Sentiment Analysis of Customer Reviews

Aditya Choudhary
21b090001

Harsh Chauhan
21b090008

Soutrik Sarangi
20b090014

I. Introduction

Sentiment Analysis of customer reviews has widespread utilisation in industries oriented towards customers. It primarily helps businesses to evaluate their products and services through analysis which often runs on numerical values. Hence there is a need for classification of reviews and feedback to the corresponding numerical 'rating' system.

II. Data Procurement

We would like to thank Jianmo Ni, Jiacheng Li, Julian McAuley *(Empirical Methods in Natural Language Processing (EMNLP), 2019)* [1] for the dataset of reviews they have collated. We decided to settle for this dataset due to its comprehensive nature of large number of reviews spread over various product categories and rating levels. The data is in English language and hence provides ease of use and analysis considering the fact English has well developed tools for analysis.
Keeping in mind the requirements of the project and efficiency in storage and processing , we decided to proceed with the 5-core dataset which is a contraction of the original dataset to account for a good amount of variation in data.

III. Initial Steps

Google Colab was chosen to be the preferred platform keeping in mind the distributed nature of the project among contributors and the requirements of high RAM and storage capacity. Another alternative of hosting on Azure Cloud service was explored but dismissed due to limited compute power available for the tier accessible and high establishment costs for upgradation.

IV. Exploratory Data Analysis

To collect the data from the remote server, a bash command **wget** was utilised within Google Colab to download the hosted data onto the hosted runtime. The data downlaoded is in **.json** format further compressed into **.gz** format. We unpacked the data using gzip module and loaded it into json format. This data was later used to form a **pandas** dataframe.
Elementary analysis showed that the data had columns with metadata which were not required for our utilisation. Column **overall** and **reviewText** were identified to be the target and input variables.
Further, it was observed that 285 rows of the data did not have values against reviewText. Such rows were

removed from the dataframe. Missing values in other columns were ignored as 'overall' didn't have any missing entries.

For our purpose, it was desired that the words be in their root form as far as possible for ease of processing and analysis. For this, contracted English words such as *couldn't, 've, 'll, mustn't etc.* were expanded to their root form using a RegEx method of replacement. The dictionary data for the contractions was obtained online.

To further decrease the redundancy in words and characters, all sentences were converted to lowercase so that words and their capitalisation don't count as different entities. Extra spaces and punctuation marks were also removed to further concentrate the data and focus more on words.

Vectorization of sentences was required for analysis. For this we considered **tf-idf** and **CountVectorizer**. Out of these two seeing that required for probabilistic features wasn't required as much and keeping in mind computational efficiency we chose CountVectorizer from sklearn library.

The parameters used for CountVectorizer are

- **max_df** : If a word is in 85% of the instances or more then it is discarded from vocabulary on account of being a lot repetitive and not being relevant.
- **min_df** : If a word is in less than 10 instances then it is discarded on account of being too obscure, incorrect or irrelevant.
- **stop_words** : In English certain words like *I, am, is, are, the, an etc.* don't contribute to the sentiment and meaning of the sentence and hence can be ignored.

V. Machine Learning Models

A. Naive Bayes Classifier

A Multinomial Naive Bayes Classifier from the sklearn library was used. Naive Bayes Classifier is found to work well on review classification because the set of words used plays an important role. However it is not completely fool proof as it doesn't take into account the ordering of these words. It also fails to account for literary devices like sarcasm which completely alter the sentiment.

