DS 5220: Supervised Machine Learning and Learning Theory I - Fall'19
Sentiment Analysis on Drug Reviews

Team No: 23 | TA: Christopher
Mounica Subramani | Sai Divya Sangeetha Bhagavatula | Sushma Suresh Kalkunte

1. Summary

Online health-related sites and opinion forums contain abundant information about user preferences and experiences over multiple drugs and treatment. This information can be leveraged to obtain valuable insights using data mining approaches such as sentiment analysis. Sentiment analysis measures the inclination of people's opinions through techniques like text analysis and natural language processing. Online user reviews in this domain contain information related to multiple aspects such as the effectiveness of drugs and side effects, which make automatic analysis very interesting but also challenging. However, analyzing the sentiments of drug reviews can provide valuable insights. They might help pharmaceutical companies and doctors to quickly get into bad reviews and know patients' complaints. This sentiment analysis on drug reviews is basically modeled as a classification problem (i.e.,) classifying the sentiment of the user whether positive, negative or neutral based on their choice of words and their reviews. A lot of symptoms and drug sides were hidden under reviews, which will be great to be automatically extracted to improve the drug and help to give a better prescription.

2. Data

The dataset used in this project provides patient reviews on specific drugs along with related conditions and a 10-star patient rating reflecting overall patient satisfaction. It is derived from Kaggle: https://www.kaggle.com/jessicali9530/kuc-hackathon-winter-2018. There are *215063* records in the dataset. The data is cleaned and split into a training set (*75%*) and a testing set (*25%*). There is a total of seven features in the dataset.

	Unnamed: 0	drugName	condition	review	rating	date	usefulCount
0	206461	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combinati...	9.0	May 20, 2012	27
1	95260	Guanfacine	ADHD	"My son is halfway through his fourth week of ...	8.0	April 27, 2010	192
2	92703	Lybrel	Birth Control	"I used to take another oral contraceptive, wh...	5.0	December 14, 2009	17
3	138000	Ortho Evra	Birth Control	"This is my first time using any form of birth...	8.0	November 3, 2015	10
4	35696	Buprenorphine / naloxone	Opiate Dependence	"Suboxone has completely turned my life around...	9.0	November 27, 2016	37
5	155963	Cialis	Benign Prostatic Hyperplasia	"2nd day on 5mg started to work with rock hard...	2.0	November 28, 2015	43
6	165907	Levonorgestrel	Emergency Contraception	"He pulled out, but he cummed a bit in me. I t...	1.0	March 7, 2017	5
7	102654	Aripiprazole	Bipolar Disorde	"Abilify changed my life. There is hope. I was...	10.0	March 14, 2015	32

Figure 1. A table of train data set displaying values for each feature

The features are described below.

- drugName - Name of the drug for which review has been posted,
- condition - condition/disease for which the drug has been prescribed
- review - user comments for the corresponding drug
- rating - 10-star rating reflecting overall patient satisfaction
- date - the date on which review has been posted
- usefulcount - count of users who found the review to be useful

2.1 Exploratory Data Analysis

Exploratory Data Analysis (*EDA*) helps us in analyzing the data sets to visually summarize their characteristics. It helps us to see what the data can tell us beyond the formal modeling or hypothesis testing task. Here, we have performed Exploratory Data Analysis to visualize the distribution of data with respect to each feature. We are creating 2 classes in sentiment column namely positive (1) and negative (0), based on the rating column.

The below *figure 2* explains the distribution of positive and negative classes across the training and testing data respectively. The distribution is equal in both training and testing data.

Figure 2. Distribution of positive and negative classes (left - training, right - testing)

Figure 3. Word cloud of the top 100 most useful reviews by the users.

Figure 3 is a word cloud of the top 100 most useful reviews posted by the reviewers or the users in the health-related forums. *Figure 4* represents the top 10 reviewed drugs.

