and rest are dropped**.**
- Two columns **helpful_num** and **helpful_denom** are generated from the **helpful** columns. The rows with **helpful_denom** less than ten are removed since those rows represent the customers who reviewed infrequently.
- All the rows with NULL values are checked and removed from the DataFrame.
- The ratio of the **helpful_num** to **helpful_denom** is computed and is used to generate the binary class labels. The class label is set to 1 if it is more than the user-specified threshold value (taken as 0.5) else it is set to 0 and assigned to a new column called **Helpful**. The summary of the class label distribution of the data for both the small and the large dataset is given below:

Dataset	Class Label - 0	Class Label - 1	Total
Small Dataset (2.92 GB)	8100	81564	89664
Large Dataset (5.47 GB)	65895	295353	361248

Table 1 – Class Distribution of the input data

- A heat map is generated to check the correlation between the input columns with the class labels. The figure is given below:

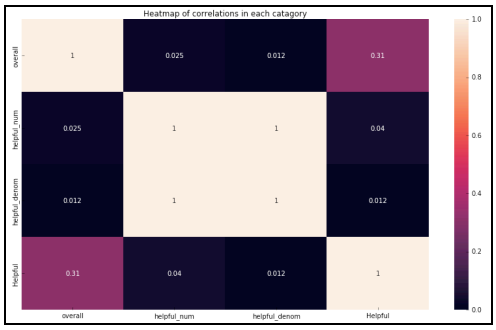

Fig 10 – Correlation heat map of the input data

The correlation between the column Overall and the class label was found to be 0.31 which is very low. This low correlation made the problem interesting and we had to do feature engineering from the **reviewText** column.

- The text data from the **reviewText** column is converted by performing the basic steps of natural language processing like – **Stemming**, where the words are converted to the root format. Then, the stop words like – a, an, the etc. are removed along with punctuations.
- The converted text is tokenized and the Term-Frequency (TF) and the Inverse Document Frequency (IDF) is computed in the unigram model to generate the features with the TF-IDF values of each words as features of the input dataset.
- Finally, the pre-processed data is saved into a pickle file for future use.

VI. EXPERIMENTAL RESULTS

We tried various values for the input parameters of each of the algorithm for the small as well as large dataset. The best results are presented in the tables given below:

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1	Running Time (mins)
Decision Tree	90.96	90.97	99.99	95.27	242.17 (1st Exec.) 0.02 (Rest)
K-Nearest Neighbour (k=3)	89.68	91.33	97.95	94.53	45.55
Logistic Regression (1000 Iterations)	77.04	92.8	81.04	86.52	0.1
Gaussian Naïve Bayes	62.28	95.87	61.17	74.68	0.03

Table 2 - Results for Small Dataset of size 2.92 GB with 89664 rows

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1	Running Time (mins)
Decision Tree	81.77	81.81	99.93	89.97	2476.72 (1st Exec.) 0.01 (Rest)
K-Nearest Neighbour (k=3)	79.5	82.85	94.48	88.29	1083.6
Logistic Regression (1000 Iterations)	74.9	86	82.78	84.36	0.37
Gaussian Naïve Bayes	63.64	89.41	62.99	73.91	0.09

Table 3 - Results for Large Dataset of size 5.47 GB with 361248 rows

Some additional results of Logistic Regression and Gaussian Naïve Bayes considering all the input features is given below:

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1	Running Time (mins)
Logistic Regression (837 Features and 100 Iterations)	90.88	91.08	99.74	95.21	0.16
Gaussian Naïve Bayes (837 Features)	77.74	95.41	79.35	86.64	0.12

Table 4 - Results for Small Dataset with 837 features

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1	Running Time (mins)
Logistic Regression (1298 Features and 100 Iterations)	82.14	82.54	99.13	90.08	7.4
Gaussian Naïve Bayes (1298 Features)	70.43	90.36	71.45	79.8	2.67

Table 5 - Results for Large Dataset with 1298 features

The comparative analysis of all the performance metrics for each of the algorithm is shown below:

