Medicine Recommendation using Review Mining

Yashaswini Joshi
School of Information
University of Michigan
yjoshi@umich.edu

December 8, 2020

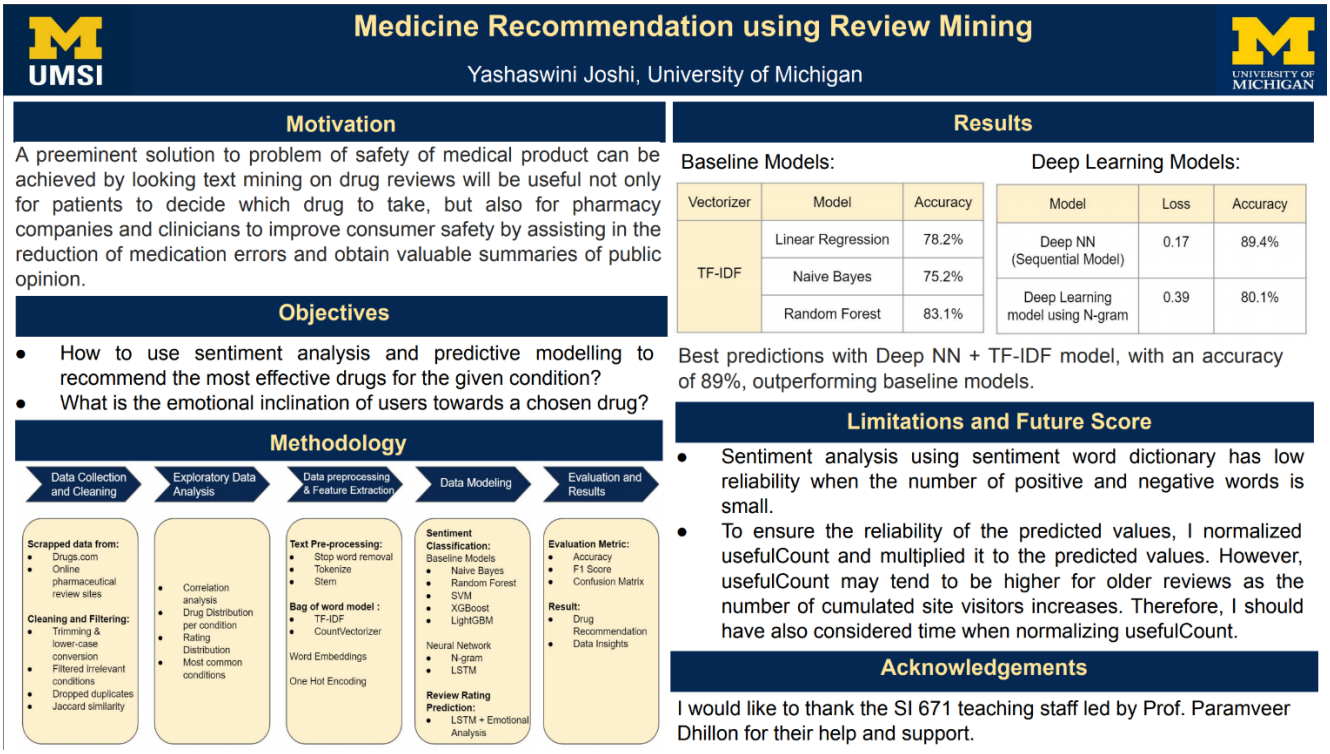


Fig. 1 Project Poster

1. Motivation:

The safety of a medical product is evaluated through predefined protocols and clinical trials. A preeminent solution to this problem can be achieved by looking at post marketing drug surveillance methods such as drug review analysis to monitor drug-related issues. Text mining on drug reviews will be useful not only for patients to decide which drug to take, but also for pharmacy companies and clinicians to improve consumer safety by assisting in the reduction of medication errors and obtain valuable summaries of public opinion.

2. Related Work:

Previous studies have shown that health-related user-generated content is useful from different points of view. One of the benchmark papers in this area was written by Jane Sarasohn-Kahn [1]. It states that users are often looking for stories from "patients like them" on the Internet, which is hard to find among their friends and family. For the past decade, researchers have been analyzing the emotional impact of user experience and the severity of adverse drug reactions by extracting sentiment and semantic information [2].

Due to a lack of trust and quality of user-expressed medical language, extensive research in the medical and health domain has not been done. Therefore, I aim to analyse symptoms and get drug recommendations, side effects of drugs and obtain insights into patients' Health Condition.

