

Review Sentiment Analysis and Rating Prediction on DrugReview Dataset

Aloukik Aditya
Student ID 1115290
Dept of Computer Science
Lakehead University
aaditya@lakeheadu.ca

Sarthak Rawat
Student ID 1101124
Dept of Computer Science
Lakehead University
srawat@lakeheadu.ca

INTRODUCTION

With an increasing number of drugs and medicines available today, people have a plethora of options to choose from. A particular medical condition can either have just one medicine or more than five. Although having options may seem like a boon, but this rather makes the decision-making process more complex for customers. One effective, but tiresome, way of overcoming this confusion is to read user given reviews for particular medicines prescribed for specific medical conditions. Such reviews also have medicine-ratings at times which can simply indicate how the response of that medicine was in that user for his/her condition. In cases where such ratings are not given, for instance reading reviews on social media platforms and in surveys, it becomes difficult for a person to go through each and every review. In addition, instead of reading such reviews of both long and short lengths, a person would prefer a short summary or a brief overview of what the reviews says.

This project is aimed at helping users to analyze reviews. Using just the review of a customer, we plan to predict the sentiment of the user for that drug, which will indicate the user response to it – positive, negative, or neutral. Sentiment analysis is a very popular and useful technique that helps reduce human effort in text simplification and understanding. Our first task will be to predict the user sentiment, after which we will use several machine learning classification algorithms and compare their competencies to predict this user sentiment. We also expect to predict the rating for a given medicine, in case its not given, out of 10. We will be comparing the performances

of some more classification models for the task of rating prediction classification.

DATASET

The dataset for this project, *DrugReview Dataset (Druglib.com)* [3] has been selected from the UCI Dataset Repository. It is comprised of a total of seven features – ‘uniqueID’, ‘drugName’ (categorical), ‘condition’ (categorical), ‘review’ (text), ‘rating’ (categorical), ‘date’ (date), ‘usefulCount’ (numerical). The dataset contains 215,063 instances, some of which contain missing values in the ‘condition’ feature. For our problem statement, we will predict the user given rating based on the review. The first task is to analyze the review and create a short summary based on extractive text summarization. Following this summarization, we will predict the user rating, and later we will predict the sentiment expressed by the review. The ground truth for these sentiments will be predicted using an existing text analyzer model. Therefore, we will add a new feature to the dataset named *sentiment*.

LITERATURE REVIEW

The problem of sentiment analysis is significant in the case of text analysis and understanding. The objective of performing sentiment analysis on text is to indicate the sentiment implied by that text – positive, negative or neutral. There has been a previous study [1], where the authors implemented the concept of transfer learning over two separate, but related, data-sets [2], [3]. The work done in [1] tests the performance of classifiers, i.e. logistic regression models, for cross-domain

and cross-data sentiment analysis – cross-domain analysis emphasized on training the model for a particular condition, and tested the performance for rating prediction over other subsets of conditions; and, cross-data analysis focused on the transferability of the trained models over different data-sources (transfer learning). The performance of transfer learning is compared in both cases – training on *Drugs.com* [2] and testing on *Druglib.com* [3], and training on *Druglib.com* [3] and testing on *Drugs.com* [2], achieving an accuracy of 75.29% in the former case, and 70.06% in the latter. It is indicated that this performance is bound to increase if more sophisticated features and applying more powerful machine learning models are used, for instance deep learning models. In this project, we have focussed on comparing the performances of different classification models for the task of rating and sentiment prediction.

DATA PROCESSING

A. Data Cleaning

On analysis of the dataset, it was found that many reviews contain garbage characters, such as '&' and '#039' for special symbols. In addition to these, users have the tendency to add multiple characters in place of just one, for instance, 'This is awesome!!!!!!' and 'I hate this... DISGUSTING!!', etc. The first step for data processing would be replacing these garbage characters with the appropriate symbols and multiple periods and exclamation points with a single period and exclamation point respectively. The data cleaning phase also checks for the presence of missing values in instances. As mentioned, the 'condition' feature in our data contains missing values, and the percentage of missing values was found to be less than 1%, hence we removed these record. An example is shown in *figure-1*

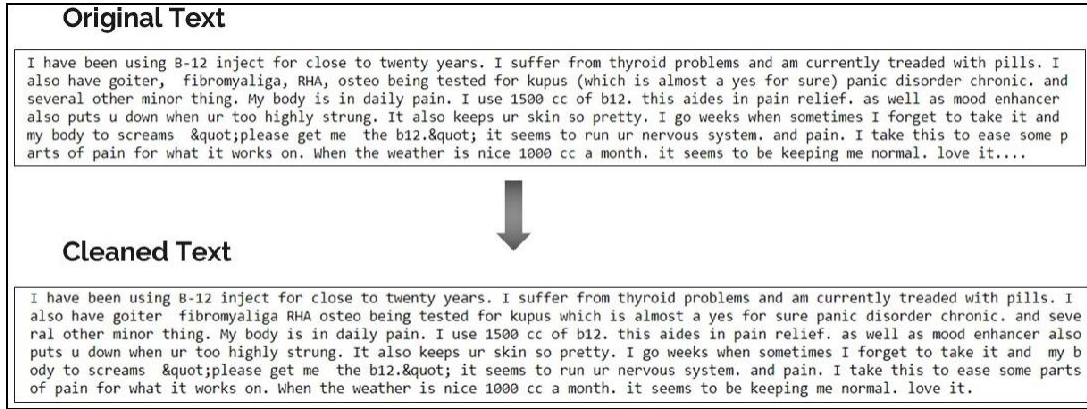


Fig 1: Data Cleaning

B. Text Summarization

After obtaining cleaned versions of reviews, we move forward to summarize them. Given the variable length of reviews – as short as two words, and as long as 15 sentences – we create a 'summary' feature for each instance in the data. This is done by examining the 'term-frequency' in each review, and assigning weights to every word in a given review. The weight assignment is done by checking the term with the maximum occurrences, and dividing the frequencies of all words by this maximum value. Having the term frequencies for a review, we can calculate the weights for each sentence in the review.

This way, each sentence in the review has a specific weight assigned to it. The next step is to get the best representative sentences for the review – this is the 'extractive summarization' method. For this we set a threshold value for these weights (at 1.0), so sentences with weights higher than the threshold will be chosen for the summary. Now that we have an extracted summarized version for each review, we can move ahead to vectorizing these summaries due to variable lengths. The figures - 2, 3, 4 show how text in each review is broken down and summarized.