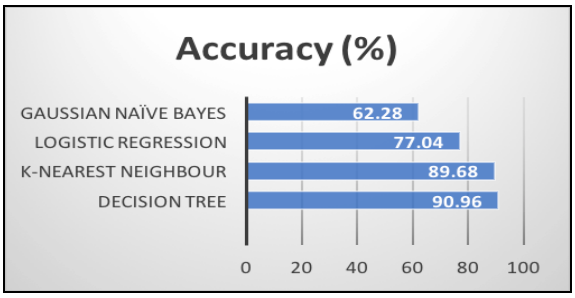

Fig 11 – Accuracy of Small Dataset with 10 features

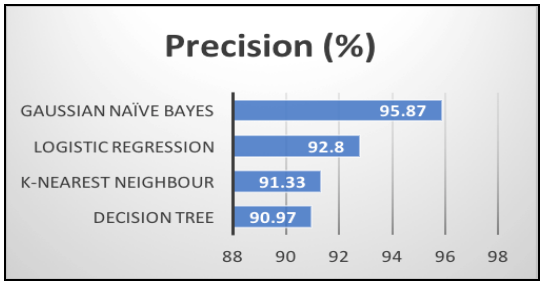

Fig 12 – Precision of Small Dataset with 10 features

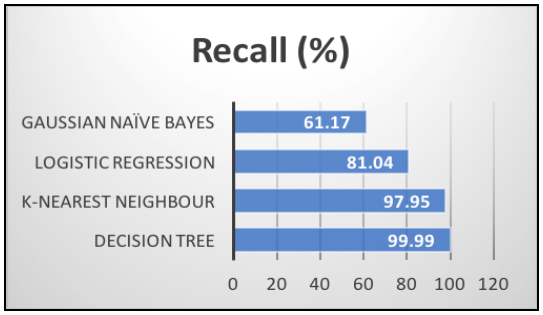

Fig 13 – Recall of Small Dataset with 10 features

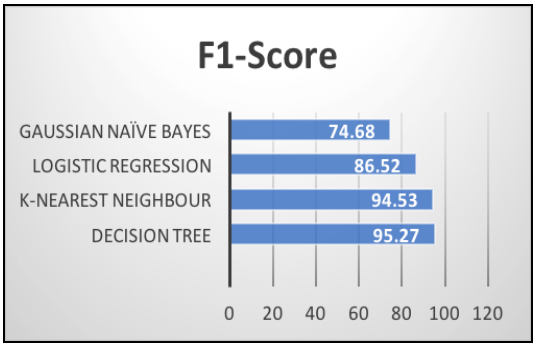

Fig 14 – F1-Score of Small Dataset with 10 features

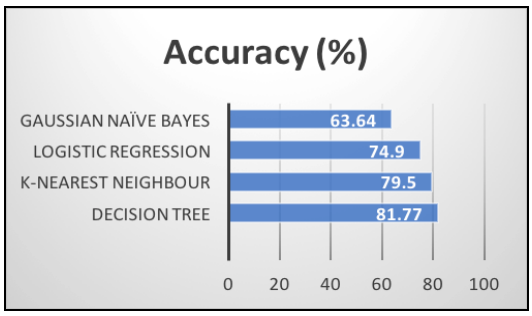

Fig 15 – Accuracy of Large Dataset with 10 features

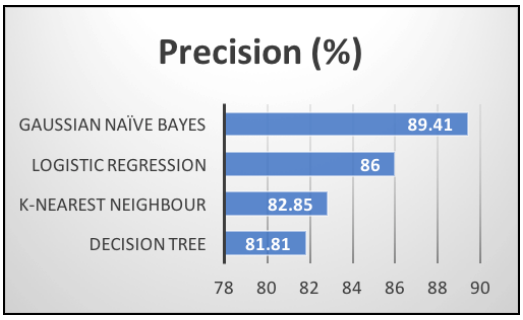

Fig 16 – Precision of Large Dataset with 10 features

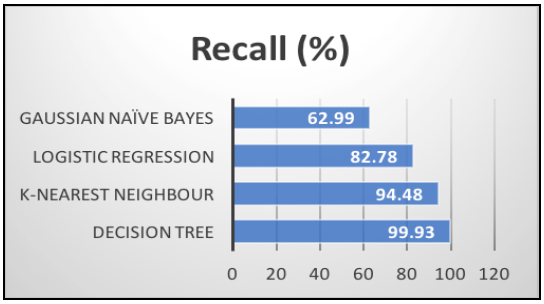

Fig 17 – Recall of Large Dataset with 10 features

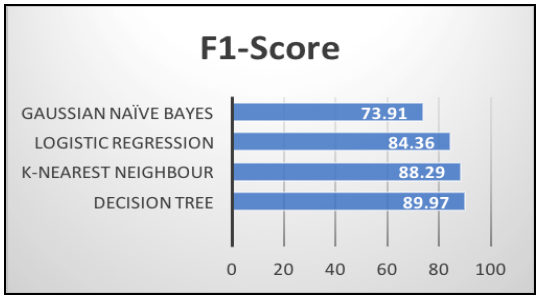

Fig 18 – F1-Score of Large Dataset with 10 features

The Decision Tree and KNN are the best performing algorithms in terms of all the performance metrics for the ten best features but they are slower to train than Logistic Regression and Gaussian Naïve Bayes.