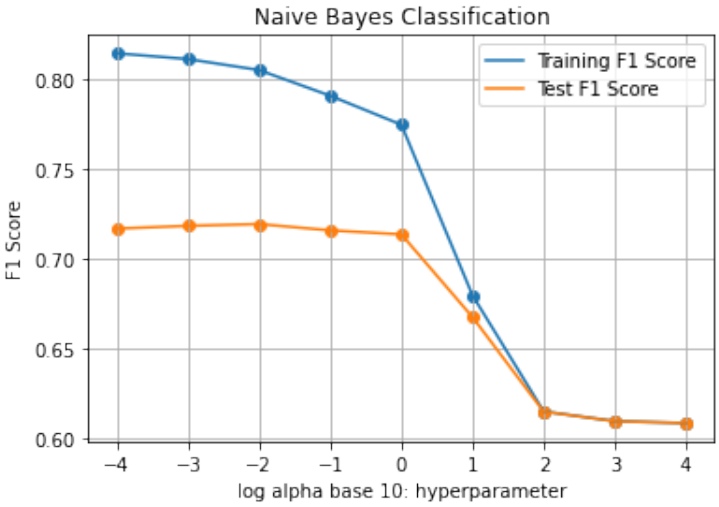


Fig. 1. Results of Grid Search

Grid Search over hyper-parameter alpha was implemented in the range 0.0001 to 10000. The scoring metric chosen for this is F1 Score and the training was done with 5 fold cross validation. Due to the problem being

multi-class classification, weighted F1 score was chosen instead of binary. The best alpha was obtained to be 0.01 and the following Confusion Matrix was found

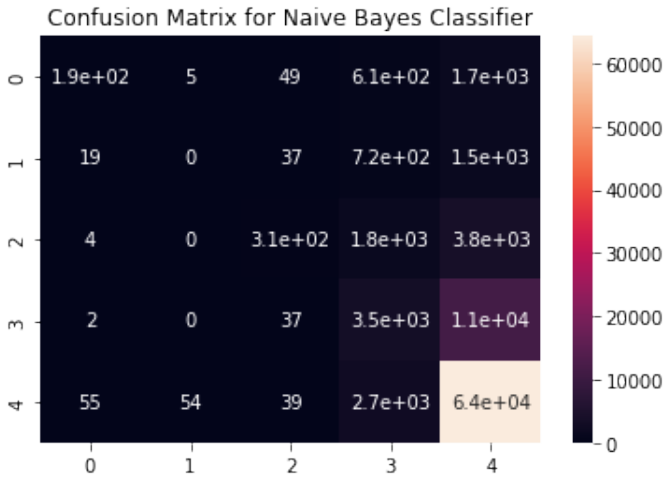


Fig. 2. Multiclass Confusion Matrix

TABLE I
ALPHA VARIATION

Alpha	Mean Test Score	Mean Train Score
0.0001	0.716876	0.814678
0.001	0.718425	0.811464
0.01	0.719419	0.805341
0.1	0.715903	0.791002
1	0.713683	0.774848
10	0.667518	0.679675
100	0.614802	0.614907
1000	0.609447	0.609447
10000	0.608404	0.608404

B. Logistical Regression Classifier

A Logistical Regression Classifier from the sklearn library was used. Motivation behind it is that it is one of the simplest and easiest to train classifier and hence can deployed on large scales easily if it performs well. L1 regularization was chosen for the model so that it can eliminate words that aren't relevant. Computationally it was found to be lot more expensive than Naive Bayes Classifier due to large number of fields in the dataset and hence a smaller sample size was chosen to train and test the model. Overall it showed poorer performance with respect to Naive Bayes.

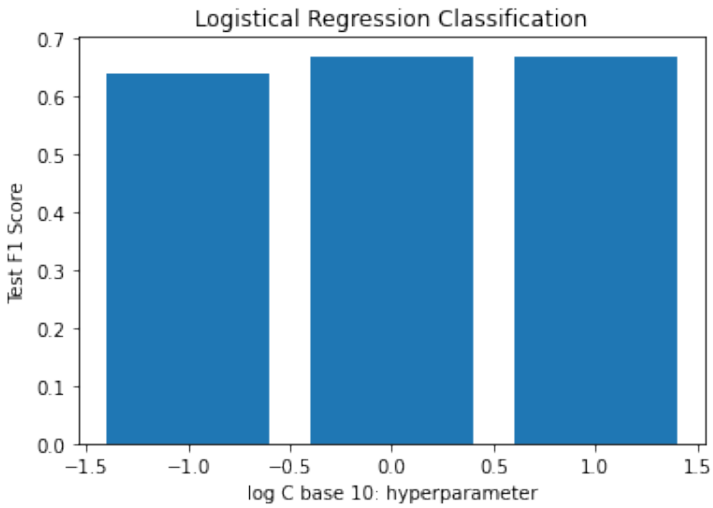


Fig. 3. Results of Grid Search

Grid Search over hyper-parameter C was implemented in the range 0.1 to 10. The scoring metric chosen for this is F1 Score and the training was done with 5 fold cross validation. Due to the problem being multi-class classification, weighted F1 score was chosen instead of binary. The best C was obtained to be 10 and the following Confusion Matrix was found