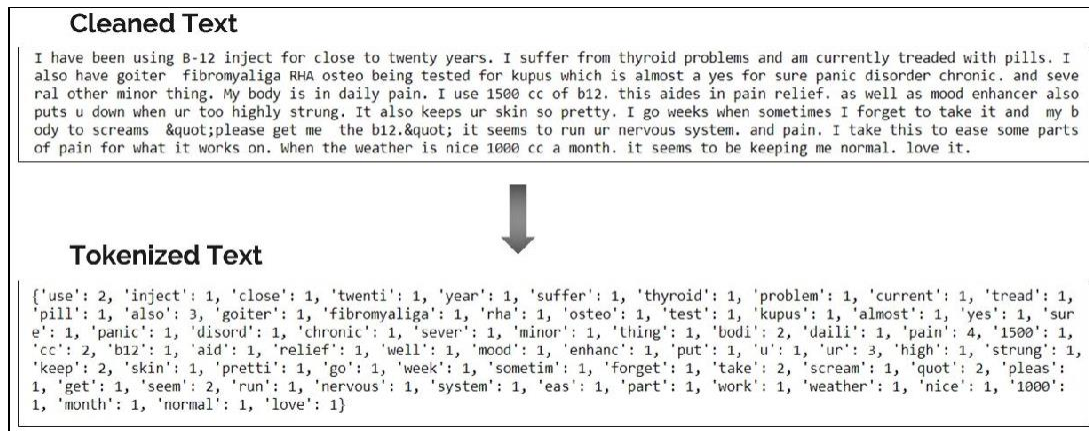


Fig 2: Review Tokenization – breaking down each review into unique words

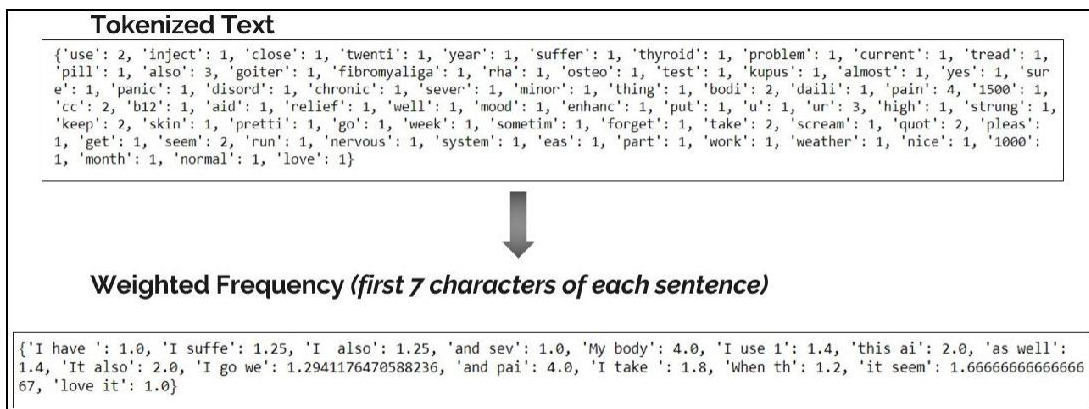


Fig 3: Weight Assignment to each sentence in a review

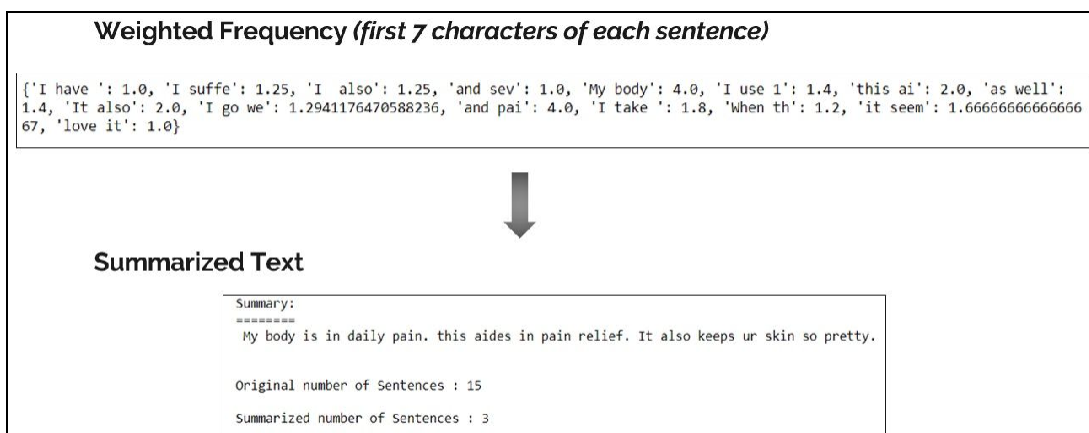


Fig 4: Final summarized review

C. Vectorization

This phase of preprocessing of data involves transformation of reviews to fixed length vectors for the purpose of feeding as input to the classification models for analysis. For review vectorization, we use the *TF-IDF Vectorizer* – ‘term-frequency inverse-document-frequency’. It takes as input the set of summarized reviews, calculates the number of occurrences (term frequency) of each unique word in the entire set of reviews (inverse-document frequency) and assigns a weight to it. Therefore, the vectorized form of the reviews is of fixed length, equal to the number of unique words in the dataset.

One feature to notice here is that we are using the summarized version of each review. If we use the original review for vectorization, the number of ‘features’ in the vector is 49,853, whereas using summaries for vectorization, this size goes down to 29,209 features. Now that we have fixed length input for the classifiers, we can finally get to the sentiment analysis part. *Figure-5* shows the dimensions of each review after vectorization for two cases – (a) original review; and, (b) summarized review.

```
print("Test Shape : ",X_test.shape)
print("Train Shape : ",X_train.shape)
print("We can see that every review in\
test and train Dataset have equal number of features now")

Test Shape : (53471, 49853)
Train Shape : (160398, 49853)
We can see that every review in test and train Dataset have equal number of features now

print("Test Shape : ",X_test_summary.shape)
print("Train Shape : ",X_train_summary.shape)
print("We can see that every review in\
test and train Dataset have equal number of features now")

Test Shape : (53471, 29209)
Train Shape : (160398, 29209)
We can see that every review in test and train Dataset have equal number of features now
```

Fig 5: Dimensions for vectorized form of reviews in (a) original review; (b) summarized review

D. Sentiment Analysis

The most important task for this project is preparing data to be used for the purpose of sentiment prediction. At first, we found some correlation between the ‘review’ feature and the ‘rating’, for which we categorized the ratings into three classes – lower ratings imply a negative sentiment, and higher ratings a positive one. However, this didn’t solve the purpose of analyzing a review for sentiment analysis. Therefore, we utilize an existing *python* library called ‘sentiment-intensity-analyzer (VADER)’ [4], which is a part of the natural language processing toolkit in *python*, ‘*nltk*’. Using this library, we can analyze the expressed sentiment in a given user-review within

[-1,1], where (-1) indicates a negative sentiment, and (+1) indicates a positive one. A threshold value was set for distinguishing a ‘neutral’ sentiment from the others; this threshold was set at ± 0.5 – values above (+0.5) are set as positive, and those lower than (-0.5) are set as negative. Reviews within this range of [-0.5,0.5] are predicted as neutral. Using the *vader-analyzer*, we create a new feature for our dataset called ‘sentiment’, which is a categorical variable with three possible values – positive, negative and neutral. This feature will be the target variable for sentiment prediction for the remainder of the project.

	uniqueID	drugName	condition	review	rating	date	usefulCount
0	206461	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combinati...	9	20-May-12	27
1	95260	Guanfacine	ADHD	"My son is halfway through his fourth week of ...	8	27-Apr-10	192
2	92703	Lybrel	Birth Control	"I used to take another oral contraceptive, wh...	5	14-Dec-09	17

Fig 6(a): Original Dataset

	uniqueID	drugName	condition	review	rating	date	usefulCount	summarize	sentiment
0	206461	Valsartan	Left Ventricular Dysfunction	'It has no side effect I take it in combinati...	9	20-May-12	27	'It has no side effect I take it in combinati...	neutral
1	95260	Guanfacine	ADHD	'My son is halfway through his fourth week of ...	8	27-Apr-10	192	See how he did at school and with getting up...	positive
2	92703	Lybrel	Birth Control	'I used to take another oral contraceptive wh...	5	14-Dec-09	17	And the period lasted for two weeks. When tak...	positive

Fig 6(b): Dataset after adding *summarize* and *sentiment* features

METHODS & TOOLS

The dataset is processed as a *pandas dataframe*, a library in python. Since the data contains missing values in the 'condition' feature, we find the percentage of records with missing data to be minuscule, and remove such records using the *drop_na_values* function of 'pandas' library. By utilizing the natural language processing library in python, 'nltk', we summarize the text in each given review by first cleaning the text using regular expressions, then removing unnecessary terms (stopwords), and summarizing each review. The 'nltk' library provides a range of other helper functions, such as the 'vader sentiment intensity analyzer' which we used to predict the sentiment of a summarized review.