3. Problem Statement:

Analyzing the Drugs Descriptions, conditions, reviews and then recommending the medicines for each Health Condition of a Patient.

Research Questions:

i. How to use sentiment analysis and predictive modelling to recommend the most effective drugs for the given condition?
ii. What is the emotional inclination of users towards a chosen drug?
iii. What elements of a review make it more helpful to others? Which patients tend to have more negative reviews?
iv. To determine if a review is positive, neutral, or negative
v. What kind of drugs are there? What sorts of conditions do these patients have?

4. Dataset:

Drug Review Dataset from the UCI Machine Learning Repository [3] and reviews from online pharmaceutical review sites. The dataset provides patient reviews on specific drugs along with related conditions and a 10-star patient rating reflecting overall patient satisfaction. The data was obtained by crawling online pharmaceutical review sites. The intention was to study.

5. Design of the methodological approach:

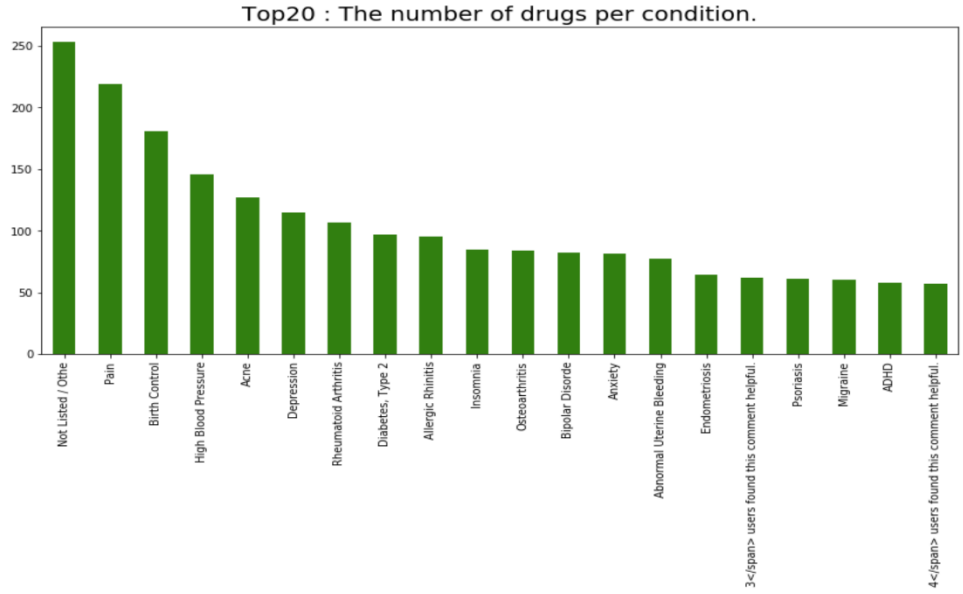
In the data exploration part, I will look at data types with visualization techniques and statistical techniques. Through this process, I can set the topic, preprocess the data to fit the objective, and create various variables to fit model.

5.1 Data Understanding and Preprocessing:

The result of looking at the data through the head () command shows us that there are six variables except for the unique ID that identifies the individual, and review is the key variable. The structure of the data is that a patient with a unique ID purchases a drug that meets his condition and writes a review and rating for the drug he/she purchased on the date. Afterwards, if the others read that review and find it helpful, they will click usefulCount, which will add 1 for the variable.

First, I will start exploring variables, starting from uniqueID. I compared the unique number of unique IDs and the length of the train data to see if the same customer has written multiple reviews, and there weren't more than one reviews for one customer.
unique values count of train: 161297
length of train: 161297