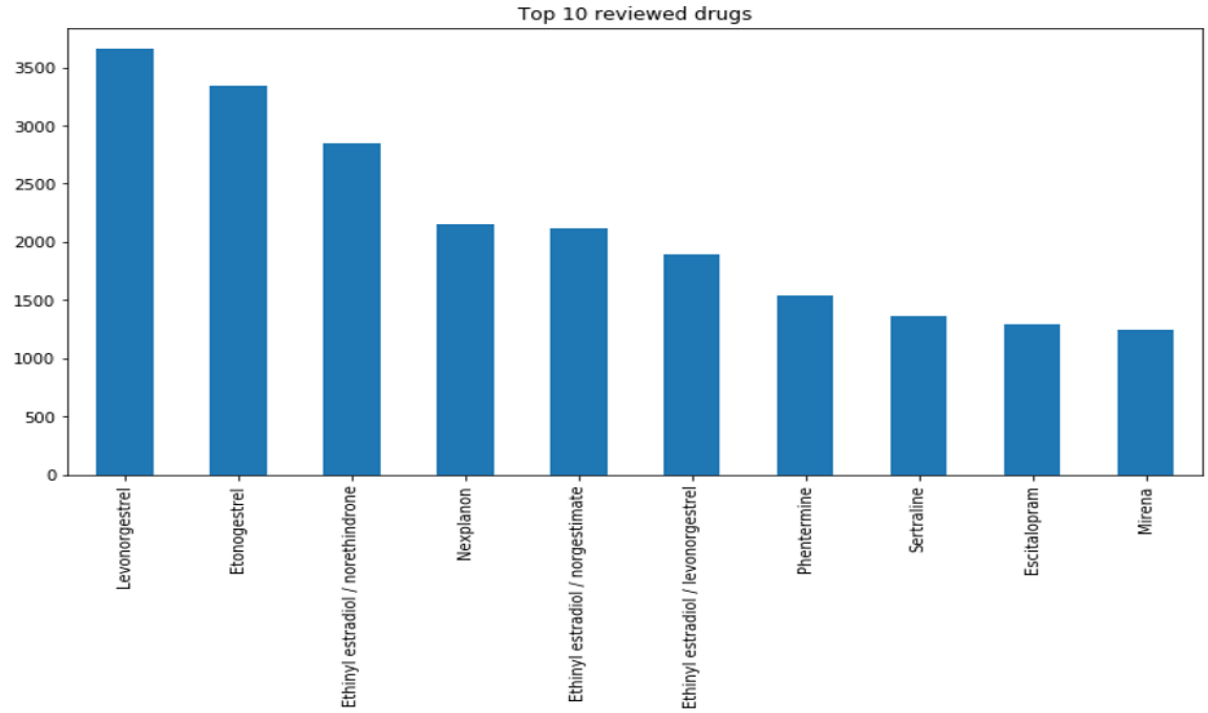


Figure 4. The distribution of the top 10 drugs that are mostly reviewed by the users.

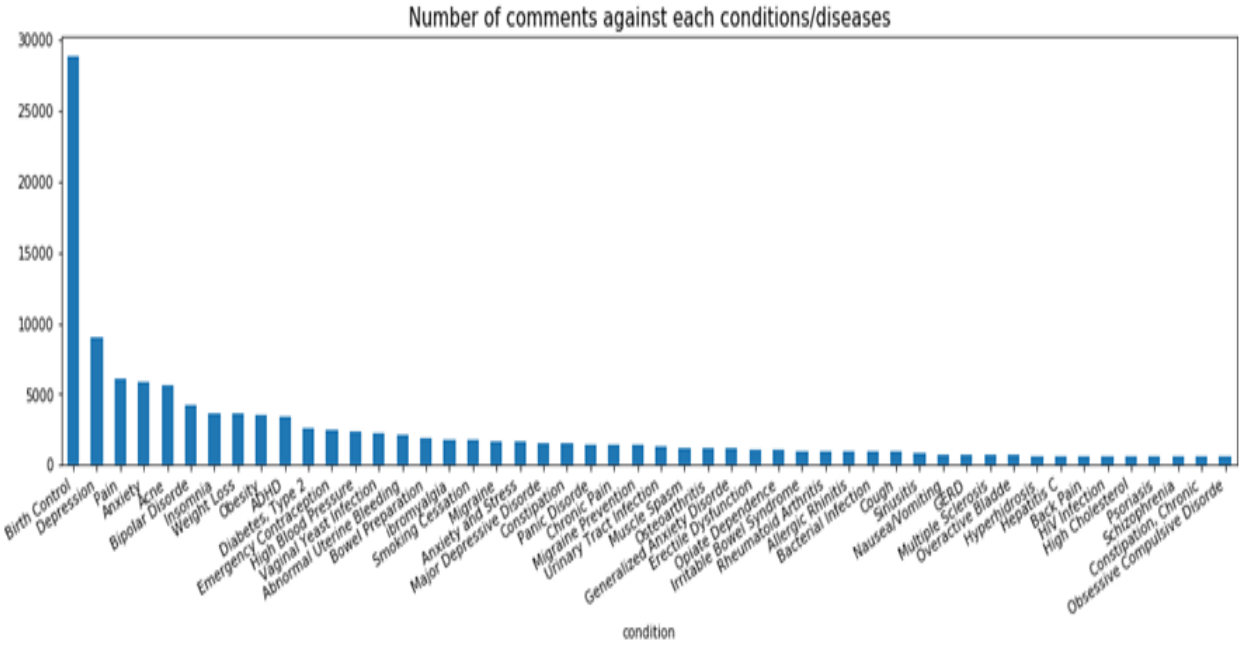


Figure 5. Number of comments against each condition/disease

Figure 5 represents the number of comments posted by the user for each condition or disease. It is observed that "birth control" has seen the greatest number of reviews, followed by depression, anxiety, acne, and pain. OCD has the least number of reviews. *Figure 6* represents the timeline of comments being posted by the users. It is observed that almost a stable pattern is seen between 2008 and 2015 after which the number of reviews increased suddenly. This can be due to an increase in the use of online forums for sharing reviews and content.