For the classification models used in this project, the 'sklearn' library is used to invoke models such as *naïve bayes*, *random forests*. This library also provides us with various other helper functions, such as feature extraction using TF-IDF Vectorizer and preprocessing data using one-hot encoding for the neural network model. The neural network model is built using the 'keras' library. It consists of different layers, namely 'dense', 'dropout', and 'activation', which are used in sequence to create the neural network.

CLASSIFICATION MODELS

Our aim in this project is to analyze how different classification models perform with textual data (reviews) for rating prediction as well as sentiment prediction. We employ a range of classifiers for rating and sentiment prediction. These models are used after the process of data cleaning, summarization and vectorization.

A. Rating Prediction

The task in this part is to predict the user given rating for a specific medicine or drug (out of 10) using only the 'summarized' review feature. The feature has approximately

29k features after vectorization, and we take these as input, and predict one of ten classes (ratings). For the purpose of comparison, the performance has been measured in two scenarios – one where the model is trained using vectorized forms of the original review, and the other using the summarized review. For rating prediction, we use the following classification models:

- Neural Network
- Random Forest
- XGBoost Classifier

B. Sentiment Prediction

In this part we use classification models to predict the sentiment of a user's review. The models are trained on the 'sentiment' feature, which is procured by using the 'vader analyzer' on each of the summarized review. The predictions are done for each summarized review and can have one of three possible values – positive, negative and neutral. The classification models employed for this task are:

- Neural Network
- Random Forest
- Naïve Bayes
- XGBoost Classifier

PERFORMANCE EVALUATION

A. Rating Prediction

In this project, the first classification model we use are Random Forests. This is done while testing for the best combination of hyperparameters, (a) number of features 'm' in the random subset of vectorized features 'M'; and (b) the number of trees in the Random Forest. The results are shown in *table-1*, where we test the performance over two conditions – one with the original reviews, and the other with the summarized reviews, which are also represented by *figures-7 and 8*.

Accuracy (%)		Number of Trees 'n'					
		Original Review			Summarized Review		
		20	50	100	20	50	100
Value of 'm'	$\log_2(M)$	0.7275	0.7372	0.7396	0.7207	0.7310	0.7326
	\sqrt{M}	0.7365	0.7446	0.7443	0.7263	0.7322	0.7344
	$2 \times \sqrt{M}$	0.7411	0.7485	0.7487	0.7278	0.7335	0.7351

Table 1: Random Forest Performance Comparison

The next model which we employed for rating prediction is the neural network model for classification. The network architecture comprises of four layers – an input layer, two hidden layers, and an output layer. The input is the vectorized summarized review – original with ~49k features and summarized ~29k features; the two hidden layers consist of 500 and 300 nodes respectively; and, the output layer containing 10 nodes for each rating. The training accuracy for neural network using the original features is 94.65% and testing accuracy is 74.01%. On the other hand, if we used the summarized review for rating prediction, the training accuracy goes down to 88.78% and that of testing drops to 70.50%. *Figures-9* shows the graphical representations for accuracy and loss on the training and testing sets when the original review is used for prediction; and *figure-10* shows the same when the summarized review is used for predicting the rating.

The XGBoost classifier [5] has proved to be of great significance for classification tasks due to the adaptive nature of its algorithm. Due to this reason, we test its performance for rating prediction on our data. By tuning a single hyperparameter 'max_depth', we achieve the highest accuracy of 56.01% at *depth*=20. *Figure-11* shows the hyperparameter tuning for this model and *table-2* displays the accuracies at different hyperparameter values..

B. Sentiment Prediction

The second task of our project Syntactic, Semantic and Sentiment

Analysis: The Joint Effect on Automated Essay Evaluation. sentiment analysis, makes use of four classification models and compares their performances. The first of these models is a simple neural network, with a similar architecture as the previous one – one input layer, two hidden layers, and an output layer. Since the target variable 'sentiment' is

constructed using the summarized review, we don't require comparisons for prediction of sentiment for original and summarized reviews. By training the neural network, we achieve an accuracy of 98.07% and a testing accuracy of 90.28% is obtained for sentiment prediction. *Figure-12* illustrates the performance of this model on the dataset.

We make use of the XGBoost classifier for sentiment prediction as well, and with hyperparameter tuning of the 'max_depth' feature, we can attain an accuracy of 83.95%. *Table-3* displays the performance of XGBoost classifier for the given set of hyperparameters and *figure-13* shows the hyperparameter tuning performance for the model.

For this task, we make use of the Random Forest classifier again, but this time for sentiment prediction. We tune two separate hyperparameters for the random forest model using a grid-search approach – testing for every possible combination of the two. *Table-4* gives an overview of the different values for these hyperparameters and their performances. The best possible combination with the highest performance is when the number of trees is 100 and the value of 'm' is kept at '2*sqrt(M)'. The best accuracy achieved using this configuration of the random forest model is 87.30% approximately. *Figure-14* shows the hyperparameter tuning for this model.

The next model we use for sentiment prediction is the Naïve Bayes classifier. For this method, we tune two hyperparameters, namely *fit_prior* and *alpha*. Since this is the 'm-estimate' version of Naïve Bayes, 'alpha' decides the value of 'm' and 'fit_prior' decides whether to use the actual or uniform class distribution. Using this classifier didn't prove to be beneficial in our case, as we could achieve an accuracy as high as 66.37% after setting 'alpha' to 0.01 and 'fit_prior' as

True. Figure-15 can be used to show the results of hyperparameter tuning in case of the Naïve Bayes classifier. Given the set of best hyperparameters for each model, the best accuracy is achieved using a neural network for both tasks of rating prediction as well as sentiment prediction. The highest

performance for rating prediction is 74.01% using the original reviews, and 70.50% by using the summarized reviews. An accuracy of 90.28% is achieved for the task of sentiment prediction.