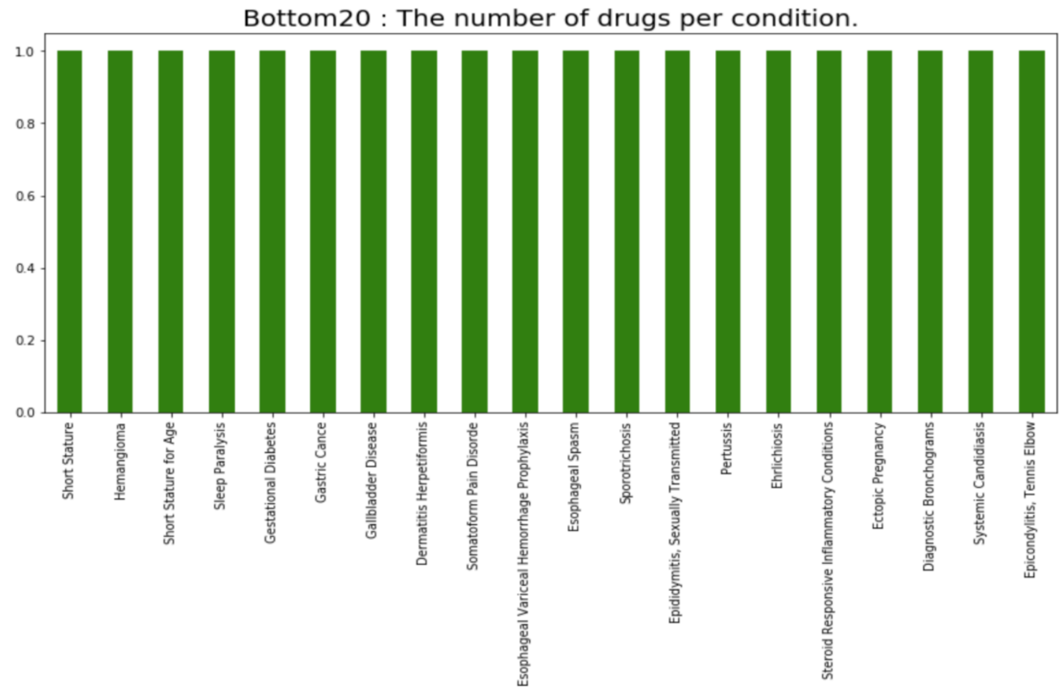
DrugName is closely related to condition, so I have analyzed them together. The unique values of the two variables are 3671 and 917, respectively, and there are about 4 drugs for each condition.



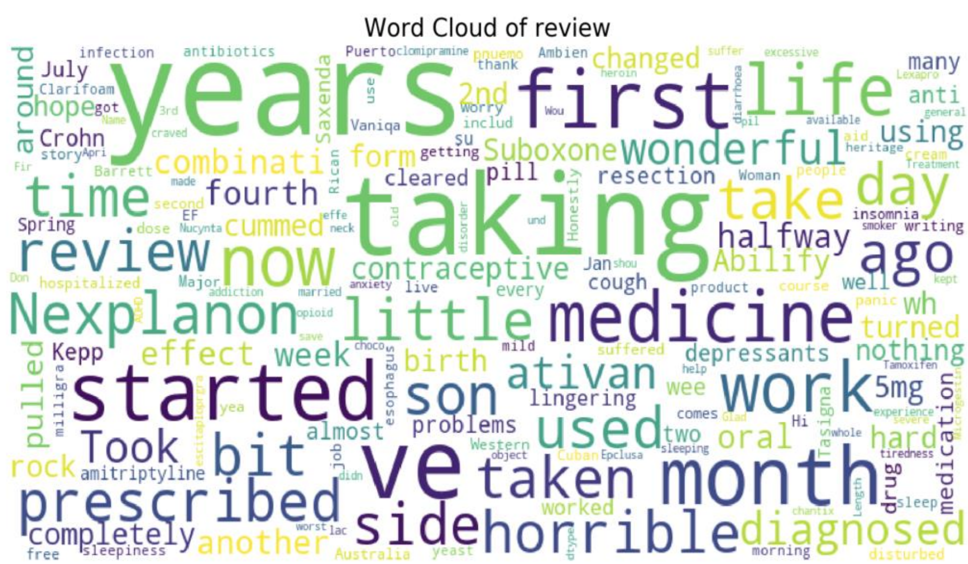
As you can see from the picture above, the number of drugs for top eight conditions is about 100 for each condition. On the other hand, it should be noted that the phrase "3 users found this comment helpful" appears in the condition, which seems like an error in the crawling process. I have investigated it to see in more details.

It is expected that for structure of '</ span> users found this comment helpful.' phrase, there will be not only 3, but also 4 as shown above, and other numbers as well. I will remove these data in the future preprocessing.

The following are the low 20 conditions of 'drugs per condition'. As you can see, the number is all 1. Considering the recommendation system, it is not feasible to recommend with that when there is only one product. Therefore, I will analyze only the conditions that have at least 2 drugs per condition.