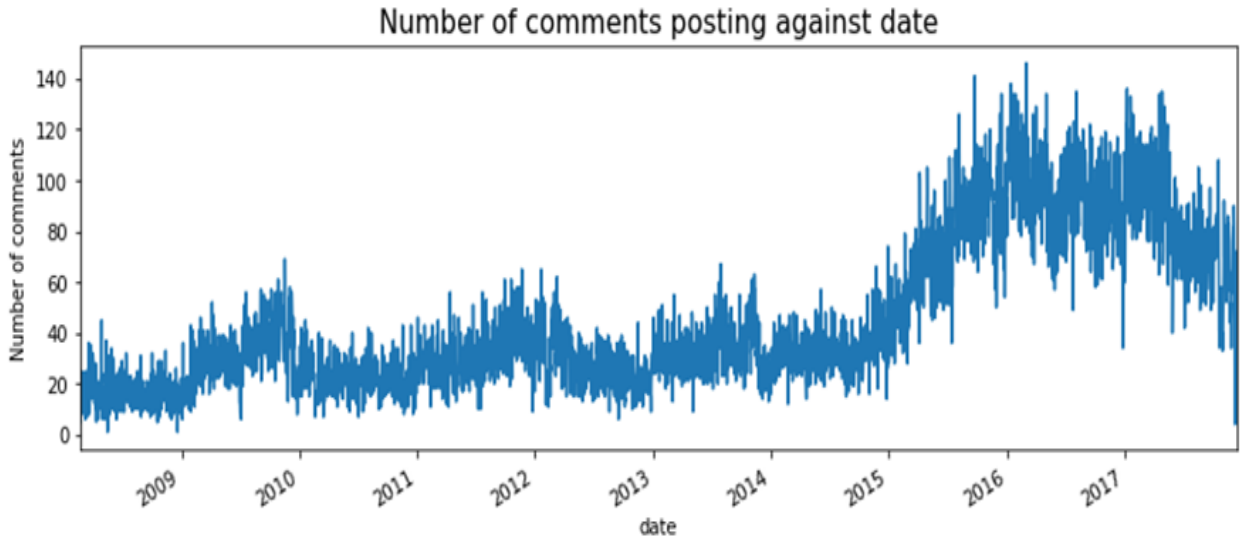


Figure 6. Number of comments posted with respect to the date and year

3. Methodology

We begin by performing data cleaning as a pre-processing step. We then implement modeling techniques to predict the ratings based on the patient reviews

3.1 Data Preprocessing

As our task is to perform sentiment analysis on reviews which is text data, words are our feature set over which we would be building models. Our dataset contains reviews with a lot of unnecessary words and characters which could hamper the performance of our models and must be removed.

The following steps have been incorporated by us to perform data preprocessing:

1) Since the table contains ratings ranging over values from 0 to 10 and we are performing binary classification, we split the ratings from 0 to 5 into the first class 0 implying that the reviews are bad and ratings from 6 to 10 into class 1, implying that the ratings are good
2) A lot of reviews in our data contained HTML characters which have been removed
3) Special characters such as @,#,&, etc. have been removed from the dataset
4) All the characters have been converted to lower-case to maintain uniformity
5) Each review comprises of a lot of words which have been tokenized
6) Lemmatization has been performed so that each word is reduced to its root from
7) Negator words such as ‘don't’, ‘not’, ‘no’ have been removed from the pool of stop words
8) The updated stop words have been removed from the reviews

3.2 Proposed Models

Given the review of the drug, we are performing sentiment analysis and producing the sentiment of the review which is either positive (1) or negative (0). We are cleaning the user reviews to remove all special encoded characters, tags, punctuations as mentioned earlier. We are performing tokenization and lemmatization and removing stop words. We divide the user ratings into two classes *positive* and *negative* and created a new column called *sentiment*. Rating is from the range 0-10, we have classified the reviews to be *positive* when their ratings are greater than or equal to 5, *negative* otherwise.

The models which we have planned to implement are,

- Naïve Bayes Classifier
- Logistic Regression
- Light Gradient Boosting Model (LGBM)
- Convolutional Neural network (CNN)
- Support vector machine (SVM)
- K nearest neighbors (KNN)

4. Implementation

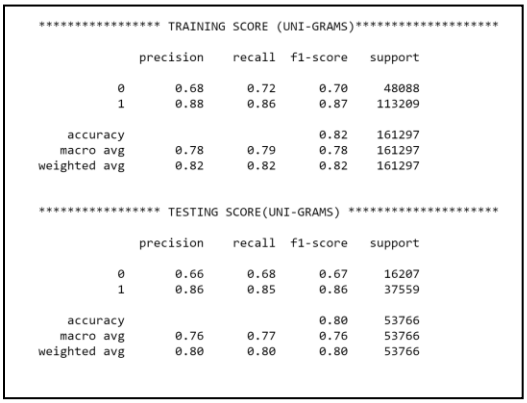
4.1 Naive Bayes Classifier

Multinomial Naïve Bayes belongs to the family of Naïve Bayes Classifiers and has been implemented to classify the reviews as either positive or negative. These classifiers return an estimate of the dependent variable (user rating) by using the probabilities computed for independent variables (patient reviews). Naïve Bayes works by assuming that the features are uncorrelated or independent as compared to other features.
Since we have two classes and we will count the number of words occurring in each class, we have implemented Binary Multinomial Naïve Bayes by writing the algorithm from scratch.