We also applied Logistic Regression and Gaussian Naïve Bayes to the small and large dataset for all the features and the comparative analysis is given below:

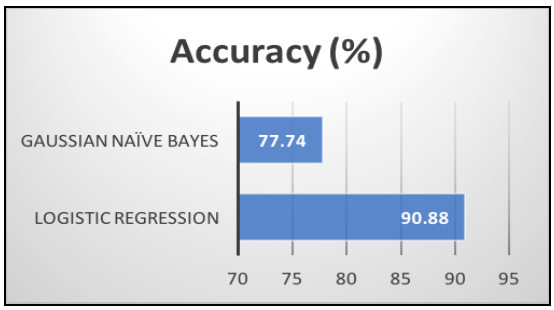

Fig 19 – Accuracy of Small Dataset with 837 features

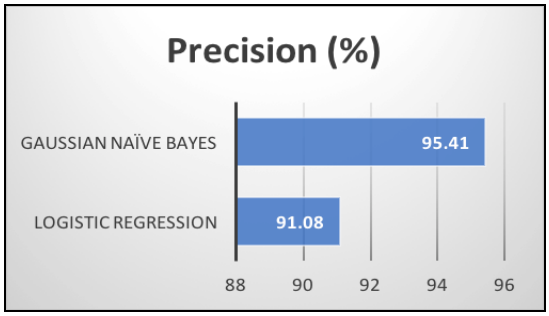

Fig 20 – Precision of Small Dataset with 837 features

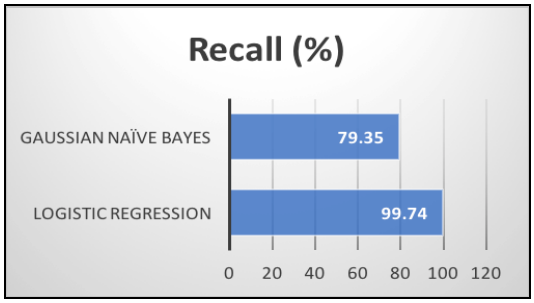

Fig 21 – Recall of Small Dataset with 837 features

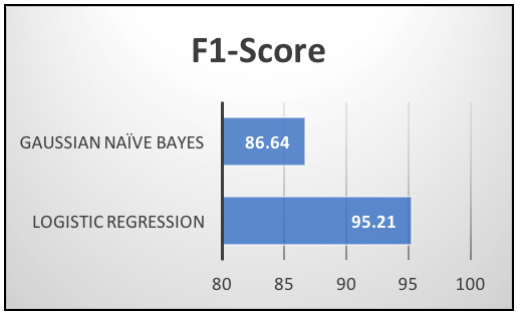

Fig 22 – F1-Score of Small Dataset with 837 features

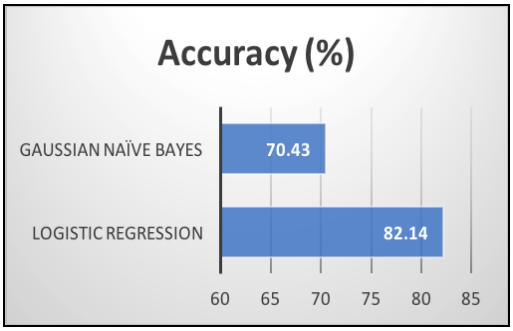

Fig 23 – Accuracy of Large Dataset with 1298 features

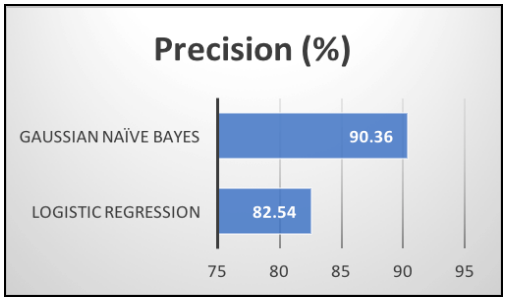

Fig 24 – Precision of Large Dataset with 1298 features

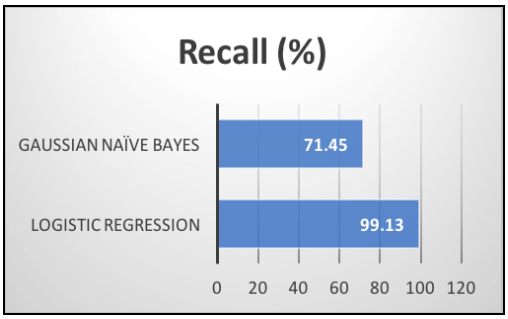

Fig 25 – Recall of Large Dataset with 1298 features

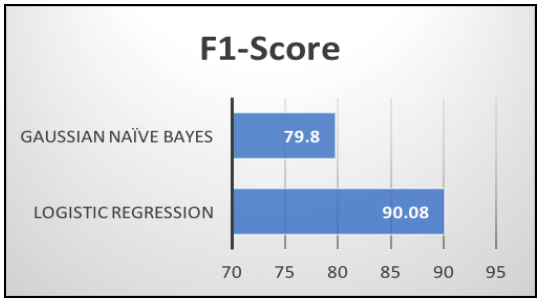

Fig 26 – F1-Score of Large Dataset with 1298 features

The performance of Logistic Regression improved with the increase in number of features but we cannot firmly say that it's the best among the four algorithms since we were unable to execute the other two algorithms due to running time limitations caused due to lack of hardware resources.

VII. LIMITATIONS AND FUTURE WORK

The implementations of all the algorithms have been done from scratch and hence, they are not fully optimized in terms of running time. Also, all the experiments have been performed in a system with 4 GB of RAM which also led to average running time performance. These are prime limitations of our implementations.

As a part of the future work, we can do the following:

- Data columns like "**overall**" and "**summary**" can be used for feature generation along with "**reviewText**" to check the model performance.
- Extraction of additional useful words from the feature set can be done using a vocabulary set.
- Perform additional feature engineering, such as proper spell checking for the reviews.
- Explore the effect of not doing pre-processing on the reviews.
- Explore additional distance metrics apart from Euclidean Distance for K-Nearest Neighbours Algorithms.