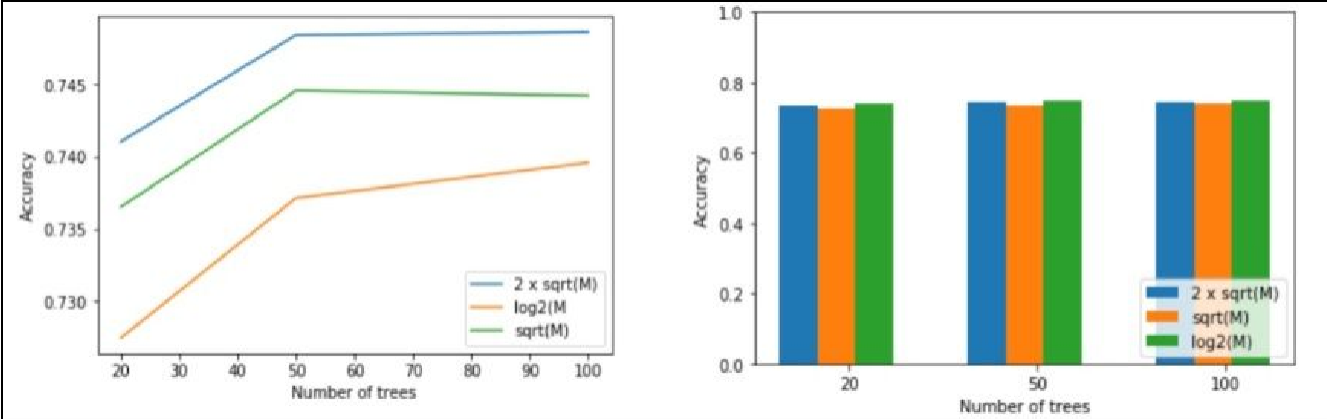


Fig 7: Random Forest Performance for Rating Prediction (original reviews)

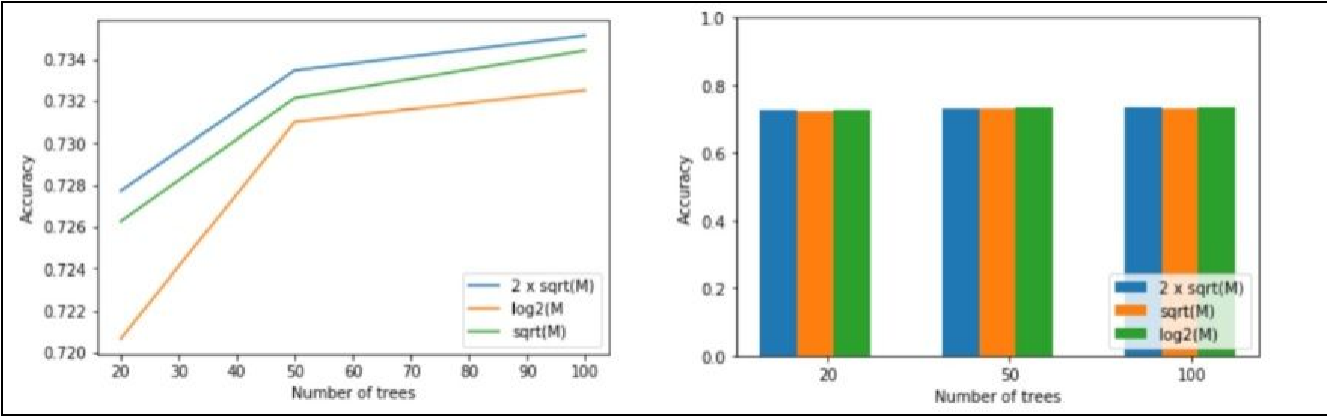


Fig 8: Random Forest Performance for Rating Prediction (summarized reviews)

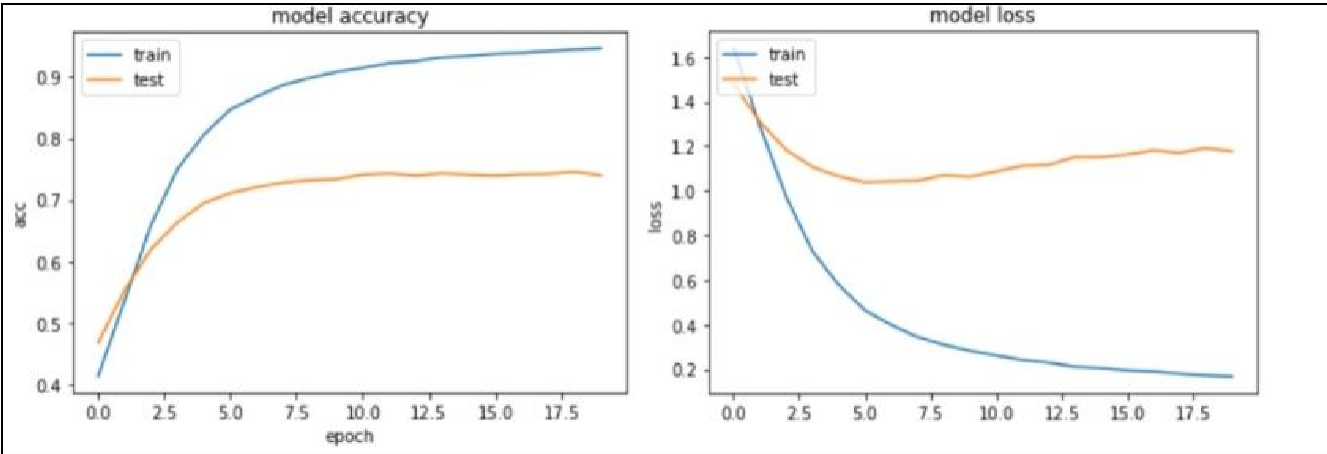


Fig 9: Neural Network Performance for Rating Prediction (original reviews)

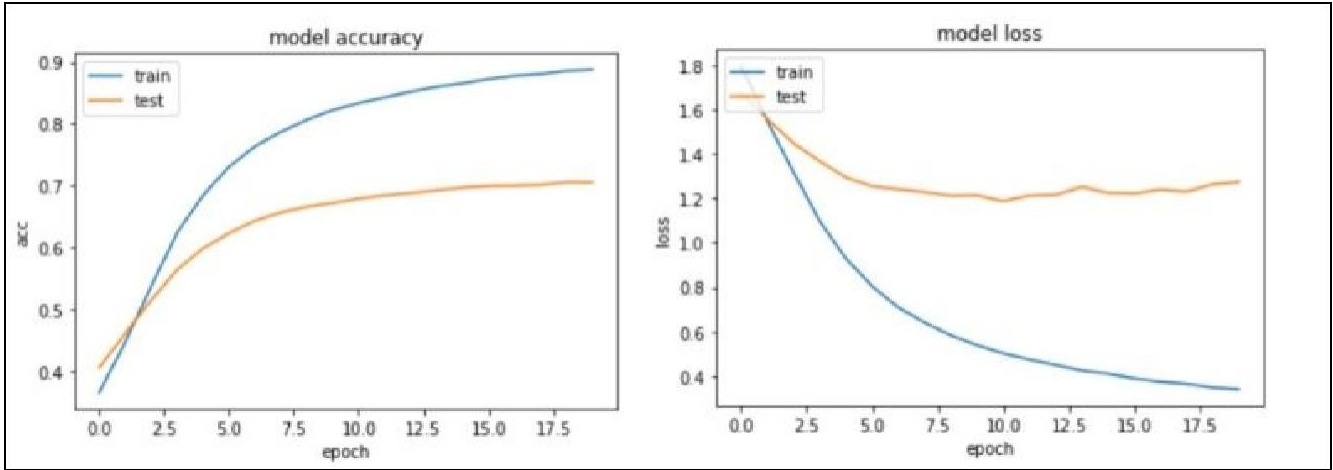


Fig 10: Neural Network Performance for Rating Prediction (summarized reviews)