Steps of implementation followed-

1) We assign 1 if the word appears and 0 if it does not and count the number of occurrences of 1 and 0.
2) The cleaned reviews have been treated as “bag of words” meaning that the order in which words appear does not matter.
3) By iterating through every word of the review of each of the classes, we compute the probability of the occurrence of each word.
4) Since our object is to identify the sentiment, whether a word appears in a certain class is more important to us than how often it appears.
5) In our implementation, we have accounted for floating-point underflow (during computation, machines return values up-till only a few decimal places and sometimes since the probabilities are all between 0 and 1, they are rounded off). In order to prevent floating-point underflow, we consider the logarithm of their probability values.
6) Laplace smoothening has been added to account for zero probabilities

Upon implementation of our Naïve Bayes Algorithm, we obtained a low accuracy of 69% on the testing data and 72% on the training data. We then proceeded to implement the same model using the sklearn package. Upon considering the features as unigrams (each word of the review is a different feature), we obtained the results as shown in figure 7.



********** TRAINING SCORE (UNI-GRAMS)**********

	precision	recall	f1-score	support
0	0.68	0.72	0.70	48088
1	0.88	0.86	0.87	113209
accuracy			0.82	161297
macro avg	0.78	0.79	0.78	161297
weighted avg	0.82	0.82	0.82	161297

********** TESTING SCORE(UNI-GRAMS) **********

	precision	recall	f1-score	support
0	0.66	0.68	0.67	16207
1	0.86	0.85	0.86	37559
accuracy			0.80	53766
macro avg	0.76	0.77	0.76	53766
weighted avg	0.80	0.80	0.80	53766

Figure 7. Train and Test Metrics of Naïve Bayes with Unigrams

We observe that the accuracy for training and testing has improved as compared to our implementation. But the precision and recall values are still low at 66% and 68% respectively on the testing dataset. To improve the predictions of the model, we then implemented Naïve Bayes by providing it with bigrams (a combination of 2 words at a time as features). The results of our implementation are shown below-

```
****************** TRAINING SCORE ********************

              precision    recall  f1-score   support

           0       0.98      0.95      0.97     48088
           1       0.98      0.99      0.99    113209

    accuracy                           0.98    161297
   macro avg       0.98      0.97      0.98    161297
weighted avg       0.98      0.98      0.98    161297


****************** TESTING SCORE *********************

              precision    recall  f1-score   support

           0       0.91      0.76      0.82     16207
           1       0.90      0.97      0.93     37559

    accuracy                           0.90     53766
   macro avg       0.90      0.86      0.88     53766
weighted avg       0.90      0.90      0.90     53766
```

Figure 8. Train and Test Metrics of Naïve Bayes with Bigrams

We now notice that the accuracy, precision, and recall values have increased significantly proving our hypothesis. Below is a ROC for the model. The Area Under Curve value upon computation was found to be 0.86.

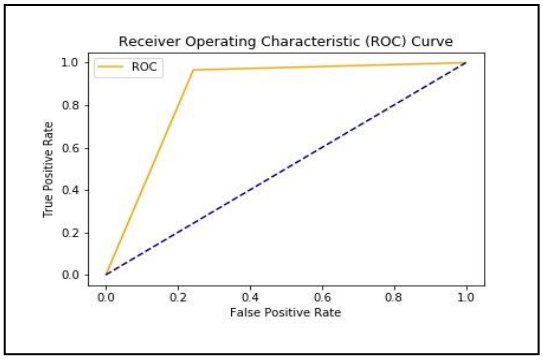


Figure 9. ROC curve for bigrams

4.2 Light Gradient Boosting Model

LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages like faster training speed and higher efficiency, lower memory usage, better accuracy, supports parallel learning and GPU learning and is capable of handling large scale data. It is different from the rest of the tree algorithms as LightGBM grows vertically (leaf wise) whereas other tree algorithms grow horizontally (level wise). However, it is sensitive to overfitting. One important task in implementing a LightGBM model is parameter tuning. We have used the following parameters for our model:

```
param = {'num_leaves':100, 'num_trees':300, 'objective':'binary', "max_bin":255, "learning_rate":0.1, "is_unbalance": True}
```

- *num_leaves*: Since LightGBM uses a leaf-wise tree growth algorithm. So, this parameter is used to control the complexity of the tree model. Other tree models use *max_depth* and the relation between these two is *num_leaves* = 2^(*max_depth*).
- *num_trees*: Otherwise called "*num_iterations*" which denotes the number of boosting iterations.
- objective: Refers to *binary_logloss* metric.
- *max_bin*: It is used for better speed

- *learning_rate*: By default, the value is 0.1
- *is_unbalance*: By default, it is set to false. It is used only in binary and "*multiclass*" applications. Enabling this to True will increase the overall performance metric of the model. But it will also result in poor estimates of the individual class probabilities.

The accuracy of the LightGBM model on the testing data is 88.5% and the training data is 90%. The ROC curve and AUC value is given below.