TABLE II
C VARIATION

C	Mean Test Score
0.1	0.641120
1	0.668824
10	0.669701

A comparison between both the

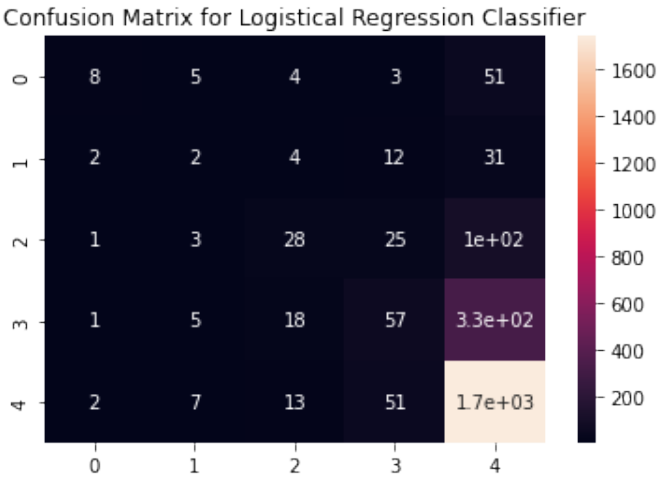


Fig. 4. Multiclass Confusion Matrix

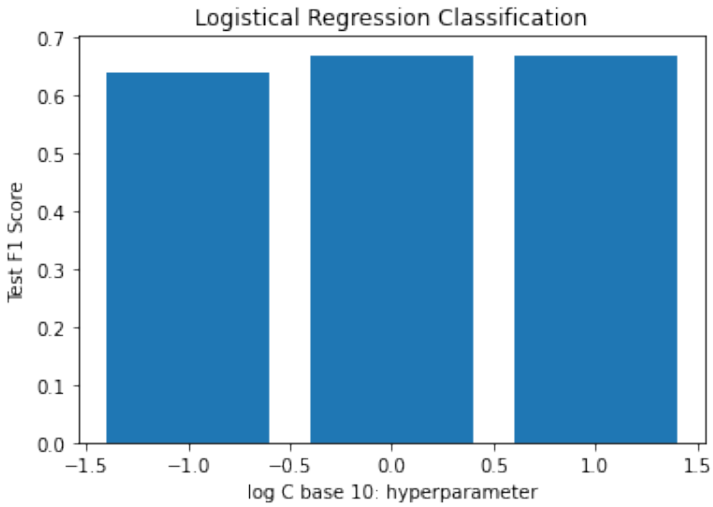
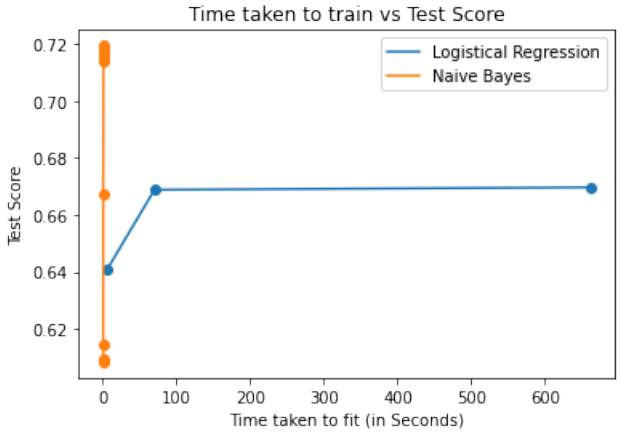


Fig. 5. Results of Grid Search

models may be seen as All iterations



of Naive Bayes take almost the same time to train which is significantly a lot less from the time taken by Logistical Regression. An instance of logistical regression with C=10 took majority of the train time.