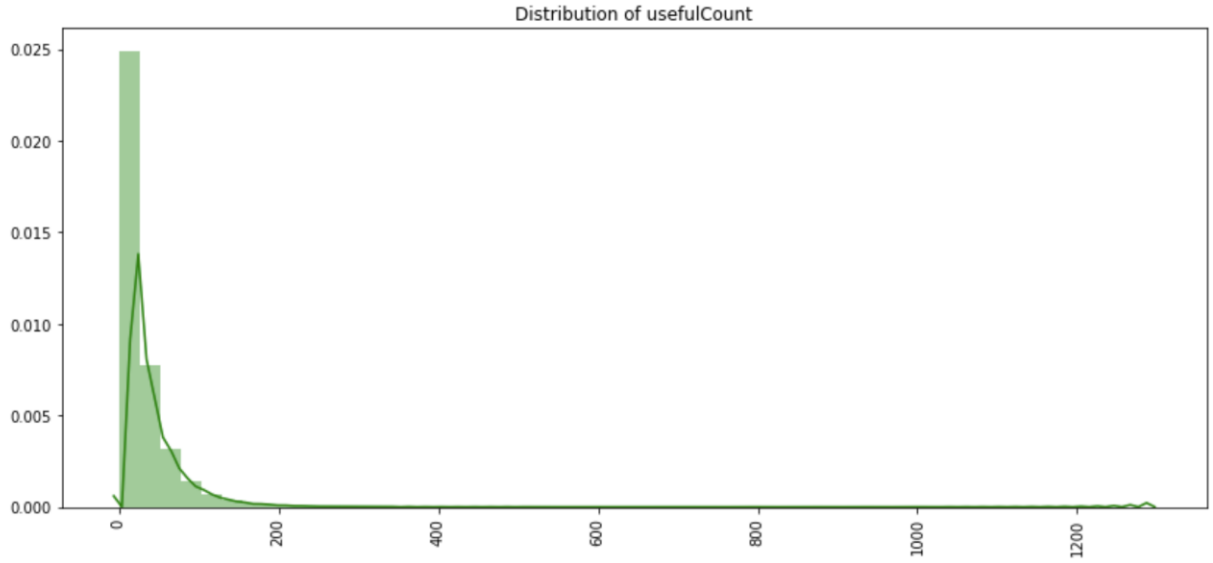


Let’s see how word cloud of review looks like.



Next, I will classify 1 ~ 5 as negative, and 6 ~ 10 as positive, and we will check through 1 ~ 4 grams which corpus best classifies emotions. When I use 1-gram, I can see that the top 5 words have the same contents, although the order of left (negative) and right (positive) are different. This means when I analyze the text with a single corpus, it does not classify the emotion well. So, I will expand the corpus. Likewise, in 2-gram, the contents of the top five corpus are similar, and it is hard to classify positive and negative. In addition, 'side effects' and 'side effects.' are interpreted differently, which means preprocessing of review data is necessary. However, I can see that this is better to classify emotions rather than previous 1-grams, like side effects, weight gain, and highly recommend. From 3-gram I can see that there is a difference between positive and negative corpus. Bad side effects, birth control pills, negative side effects are corpus that classify positive and negative. However, both positive and negative parts can be thought that it has missing parts that reverses the context, such as' not' in front of a corpus. Clearly, 4-gram classifies emotions much better than other grams. Therefore, we will use 4-gram to build deep learning model. (I will add the figures in final report).

I also checked if any day of the month affects the rating like salary day, but it seems like it does not make much difference. Next, I looked for relationship between rating and weather. Interestingly, I found that the average rating differs by year, but it is similar by month.



From the distribution of usefulCount shown above, it can be observed that the difference between minimum and maximum is 1291, which is high. In addition, the deviation is huge, which is 36. The reason for this is that the more drugs people look for, the more people read the review no matter their contents are good or bad, which makes the usefulcount very high. So, when I create the model, I will normalize it by conditions, considering people's accessibility.

I preprocessed data by removing the missing values and also deleting conditions with only one drug. Removed the stop words from the reviews.

At the model part, for sentiment analysis of reviews I will apply random forests. The obvious starting question for such an approach is how I can convert the raw text of the review into a data representation that can be used by a numerical classifier. To this end, I will use the process of vectorization. By vectorizing the 'review' column, I can allow widely varying lengths of text to be converted into a numerical format which can be processed by the classifier. This can be achieved via the TF-IDF method which involves creating tokens (i.e. individual words or groups of words extracted from the text). The main limitation of this approach can be that it does not consider the relative position of words within the document. Hence, I am planning on only using the frequency of occurrence. For emotion analysis using word dictionary, deep learning with n-gram, etc. can also be used. For the emotional analysis I am planning to use Harvard emotional dictionary. I can compare both these and use the one with good results. For classification I am planning to use Neural network with Keras. Keras gives us lots of freedom to play with hyperparameters and design a network that would be best suited for this data. For feature analysis idea of attempting to derive importance out of the vectorized features by k-clustering by similarity, as keras has no built-in function to check for feature importance.