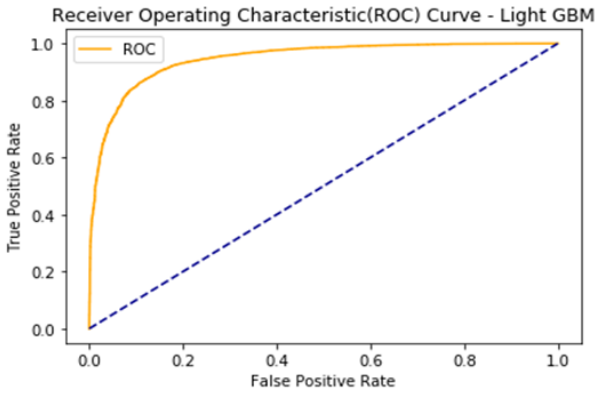


Figure 10. ROC curve for LightGBM (AUC = 0.95)

4.3 K-Nearest Neighbors

K-Nearest Neighbors algorithm is used to classify by finding the K nearest matches in training data and then using the label of closest matches to predict. Traditionally, distance such as Euclidean distance is used to find the closest match. We have used the nltk library to generate synonyms and use similarity scores among texts. How KNN works is, initially the entire training data set is stored and when a prediction is required, the k-most similar records to a new record from the training dataset are then located. From these neighbors, a summarized prediction is made. This similarity between records is measured by Euclidean distance as mentioned earlier. Once the neighbors are discovered, the summary prediction can be made by returning the most common outcome or taking the average. This is a lazy learning method as the computation is deferred until prediction. It accepts one parameter K which denotes how many closest neighbors will be used to make the prediction. We have tried K= 1 to 5 and the metrics reported here are with respect to the k value 5. We have implemented KNN using bigrams as unigrams didn't provide better results.

The accuracy of the K-Nearest Neighbor algorithm is 78% on training data and 77% on testing data. The ROC curve and AUC value are given below.

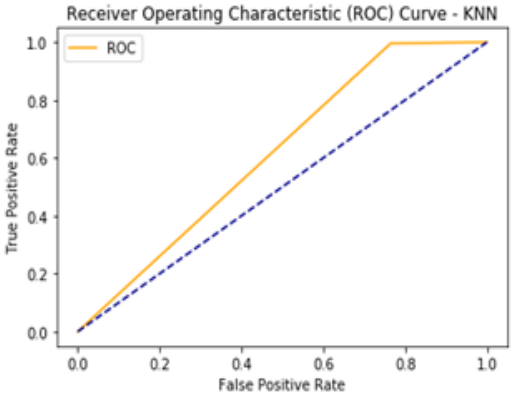


Figure 11. ROC curve for KNN model (AUC = 0.62)

4.4 Support Vector Machine (SVM)

A support vector machine is a linear classifier that works by finding a hyperplane in a 'n' dimensional space where n is the total number of features. In order to separate the features, there are many possible planes that could be drawn but the motive of SVM is to find one plane that has maximum distance between the vectors of both the classes. These hyperplanes act as decision boundaries that help in classifying the data distinctly. The data transformation is done with the help of kernels and we have used the linear kernel as our data is linearly separable into two classes. Vectorization of the data has been performed used the TF-IDF method (this process would be explained later as a part of the Logistic Regression model). We first train the SVMs on unigrams and after observing a lower accuracy, we proceed to train them on bigrams. The metric after training and testing are as shown in Figure 12.

********** SVM (BIGRAMS)**********

**********TRAINING REPORT **********

	precision	recall	f1-score	support
0	1.00	1.00	1.00	48088
1	1.00	1.00	1.00	113209
accuracy			1.00	161297
macro avg	1.00	1.00	1.00	161297
weighted avg	1.00	1.00	1.00	161297

**********TESTING REPORT **********

	precision	recall	f1-score	support
0	0.92	0.88	0.90	16207
1	0.95	0.97	0.96	37559
accuracy			0.94	53766
macro avg	0.93	0.92	0.93	53766
weighted avg	0.94	0.94	0.94	53766

Figure 12. Metrics for SVM

As it can be observed, the model has achieved an accuracy of 94% which is very good. The precision and recall values are also high.
The ROC curve for this model is as shown in Figure 13. The computed AUC is 0.82

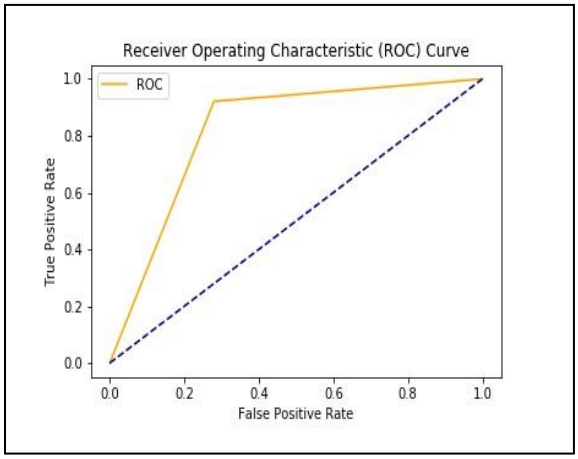


Figure 13. ROC curve for SVM model (AUC = 0.82)

4.5 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of deep learning technique that uses the architecture of artificial Recurrent Neural Network (RNN). RNN uses feedback loops to retain information in the neuron and the output from

the last step is fed as input in the current step. Although RNNs can accurately predict new data based on learning from recent information, they perform poorly while predicting from long term memory. LSTM, on the other hand, retains information for long periods of time and are best suited for applications with sequential data. Since the context of the sentence is extremely important to understand the emotion associated with it, LSTM has been used to predict the sentiment of the patient reviews.