- Running time of the Decision tree and the KNN algorithms can be improved by parallelization and also by using better hardware resources.
- Many other classification algorithms like Random Forest, SVM etc. can be implemented and the performance metrics can be compared with the three algorithms which we already implemented here.

VIII. CONCLUSION

This project was interesting in the sense that it was not like other mainstream text classification problems where we are always trying to reduce the dimensionality of the text data by applying various techniques like combining synonyms, stemming words, and correcting spelling mistakes. We applied some of the techniques but it was not mandatory since people found the reviews unhelpful because of these imperfections. Another challenging aspect of this project was to implement all the classification algorithms from scratch without using the existing python libraries. This helped in improving our concepts of these algorithms along with our programming skills.

IX. REFERENCES

[1] C. Dellarocas, N. F. Awad, and X. M. Zhang, "Exploring the value of online product ratings in revenue forecasting: The case of motion pictures," 2007, working Paper, Robert H. Smith School Research Paper.

[2] C. Forman, A. Ghose, and B. Wiesenfeld, "Examining the relationship between reviews and sales: The role of reviewer identity disclosure in electronic markets," Information Systems Research, vol. 19, no. 3, Sep. 2008.

[3] R. G. Hass, "Effects of source characteristics on cognitive responses and persuasion," in Cognitive responses in persuasion, R. E. Petty, T. M. Ostrom, and T. C. Brock, Eds. Lawrence Erlbaum Associates, 1981, pp. 1–18

[4] Inferring networks of substitutable and complementary products. J. McAuley, R. Pandey, J. Leskovec Knowledge Discovery and Data Mining, 2015.

[5] Image-based recommendations on styles and substitutes J. McAuley, C. Targett, J. Shi, A. van den Hengel SIGIR, 2015.

[6] Y. Y. Hao, Y. J. Li, and P. Zou, "Why some online product reviews have no usefulness rating?" in Pacific Asia Conference on Information Systems (PACIS 2009), 2009.