To compensate the limitation of natural language processing, Lightgbm machine learning model can be used, and reliability can be further secured through useful count.

5.2 Classification with sk-learn and Random Forests:

As mentioned in the project proposal, I am vectorizing the 'review' column and tokenizing them using TF-IDF. Once the list of tokens is created, they are assigned an index integer identifier which allows them to be listed. I can then count the number of words in the document and normalize them in such a way that de-emphasizes words that appear frequently (like "a", "the", etc.). This creates what is known as a bag (multi-set) of words. Such a representation associates a real-valued vector to each review representing the importance of the tokens (words) in the review. This represents the entire corpus of reviews as a large matrix where each row of the matrix represents one of the reviews and each column represents a token occurrence.

Term-Frequency Inverse Document-Frequency (TF-IDF) is a way of handling the excessive noise due to words such as "a", "the", "he", "she", etc. Clearly such common words will appear in many reviews, but do not provide much insight into the sentiment of the text and their high frequency tends to obfuscate words that provide significant insight into sentiment. More details about the method can be found on wikipedia.

The simplest type of model I can attempt to fit on this data is the Naive Bayes classifier. I will first test Naive Bayes on a binarized version of the rating column which attempts to identify which reviews are favorable. I define a favorable review as one which

received a rating above 5. Given the sample size involved for the data set, I choose Naive Bayes over other classifiers due to its scalability.

Accuracy: 0.7527247703009337

For Random Forest,
Accuracy: 0.8310084439980657
Confusion Matrix
[[7252 131]
 [8955 37428]]

It can be observed that a more complicated classifier gets us a roughly 8% increase in the accuracy of my classifier.

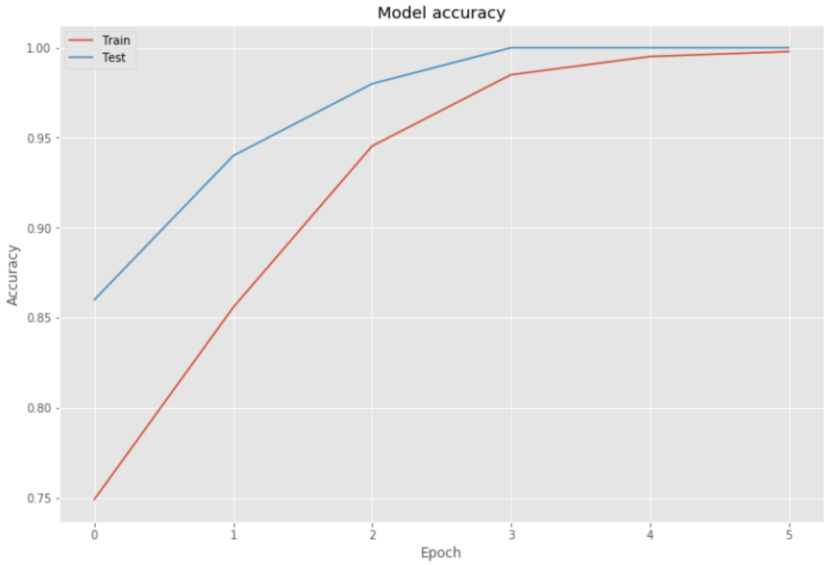
5.3 Classification with Keras:

Neural networks usually scale very well with lots of data, and that is exactly what I have here (150,000 training examples, 50,000 test examples). I could not run all of these through sk-learn as the kernel would just crash after a couple of minutes. However, keras plays with this many data very nicely. While I am sticking to a simple NN in this kernel, other types of NN architecture can deal with natural language processing problems very well (long short-term memory models, other types of recurrent networks). Keras gives me lots of freedom to play with hyperparameters and design a network that would be best suited for the data.