Steps to implement LSTM-

1) Compute the total number of words in the training and testing dataset
2) Compute the vocabulary length
3) Compute the length of the longest sentence
4) Tokenize the reviews and convert them to sentences (This creates an index mapping dictionary so that frequently used words are assigned indexes that are low)
5) Pad the shorter reviews to the maximum length of the sentence or truncate longer sentences to a fixed number to maintain uniformity
6) Split the training dataset in the ratio of 80:20 for training and validation respectively
7) Build the architecture of the network
8) Use the architecture to train the model on the training dataset and then validate on the testing set
9) Choose the model with the best performance (Highest accuracy and low loss)

Apply the chosen model on the testing data set and measure accuracy and error

```
def create_conv_model():
    model_conv = Sequential()
    model_conv.add(Embedding(41914, 100, input_length=973))
    model_conv.add(Dropout(0.2))
    model_conv.add(Conv1D(64, 5, activation='relu'))
    model_conv.add(MaxPooling1D(pool_size=4))
    model_conv.add(LSTM(100))
    model_conv.add(Dense(1, activation='sigmoid'))
    model_conv.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model_conv
```

Figure 14. The architecture of LSTM Model

Figure14 shows the architecture of the LSTM network used for training the model. The networks start with an embedding layer which expands each of the tokens to a large vector. The first argument in this layer is the size of the vocabulary and the second parameter represents the dimension of the embeddings. The third parameter is the length of input (maximum length of the review).

Only using the LSTM layer takes up a lot of time (on a CPU). Keeping this in mind, the number of epochs has only been set to 3. To further improve the training time and accuracy, a convolution layer with a filter size of 5 and “relu” activation has been added to the network before LSTM. The last layer is a dense feed-forward network with “sigmoid” activation. Binary cross entropy has been used to compute the loss. To prevent the model from overfitting a dropout of 0.2 has been considered.

```
112907/112907 [==============================] - ETA: 58s - loss: 0.1686 - accuracy: 0.934 - ETA: 56s - loss: 0.1686 - ac
curacy: 0.934 - ETA: 55s - loss: 0.1686 - accuracy: 0.934 - ETA: 54s - loss: 0.1687 - accuracy: 0.934 - ETA: 52s - loss:
0.1687 - accuracy: 0.934 - ETA: 51s - loss: 0.1688 - accuracy: 0.934 - ETA: 49s - loss: 0.1688 - accuracy: 0.934 - ETA: 4
8s - loss: 0.1689 - accuracy: 0.934 - ETA: 47s - loss: 0.1689 - accuracy: 0.934 - ETA: 45s - loss: 0.1689 - accuracy: 0.9
34 - ETA: 44s - loss: 0.1689 - accuracy: 0.934 - ETA: 43s - loss: 0.1689 - accuracy: 0.934 - ETA: 41s - loss: 0.1689 - ac
curacy: 0.934 - ETA: 40s - loss: 0.1688 - accuracy: 0.934 - ETA: 39s - loss: 0.1688 - accuracy: 0.934 - ETA: 37s - loss:
0.1688 - accuracy: 0.934 - ETA: 36s - loss: 0.1689 - accuracy: 0.934 - ETA: 35s - loss: 0.1689 - accuracy: 0.934 - ETA: 3
3s - loss: 0.1690 - accuracy: 0.934 - ETA: 32s - loss: 0.1690 - accuracy: 0.934 - ETA: 31s - loss: 0.1691 - accuracy: 0.9
34 - ETA: 29s - loss: 0.1691 - accuracy: 0.934 - ETA: 28s - loss: 0.1691 - accuracy: 0.934 - ETA: 27s - loss: 0.1692 - ac
curacy: 0.934 - ETA: 25s - loss: 0.1691 - accuracy: 0.934 - ETA: 24s - loss: 0.1692 - accuracy: 0.934 - ETA: 23s - loss:
0.1692 - accuracy: 0.934 - ETA: 21s - loss: 0.1693 - accuracy: 0.934 - ETA: 20s - loss: 0.1693 - accuracy: 0.934 - ETA: 1
8s - loss: 0.1692 - accuracy: 0.934 - ETA: 17s - loss: 0.1693 - accuracy: 0.934 - ETA: 16s - loss: 0.1693 - accuracy: 0.9
34 - ETA: 14s - loss: 0.1693 - accuracy: 0.934 - ETA: 13s - loss: 0.1693 - accuracy: 0.934 - ETA: 12s - loss: 0.1693 - ac
curacy: 0.934 - ETA: 10s - loss: 0.1693 - accuracy: 0.934 - ETA: 9s - loss: 0.1692 - accuracy: 0.934 - ETA: 8s - loss: 0.
1693 - accuracy: 0.93 - ETA: 6s - loss: 0.1693 - accuracy: 0.93 - ETA: 5s - loss: 0.1693 - accuracy: 0.93 - ETA: 4s - los
s: 0.1694 - accuracy: 0.93 - ETA: 2s - loss: 0.1693 - accuracy: 0.93 - ETA: 1s - loss: 0.1693 - accuracy: 0.93 - ETA: 0s
- loss: 0.1693 - accuracy: 0.93 - 1672s 15ms/step - loss: 0.1693 - accuracy: 0.9345 - val_loss: 0.2821 - val_accuracy: 0.
8924
```