VI. Deep Learning Models: LSTM

A RNN batch LSTM architecture has been used in retrieving the long term dependencies on the text sentiment.The model has been designed using keras library from TensorFlow framework. We've used a 70-15-15 train-validation-test split. Motivation behind it was to use the sequential nature of the text and to include long term dependencies in the deep learning model: hence LSTM seemed to be the perfect choice.The architecture used is depicted below: Computationally, the training time was quite significant. The accuracy vs time and the loss vs epochs have been shown below:

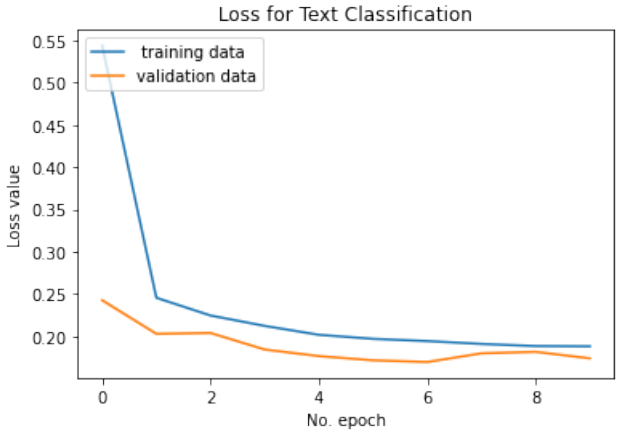


Fig. 6. Loss vs epochs over training and validation data

The accuracy over no of epochs has been depicted as follows:

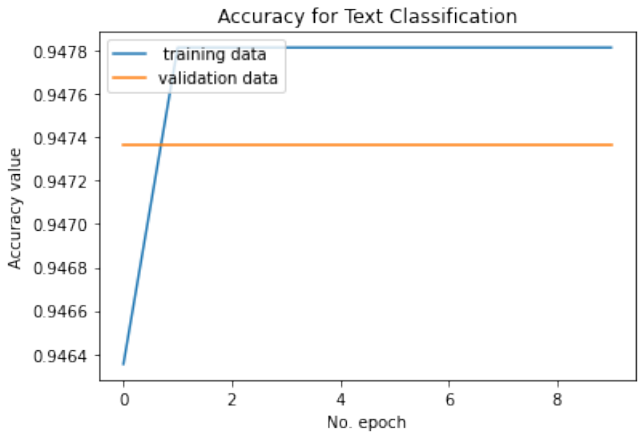


Fig. 7. Accuracy of the model over training period

As this is a classification task, we have used the ROC curve as the performance metric, where we've plotted the true positive and the false positive rates. The graph obtained is as follows:

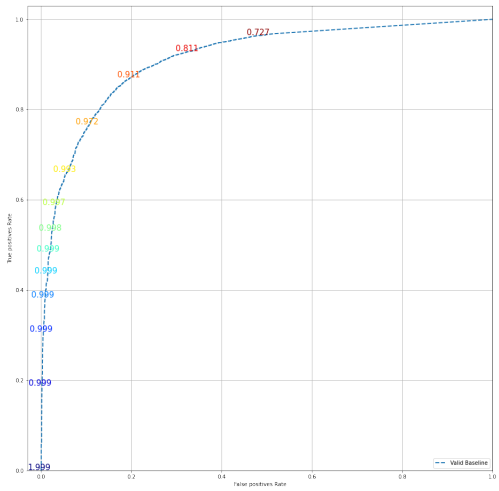


Fig. 8. Rate of false positives and false negatives

REFERENCES

[1] https://nijianmo.github.io/amazon/index.html

[2] https://www.analyticsvidhya.com/blog/2020/04/beginner guide-exploratory-data-analysis-text-data/

[3] https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

[4] https://towardsdatascience.com/basics-of-countvectorizer-e26677900f9c

[5] https://medium.com/mlearning-ai/a-bow-vs-a-tfidf-feature-extractor-a-practical-application-on-a-na%C30%AFve-bayes-classifier-in-python-a68e8fb2248c

[6] https://www.youtube.com/watch?v=DUZn8hMLnbI

[7] https://www.youtube.com/watch?v=DUZn8hMLnbI