For this I did some preprocessing by removing all symbols from the reviews. I use a CountVectorizer to vectorize each of the different reviews into 5000 feature row vectors. This is a very similar approach to vectorization as the TDIF explained above. This vectorizer is also set to not add in very common words such as "and" and "the" as features. This is specified in stop_words. Then I Created a helper function to easily create the y labels. I need to transform the review column to a m number of reviews by 3 columns, where index 0 is negative, 1 is positive and 2 is neutral. I experimented a lot with different kinds of simple NN architecture and this is what I concluded:

1. Softmax was the best activation function for the output layer.
2. Vectorizing with the TFIDF vectorizer took longer to converge but had roughly the same accuracy as when I vectorized with the CountVectorizer. Changing binary and lowercase didn't result in any changes.
3. 256 units was the sweet spot for accuracy without overfitting.
    a. Any units above would give me less accuracy as the model continued to train as well as overall performing worse, and less units just gave me worse accuracy overall.
    b. I also discovered that adding any additional layers to the model resulted in a loss of accuracy, even when I kept the number of layer units small.

4. I experimented with different numbers of epochs and batch sizes as well, and found roughly 6 epochs and a batch size of 128 to be optimal.
5. Anything more than 6 epochs resulted in the model beginning to overtrain and lose accuracy.
6. Before running preprocessing on the reviews by eliminating symbols, the validation accuracy stayed around 84%. Now it scores a perfect 100% on the 100 examples.



This model predicts the if the rievew is positive, negative or neutral with a ~89.4% accuracy, according to this test set output score.

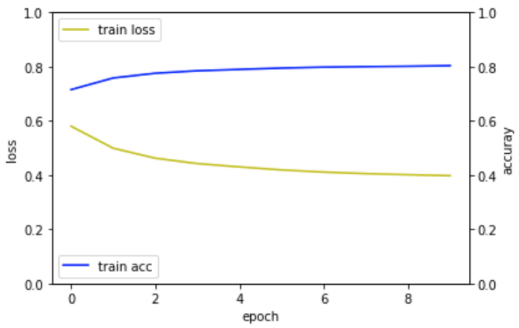
5.4 Deep Learning Model Using N-gram:

Here I am using sentiment feature for N-gram analysis. As mentioned in the project proposal I am using Keras as it gives me lots of freedom to play with hyperparameters and design a network that would be best suited for this data. I am using activation function relu, and adam optimizer [4][5].

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 200)	4000200
batch_normalization_1 (Batch	(None, 200)	800
activation_1 (Activation)	(None, 200)	0
dropout_1 (Dropout)	(None, 200)	0
dense_2 (Dense)	(None, 300)	60300
batch_normalization_2 (Batch	(None, 300)	1200
activation_2 (Activation)	(None, 300)	0
dropout_2 (Dropout)	(None, 300)	0
dense_3 (Dense)	(None, 100)	30100
dense_4 (Dense)	(None, 1)	101

Total params: 4,092,701
Trainable params: 4,091,701
Non-trainable params: 1,000

```
Epoch 1/10
142075/142075 [==============================] - 39s 272us/step - loss: 0.5802 - acc: 0.7152
Epoch 2/10
142075/142075 [==============================] - 43s 302us/step - loss: 0.4990 - acc: 0.7582
Epoch 3/10
142075/142075 [==============================] - 35s 246us/step - loss: 0.4621 - acc: 0.7758
Epoch 4/10
142075/142075 [==============================] - 42s 299us/step - loss: 0.4422 - acc: 0.7848
Epoch 5/10
142075/142075 [==============================] - 43s 301us/step - loss: 0.4300 - acc: 0.7899
Epoch 6/10
142075/142075 [==============================] - 43s 300us/step - loss: 0.4189 - acc: 0.7948
Epoch 7/10
142075/142075 [==============================] - 43s 301us/step - loss: 0.4109 - acc: 0.7983
Epoch 8/10
142075/142075 [==============================] - 43s 301us/step - loss: 0.4053 - acc: 0.8000
Epoch 9/10
142075/142075 [==============================] - 43s 305us/step - loss: 0.4013 - acc: 0.8015
Epoch 10/10
142075/142075 [==============================] - 43s 302us/step - loss: 0.3976 - acc: 0.8036
```