Figure 15. Model Results

As shown in Figure15 the model after training has achieved an accuracy of 0.9345 and the loss value is 0.1693 which is good. Testing has resulted in an accuracy of 0.8924 and 0.28 loss.

Figure 16. Model Accuracy

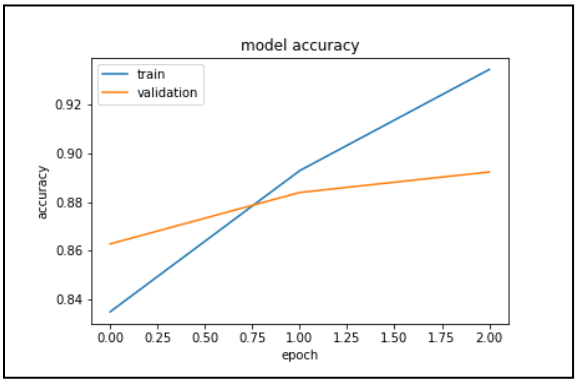


Figure 17. Model Loss

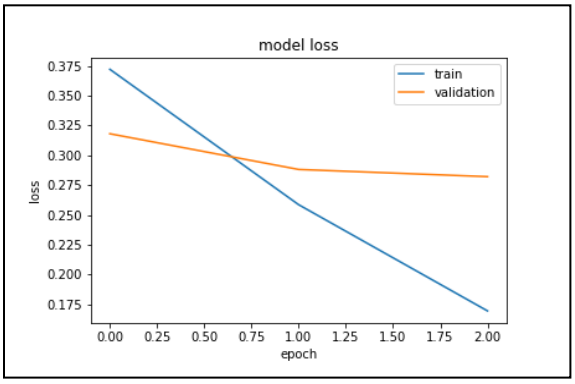


Figure16 and Figure17 represent model accuracy and loss. The training accuracy can be noticed to increase linearly until two epochs. The validation accuracy starts from around 86% and increases until it reaches 89%, after which it becomes stable. The validation loss decreases rapidly from 0.37 on the training set but fails to show the same rate during testing. On the whole, it can be observed that LSTM performs well correctly predicting the sentiments. The model can be made better by training with a greater number of layers.

4.6 Logistic Regression

Logistic Regression is a linear model that is used when observations must be assigned to a discrete set of classes. For sentiment analysis, it can help predict the rating of the review by probabilistically assigning them to one of the two classes (good or bad). In order to map the predicted outcomes to the classes, it uses a sigmoid function. Using this model for sentiment analysis is advantageous as it is a simple classifier that can train fast.

For the model to learn on the features, the words must be converted to the numerical format. For this purpose, the TF-IDF vectorizer has been used. TF-IDF stands for Term Frequency-Inverse Document Frequency. It works by considering the frequency of a word in each review and comparing it with its occurrence in all the reviews. Hence,

the importance of a word in a review increases proportionally to the number of times it appears in a review, but it's offset by the frequency of the words in the complete corpus of reviews.
The logistic regression model has been trained with L1 Regularization and the model has achieved an accuracy of 87.32% on the testing data. Other metrics computed for training and testing are shown in figure18 and figure19 respectfully.

```
**** Logistic Rgression Metrics(Training) ****
Accuracy- 0.8732772463220023
Error- 0.12672275367799768
Precision-  0.8924043732868179
Recall-  0.9453647027767237
F1 Score-  0.9181214398448955
```

Figure 18. Training Metric

```
**** Logistic Rgression Metrics(Testing) ****
Accuracy- 0.8615481903061414
Error- 0.13845180969385862
Precision-  0.8841787411343367
Recall-  0.9380168367727035
F1 Score-  0.9103024460778407
```

Figure 19. Testing Metric

The performance can be seen in the ROC Curve for training and testing as shown. Although the model doesn't perform as well as LSTM, it still can be considered a good option to predict the rating. Upon computation, the Area Under the Curve (AUC) comes up to 0.80 for training and 0.79 for testing.