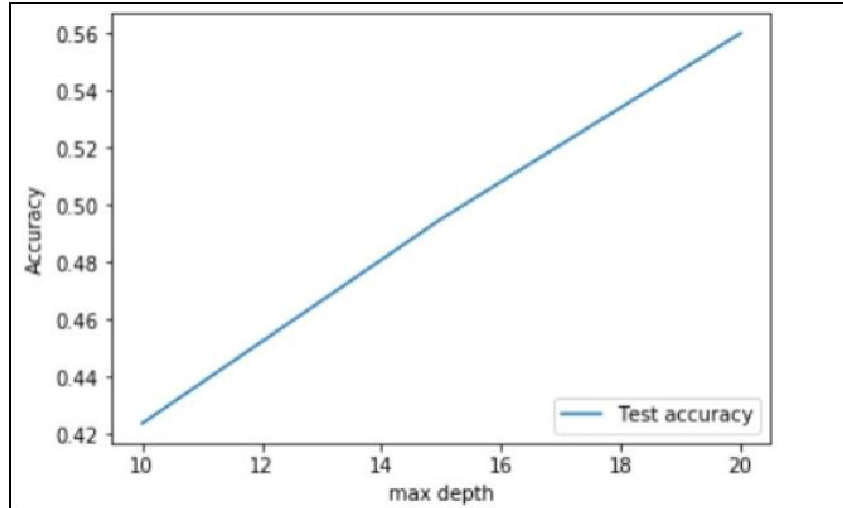


Fig 11: XGBoost Classifier Performance for Rating Prediction

Max-Depth (n)	Accuracy (%)
10	42.37
15	49.52
20	<i>56.01</i>

Table 2: XGBoost Classifier Performance for Rating Prediction

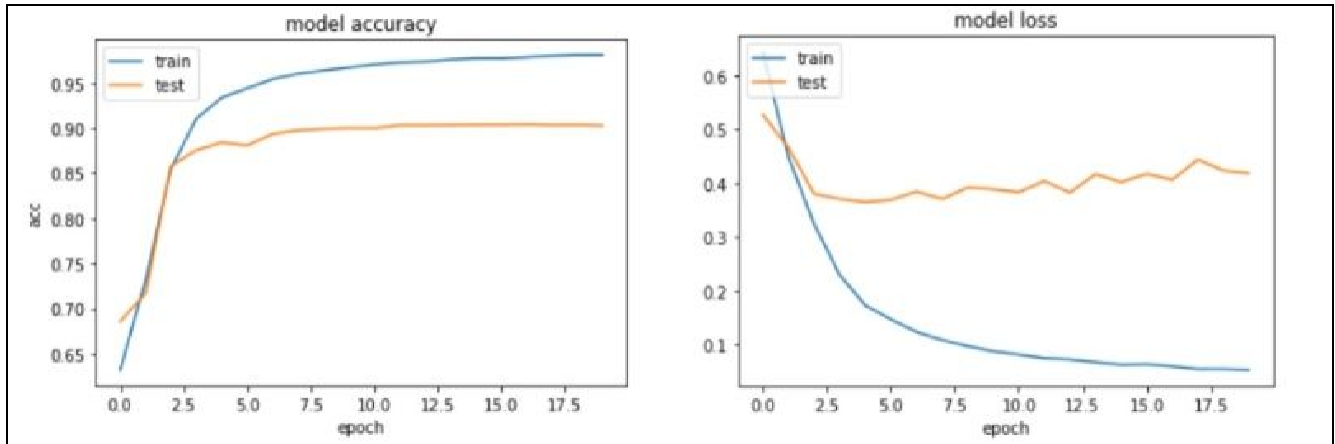


Fig 12: Neural Network Performance for Sentiment Prediction (summarized reviews)

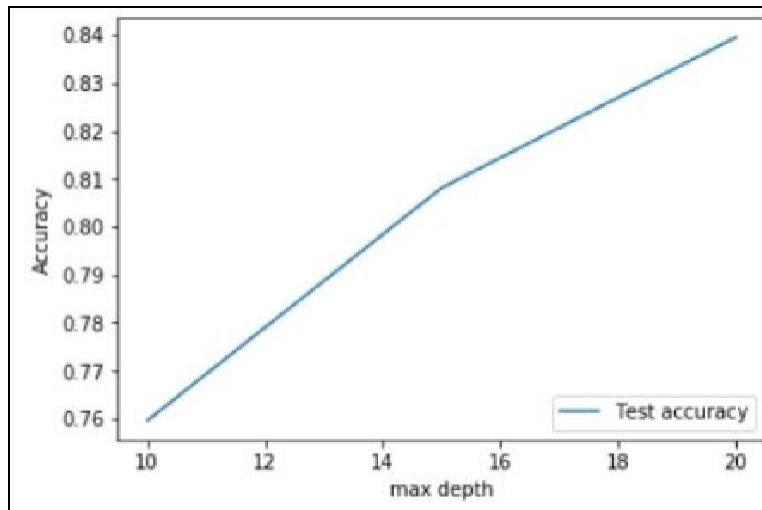


Fig 13: XGBoost Classifier Performance for Sentiment Prediction

Max-Depth (n)	Accuracy (%)
10	75.96
15	80.81
20	83.95

Table 3: XGBoost Classifier Performance for Sentiment Prediction

Accuracy (%)		Number of Trees 'n'		
		20	50	100
Value of 'm'	$\log_2(M)$	0.8415	0.8431	0.8396
	\sqrt{M}	0.8544	0.8600	0.8613
	$2 \times \sqrt{M}$	0.8640	0.8710	0.8730