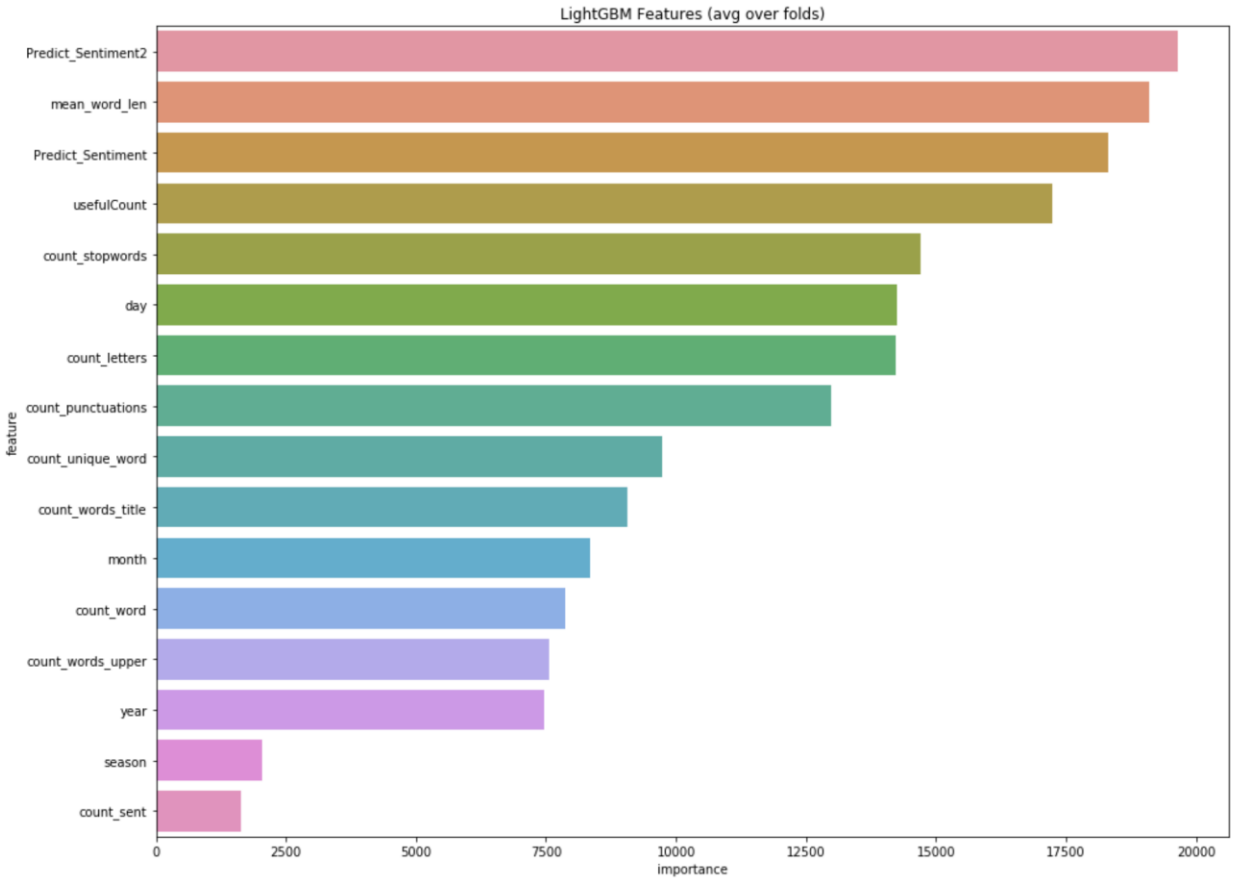



```
69978/69978 [==============================] - 12s 165us/step
loss_and_metrics : [1.0738699619418919, 0.6469318928772125]
```

From the above graph and loss matrices it can be observed that I will have to improve the low accuracy.

5.5 Lightgbm:

To improve the low accuracy, I will use machine learning. First, this is the sentiment analysis model using only usefulCount. I will add variables for higher accuracy. I added a season variable. I also normalized useful count.



5.4 Dictionary Sentiment Analysis

As the package used for prediction of 'Predict value' is formed with movie review data, it can be unsuitable for this project which analyzes reviews for drugs. To make up for this, I conducted additional emotional analysis using the Harvard emotional dictionary.

I counted the number of words in review_clean which are included in dictionary and found that there are 2006. I defined Positiv_ratio = the number of positive words / (the number of positive words+the number of negative words) If the ratio is lower than 0.5, I classified as negative and if it's higher than 0.5, I classified as positive. With remainders, I classified as neutral, which includes the sentence without either positive or negative words.