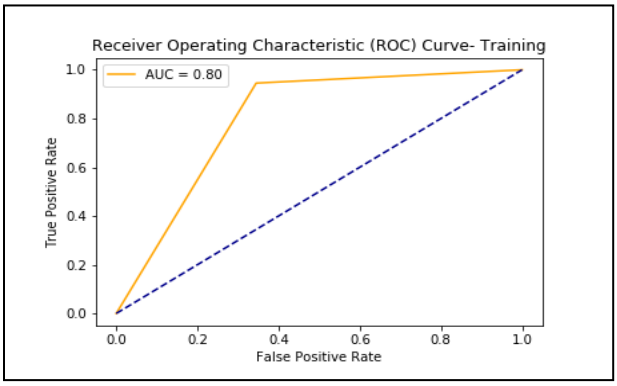


Figure 20. Training ROC

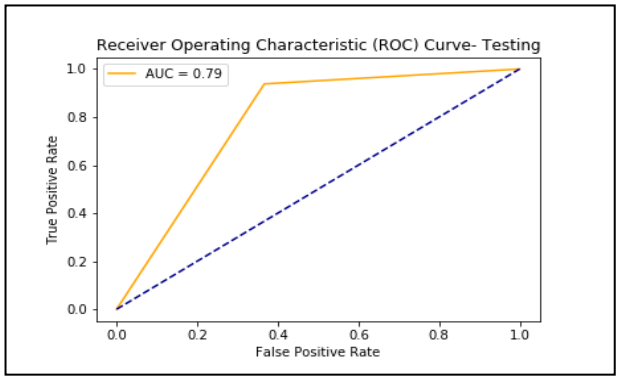


Figure 21. Testing ROC

5. Results

	NBC	SVM	KNN	LGBM	LSTM	Logistic
Accuracy	**0.90**	**0.94**	**0.77**	**0.8805**	**0.8924**	**0.8732**
Error	**0.10**	**0.06**	**0.23**	**0.1195**	**0.1076**	**0.1268**
Precision	**0.905**	**0.94**	**0.708**	**-**	**-**	**0.8841**
Recall	**0.965**	**0.965**	**0.96**	**-**	**-**	**0.9380**
F1	**0.93**	**0.957**	**0.825**	**-**	**-**	**0.9103**

6. Conclusion

We have applied all our proposed approach to the drug review dataset, particularly focusing on the sentiment of the review by classifying it as either positive or negative. Among all the approached models, Support Vector Machine (SVM) performs the best in our scenario of sentiment analysis. Long Term Short Term (LSTM) model also performs well but with high training and testing time. Light GBM performs moderately well and it is fast compared to other models. Naive Bayes model has low accuracy comparatively. And K-Nearest Neighbor has the highest error as it isn't learning anything new but just classifying the given data.

7. Future Work

So far, we have taken two classes negative (0) and positive (1) for classifying the sentiment. In the future, we can try splitting the rating into multiple classes and implement the approached models to compare their performance. Also, we can build a recommendation engine to recommend medicine to the patient based on previous reviews of the drugs. In terms of model, we can try an ensemble of Convolutional Neural Network and LSTM with multiple layers instead of a single layer and observe if there are any changes in the performance of the model. We can try implementing logistic regression with bigrams instead of unigrams. We can also try to improve the K-Nearest Neighbor model for various values of K greater than 5 and see where the model converges for better accuracy.

8. Statement of Contributions

1. Mounica Subramani: Performed EDA, cleaned data and build Light Gradient Boosting Model and K nearest neighbor implementation and optimized the models built. Prepared presentation and report.
2. Sai Divya Sangeetha Bhagavatula: Performed EDA, cleaned data and build Naïve Bayes Model and Support Vector Machine implementation and optimized the models built. Prepared presentation.
3. Sushma Suresh Kalkunte: Performed EDA, cleaned data and working on building a Convolutional Neural Network Model and Logistic Regression implementation and optimized the models built. Prepared presentation and report.

8. Reference

1. https://www.kaggle.com/jessicali9530/kuc-hackathon-winter-2018#drugsComTest_raw.csv
2. https://www.kaggle.com/jessicali9530/kuc-hackathon-winter-2018#drugsComTrain_raw.csv
3. Sentiment Analysis on Twitter Data using KNN and SVM. https://thesai.org/Downloads/Volume8No6/Paper_3-Sentiment_Analysis_on_Twitter_Data_using_KNN_and_SVM.pdf
4. Sentiment Analysis of user-generated content on drug review websites https://www.researchgate.net/publication/277625450_Sentiment_Analysis_of_User-Generated_Content_on_Drug_Review_Websites
5. Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780. DOI:10.1162/neco.1997.9.8.1735. PMID 9377276
6. Vance, Ashlee (May 15, 2018). "Quote: These powers make LSTM arguably the most commercial AI achievement, used for everything from predicting diseases to composing music". Bloomberg Business Week. Retrieved 2019-01-16.
7. Strobl C, Boulesteix A L, Zeileis A, et al. Bias in random forest variable importance measures: Illustrations, sources and a solution[J]. BMC bioinformatics, 2007, 8(1): 25.
8. Github link for the code: https://github.com/MounicaSubramanium/Sentiment-Analysis-on-Drug-Reviews
9. Presentation Link: https://github.com/MounicaSubramanium/Sentiment-Analysis-on-Drug-Reviews/blob/master/presentation_sml_final.pptx