Table 4: Random Forest Performance for Sentiment Prediction

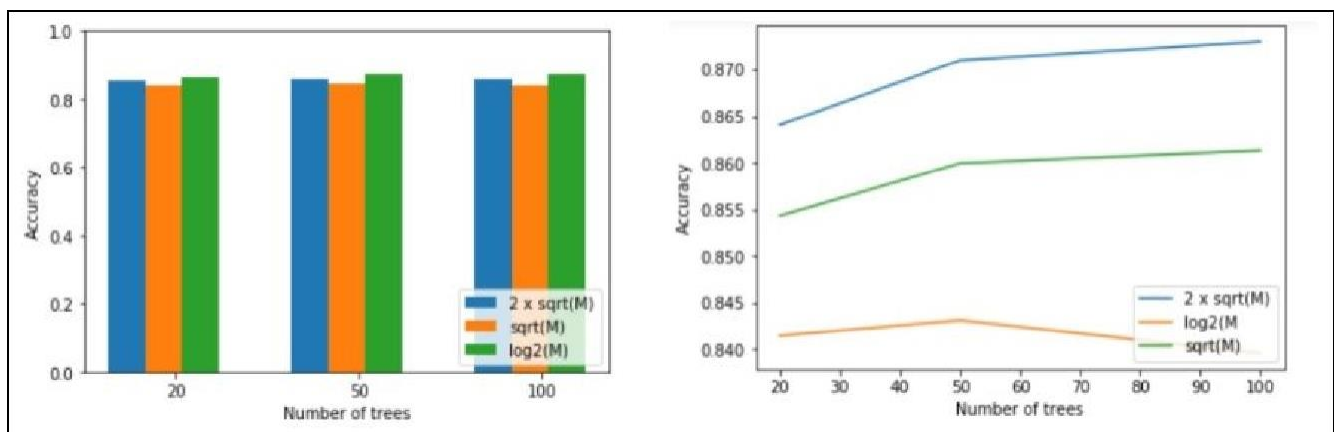


Fig 14: Random Forest Performance for Sentiment Prediction

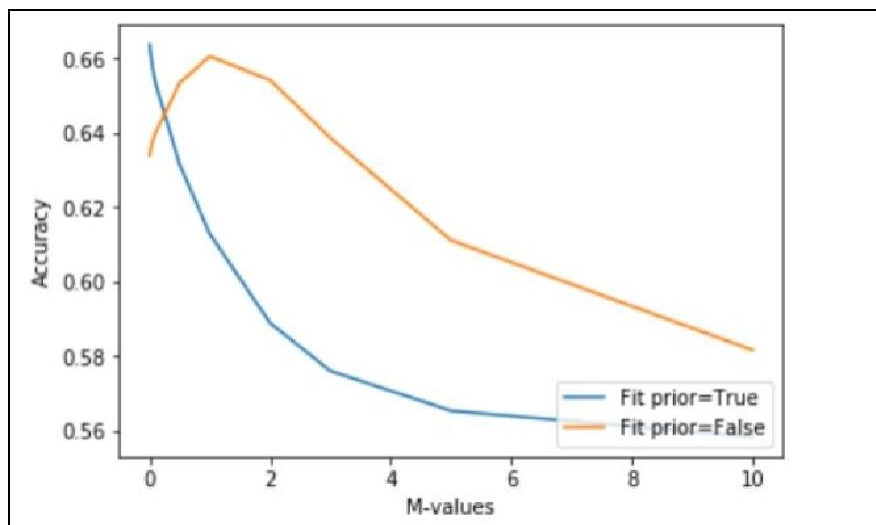


Fig 15: Naïve Bayes Classifier Performance for Sentiment Prediction

EXPLORATORY DATA ANALYSIS

The data collected from UCI repository contains a number of missing values in the 'condition' feature. The percentage of records with missing values was found to be approximately 0.55% (extremely low), hence we removed such instances (899 records) from the dataset. Also, as mentioned above, the 'data cleaning' phase is a crucial part of our data processing since many reviews contain garbage characters and multiple punctuations, which we remove using regular expressions.

For rating prediction, the target variable - 'rating' feature - is a categorical variable with 10 possible values. However, the distribution of these ratings is non-uniform, thereby inducing the problem of class imbalance. This distribution can be observed in *figure-16 (a)*. Furthermore, for sentiment prediction, we create a new feature called 'sentiment' based on

each user's review, and label it as positive, negative or neutral. The class distribution for this label can also be viewed in *figure-16 (b)*. The distribution of the top 10 most popular and least popular drugs can also be observed in *figure-17* and *figure-18* respectively.

A 'word-cloud' is a diagram which displays the concentration of most frequently occurring terms differing by sizes. In other words, terms appearing larger in a word-cloud are the ones which appear the most frequently (in all reviews combined). This can be explained with the help of different word-clouds constructed on the 'reviews' based on different conditions - *figure-19 (a)* for the most occurrences in the review features, *figure-19 (b)* for top 100 most 'useful' reviews, and *figure-19 (c)* for top 100 least 'useful' reviews.

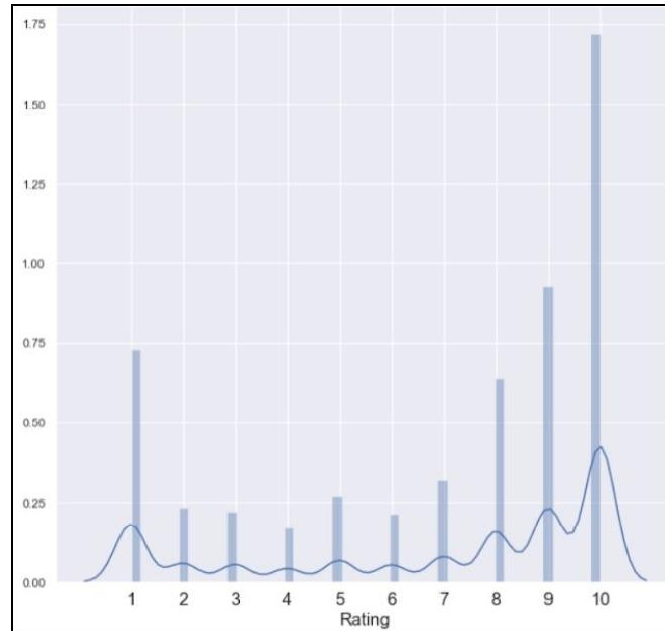
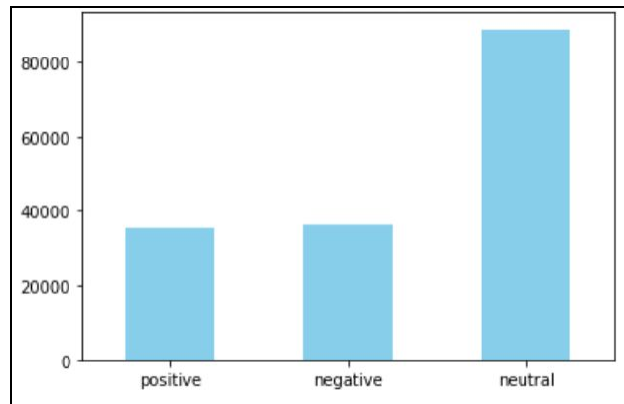


Fig 16: (a) Rating-feature class distribution (above); (b) Sentiment-feature class distribution (below)



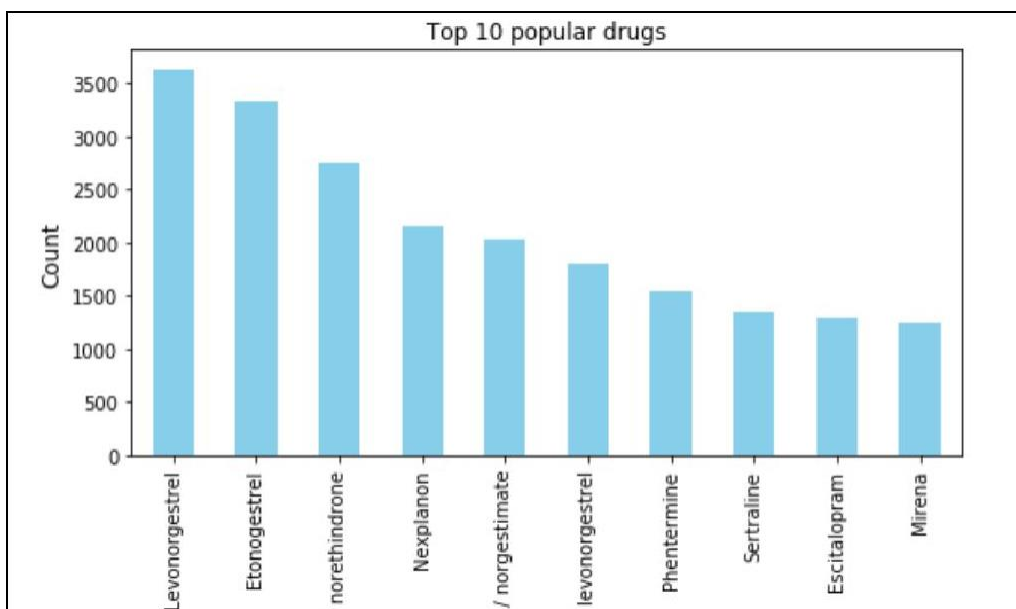


Fig 17: Top 10 most popular drugs ('drugName'-feature)