As mentioned earlier, we have normalized usefulCount by condition to solve the problem that usefulCount shows bias depending on condition. You can then add three predicted emotion values and multiply them by the normalized usefulCount to get the predicted value. The below table as the calculated final predicted value and recommend the appropriate drug for each condition according to the order of the value.

		total_pred
		mean
condition	drugName	
ADHD	Adderall	0.070960
	Adderall XR	0.042328
	Adzenys XR-ODT	0.010250
	Amantadine	0.011098
	Amphetamine	0.013925
	Amphetamine / dextroamphetamine	0.046908
	Aptensio XR	0.005885
	Armodafinil	0.028856
	Atomoxetine	0.047597
	Bupropion	0.083736
	Catapres	0.044449
	Clonidine	0.059211
	Concerta	0.059579
	Cylert	0.014713
	Daytrana	0.031764
	Desoxyn	0.133611
	Desvenlafaxine	0.006131
	Dexedrine	0.064658
	Dexmethylphenidate	0.041450
	Dextroamphetamine	0.052630
	Dextrostat	0.045610
	Dyanavel XR	0.016457
	Evekeo	0.008692
	Focalin	0.046282
	Focalin XR	0.044685
	Guanfacine	0.070408
	Intuniv	0.078066
	Kapvay	0.127024
	Lisdexamfetamine	0.045679
	Metadate CD	0.037710
...	...	...
ibromyalgia	Nuvigil	0.255291
	Prednisone	0.082950
	Pregabalin	0.131091
	Pristiq	0.076931
	Prozac	0.186982
	Savella	0.114989

6. Analysis of results:

Baseline models: Evaluation of classification models using confusion matrix and accuracy as shown below.

Vectorizer	Model	Accuracy
TF-IDF	Linear Regression	78.2%
	Naive Bayes	75.2%
	Random Forest	83.1%

Deep learning models: Evaluation of deep learning models using metrics like loss and model accuracy.

Model	Loss	Accuracy
Deep NN (Sequential Model)	0.17	89.4%
Deep Learning model using N-gram	0.39	80.1%

Deep Neural Network Sequential model gave the best accuracy for this problem of 89.4%.

7. Discussion (Limitation and Future Scope:

In conclusion, these are the limitations I had during the project and the future scope:

1. Sentiment analysis using sentiment word dictionary has low reliability when the number of positive and negative words is small. For example, if there are 0 positive words and 1 negative word, it is classified as negative. Therefore, if the number of sentiment words is 5 or less, I could exclude the observations.
2. To ensure the reliability of the predicted values, I normalized usefulCount and multiplied it to the predicted values. However, usefulCount may tend to be higher for older reviews as the number of cumulated site visitors increases. Therefore, I should have also considered time when normalizing usefulCount.
3. If the emotion is positive, the reliability should be increased to the positive side, and if it is negative, the reliability should be increased toward the negative side. However, I simply multiplied the usefulCount for reliability and did not consider this part. So I should have multiplied considering the sign of usefulCount according to different kinds of emotion.

8. Conclusion:

In summary, I aimed to extract effective inferences from our data that would benefit drug users, pharma companies and clinicians by receiving feedback of the drug based on opinion mining. I recommended top drugs for a given condition based on the sentiment score using deep learning model rating prediction. I also analyzed the emotion inclination towards a drug. I got the best predictions with Deep NN + TF-IDF model, with an accuracy of 89%, outperforming baseline models.

References

1. Sarasohn-Kahn, Jane. "The wisdom of patients: Health care meets online social media." (2008).

2. Gopalakrishnan, Vinodhini, and Chandrasekaran Ramaswamy. "Patient opinion mining to analyze drugs satisfaction using supervised learning." Journal of applied research and technology 15.4 (2017): 311-319.

3. https://archive.ics.uci.edu/ml/datasets/Drug+Review+Dataset+%28Drugs.com%29

4. Felix Gräßer, Surya Kallumadi, Hagen Malberg, and Sebastian Zaunseder. 2018. Aspect-Based Sentiment Analysis of Drug Reviews Applying Cross-Domain and Cross-Data Learning. In Proceedings of the 2018 International Conference on Digital Health (DH '18). ACM, New York, NY, USA, 121-125. DOI: [Web Link]

5. Ozsoy, Makbule Gulcin et al. “Realizing drug repositioning by adapting a recommendation system to handle the process.” BMC bioinformatics vol. 19,1 136. 12 Apr. 2018, doi:10.1186/s12859-018-2142-1

6. https://github.com/corazzon/KaggleStruggle/blob/master/word2vec-nlp-tutorial/tutorial-part-1.ipynb

7. https://stackoverflow.com/questions/28160335/plot-a-document-tfidf-2d-graph