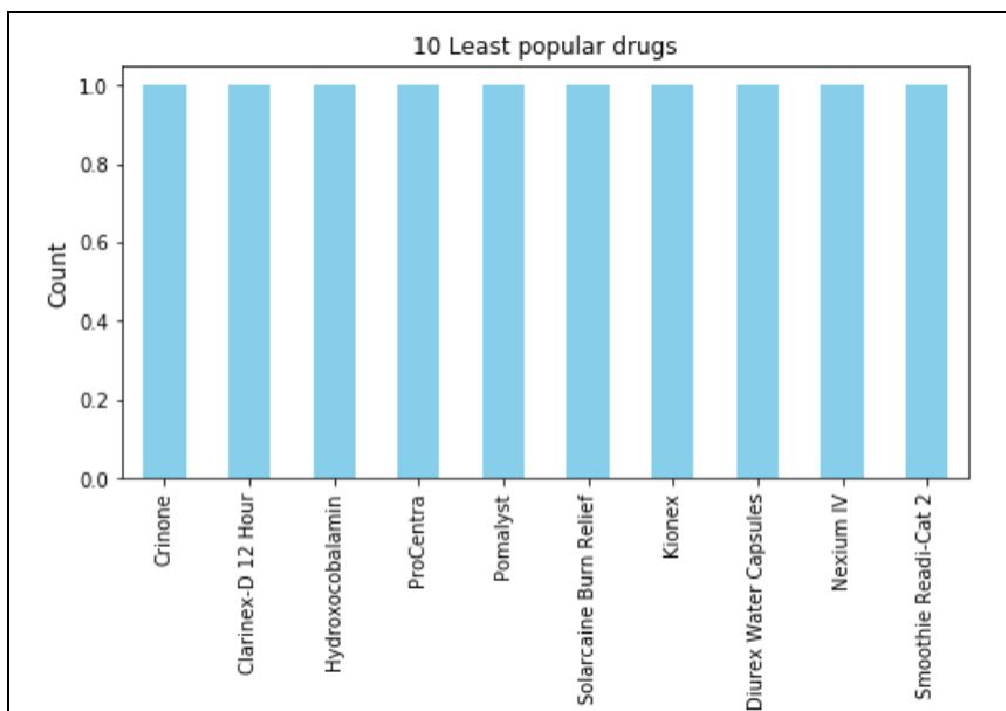


Fig 18: Top 10 least popular drugs ('drugName'-feature)

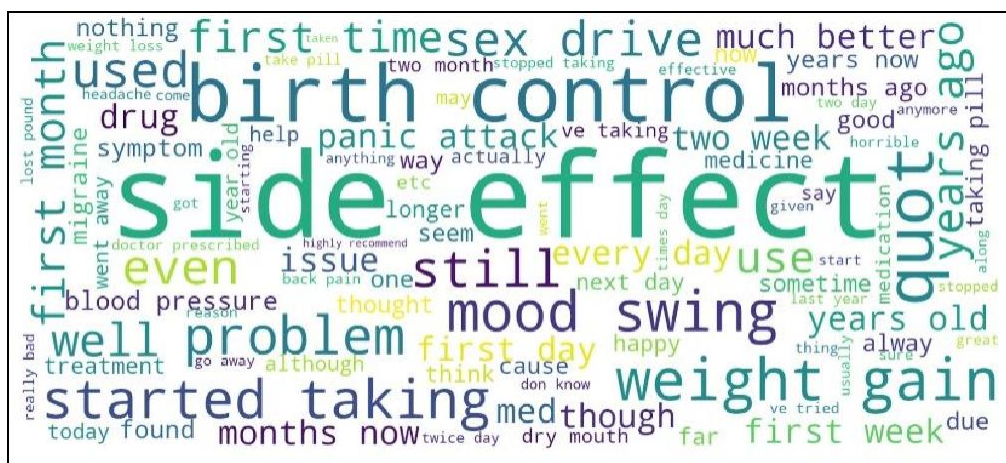


Fig 19(a): Word-cloud for all reviews

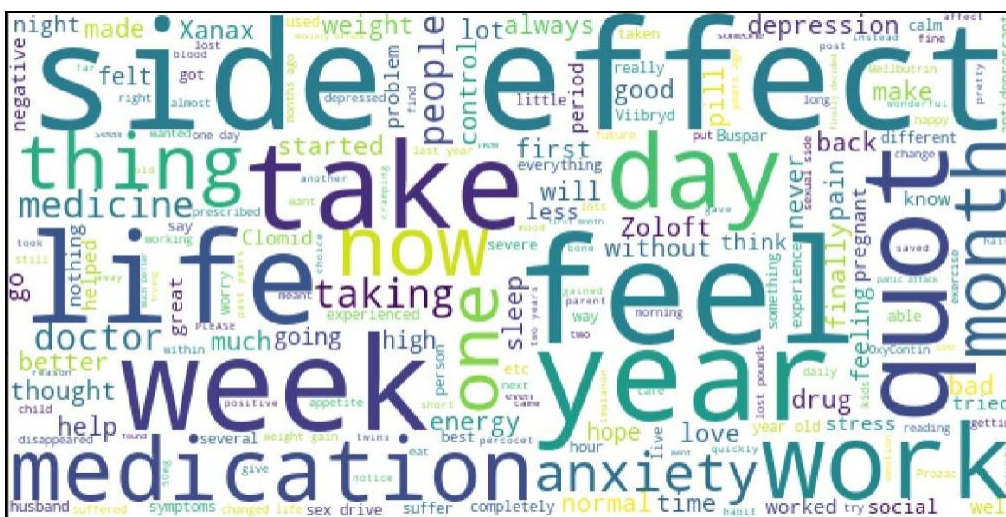


Fig 19(b): Word-cloud for Top 100 ‘useful’ reviews



Fig 19(c): Word-cloud for Lowest 100 'useful' reviews

DISCUSSION

The first trend that we observe in the performances of different classifiers is that they work better on the original reviews than on the summarized versions. This may be due to the inefficiency of ‘extractive summarization’ method used in this project. Another method of text summarization can be ‘abstractive’, where the machine learns the hidden meaning of a review and paraphrases the review to get an abstract meaning of it, which can be our summarized version of the review. The best performance for ‘rating prediction’ is achieved by the neural network model, i.e. 74.01% on the test-set, on the original set of reviews and not the summarized ones. In addition to this, the neural network model performs the best among all the models for ‘sentiment prediction’ as well, attaining a testing accuracy of 90.28% on the summarized reviews.

It can be observed that the basic neural network model with minimal hyperparameter tuning can produce the best results among the models used. Given the background of text analysis in this project, a more sophisticated LSTM (long short-term memory) model can be used, which works on a sequence-to-sequence basis. This should definitely give a much better performance than a simple neural network model for text analysis.

Also, due to limited computational resources and time constraints, we could not perform an exhaustive grid-search for hyperparameter tuning, and could not include other







classification models that were tried, such as the k-nearest neighbors (kNN) and support vector machines (SVM) models. However, an LSTM model should outperform these as well, with appropriate hyperparameter tuning.

REFERENCES

- [1] F. Gräßer, S. Kallumadi, H. Malberg, and S. Zaunseder, “Aspect-based sentiment analysis of drug reviews applying cross-domain and cross-data learning,” in *Proceedings of the 2018 International Conference on Digital Health*, pp. 121-125, ACM, 2018.
- [2] [Online], “(Drugs.com) Drug Review Dataset.” <https://archive.ics.uci.edu/ml/datasets/Drug+Review+Dataset+%28Drugs.com%29>.
- [3] [Online], “(Druglib.com) Drug Review Dataset.” <https://archive.ics.uci.edu/ml/datasets/Drug+Review+Dataset+%28Druglib.com%29>.
- [4] Hutto, Clayton J., and Eric Gilbert, “Vader: A parsimonious rule-based model for sentiment analysis of social media text.” *Eighth international AAAI conference on weblogs and social media*, 2014.
- [5] T. Chen, and Carlos Guestrin, “Xgboost: A scalable tree boosting system.” *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016.

APPENDIX

- The following files should be present in source code folder :

Name	Date modified	Type	Size
 data	05-12-2019 03:52	File folder	
 exploratory_data_analysis.ipynb	05-12-2019 05:40	IPYNB File	166 KB
 requirements.txt	05-12-2019 03:39	Text Document	1 KB
 review_preprocess.py	01-12-2019 15:45	JetBrains PyCharm	2 KB
 source_code_done.ipynb	05-12-2019 03:47	IPYNB File	395 KB
 summarize_review.py	30-11-2019 18:10	JetBrains PyCharm	4 KB

STEP I:

In command-prompt/ terminal open the source code folder and install the requirements of the code using the following command:

pip install -r requirements.txt

```
~$ pip install -r requirements.txt
```

In case the above command gives an error → install the dependencies using “pip” with the following commands:

- **pip install pandas**
- **pip install numpy**

And similarly for the following dependencies : *pandas, numpy, nltk, xgboost, scikit-learn, matplotlib, keras, tensorflow, protobuf, np_utils, wordcloud*

Note : In addition to the above dependencies we need to install **wordcloud** for our ‘Exploratory_data_analysis.ipynb’ notebook. This is required to plot the **wordcloud** from our reviews.

To install wordcloud library run the following code :

>> pip install wordcloud

STEP II:

Run the following command in your python console to download the required data files of nltk for text processing.

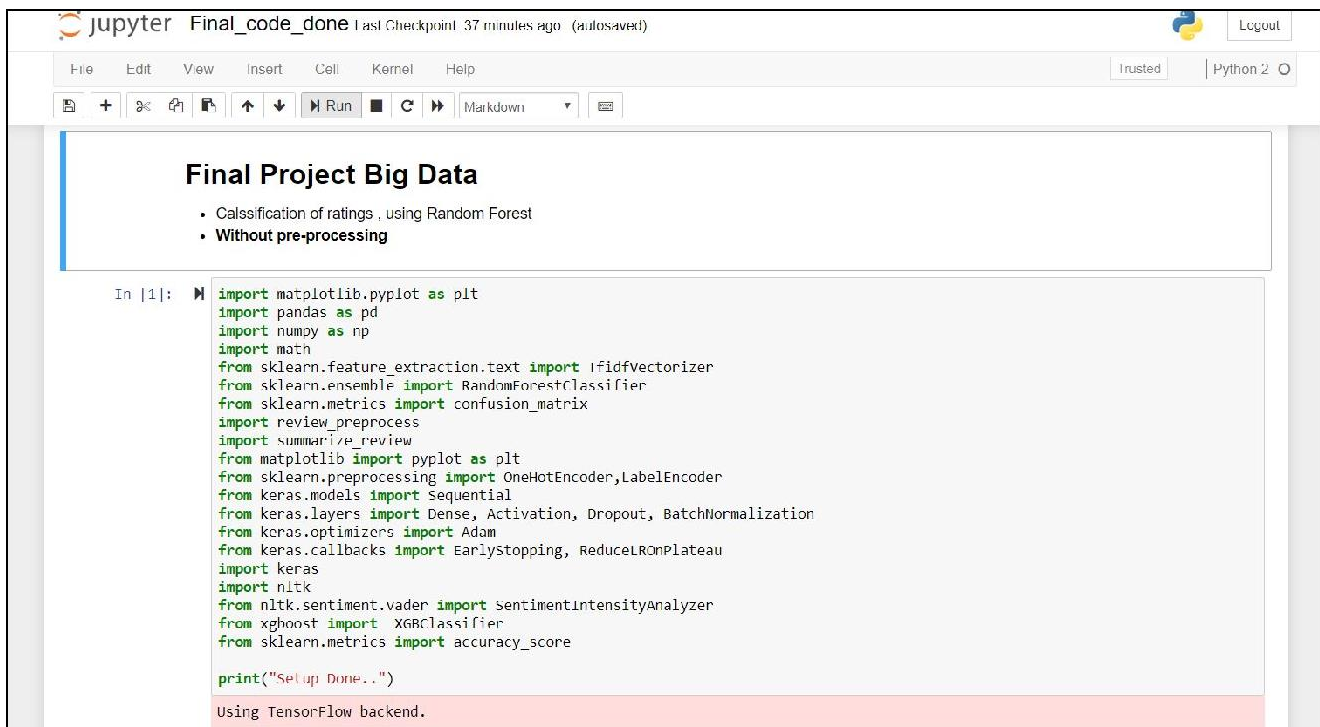
```
>>import nltk
>>nltk.download('stopwords')
>>nltk.download('punkt')
>>nltk.download('vader_lexicon')
```

The output should be like below after downloading:

```
sarthakrawat7295@instance-2:~$ python
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> nltk.download('vader_lexicon')
[nltk_data] Downloading package vader_lexicon to
[nltk_data] /home/sarthakrawat7295/nltk_data...
True
>>>
```

STEP III:

- Start jupyter notebook server and open "source_code.ipynb"
- The file should look like below.
- Run the cells accordingly in sequence.



NOTE:

In case you encounter a **LookupError** like the one shown below:

```
-----
LookupError                                Traceback (most recent call last)
<ipython-input-6-3b7e7bd3b737> in <module>()
----> 1 get_ipython().run_cell_magic(u'time', u'', u'\nimport nltk\nfrom nltk.sen
zer\n\nsid = SentimentIntensityAnalyzer()\n# Create list (cast to array) of compo
sentiments = []\nfor i in df_train.summarize:\n    if sid.polarity_scores(i).get(
append(\'negative\')\n    elif sid.polarity_scores(i).get(\'compound\') < 0.5 :\n
elif sid.polarity_scores(i).get(\'compound\') <= 1 :\n        sentiments.append(
in df_test.summarize:\n    if sid.polarity_scores(i).get(\'compound\') < -0.5 :\n
\')\n    elif sid.polarity_scores(i).get(\'compound\') < 0.5 :\n        sentiment
larity_scores(i).get(\'compound\') <= 1 :\n        sentiments_test.append(\'posit
ents_test\)\nsentiments = np.array(sentiments)\n\nndf_train["sentiment"] = sentimen
t')

LookupError:
*****;
Resource vader_lexicon not found.
Please use the NLTK Downloader to obtain the resource:

>>> import nltk
>>> nltk.download('vader_lexicon')

For more information see: https://www.nltk.org/data.html
```

It is probably because **STEP II** was not executed properly. In that case just follow the instructions in **STEP II** again:

```
>> import nltk
>> nltk.download('<required file name>')
```