Homework Assignment # 6

Akshay Naik (aunaik@iu.edu)

April 23rd, 2018

Project Title:

Predict Ratings from Reviews

Team Members:

Jigar Madia (jmadia@iu.edu)

Akshay Naik (aunaik@iu.edu)

Praneta Paithankar (ppaithan@iu.edu)

I. Objectives and Significance

Most of the shopping or information websites available today are equipped with Rating and Review feature which allows users to share their own experience about a certain product or service and also provide an overall rating, mostly out of 5. These ratings provide a method to estimate an overall rating of the product which can be used to provide users with a single rating to judge a product's worthiness. However the main problem with ratings these days is that they are heavily biased on the user's perspective and opinion which may not be actually true or it may be influenced by a certain feature which only miniscule percent of users consider important. For example there may be users who always insist on impeccable service and provide bad ratings for less pricey restaurants which are not service oriented and generally look out to please their customers with their food. In this case a few bad ratings can impact the overall rating of the restaurant.

Another problem arises when users generally follow a trend in their ratings. It may happen that certain users have an overall good experience but they never give out 5 star ratings and generally follow a pattern of at max 3.5 - 4 rating even if they like everything or that some users always give out good ratings even though they did not have a good experience but are generally used to giving ratings no less than 3. In such case it is essential to try and consider all these factors when we calculate overall ratings.

One solution which we feel may help here is to consider the other part of this system which allows users to provide a brief or detailed review of their experience. When yours describe it using natural language, it becomes easier to understand how their overall experience was in an unbiased manner. This is the feature we want to leverage in our project. We aim to create a system which takes users reviews and try to create a model which can accurately and in an unbiased manner predict rating for that particular review. This will not only help us provide independence to overall rating calculation but also provide a uniformity in judging all products and services on a single platform. In the next few sections we provide some details as to how we aim to achieve this solution. Our overall objective in this project is to remove any user bias which may exist in the rating using reviews.

II. Background

- **Background Information**

The general problem in depending on ratings alone is that they are not descriptive in nature. When a certain user gives a rating of 4 out of 5, he is not specifying what particular aspects of the restaurant he likes. It is possible he does not give much importance to food but to ambience and service, which can make his rating biased towards restaurants which are known just for good food. Now if the same user is to write a review, he has more space to be descriptive about his experience. It is very much likely that the user at some point mention the features he did not find good along with the features he liked and vice versa. So if someone reads his review, he is more likely to know that this user for example did not like food of this restaurant and has given a rating of 4 stars just for his experience. In the above case, a user reading reviews will get his share of information, but when it comes to overall rating of the restaurant, it has a bias of the user. If this keeps on accumulating, it can be misleading for potential future users. This is the aspect we are targeting in this project.

From our study of some papers in this topic, we have observed that one of the most popular methods of targeting this problem is using review as a source to identify user's rating and consider that for overall rating. This method exploits the descriptive nature of the review and may realize that this restaurant is only half good and should be rated 3 instead of 4 or 4.5 which is more correct in terms of overall rating of the restaurant. Most of the approaches to tackle this problem try to create a predictive model from the existing data of reviews which they feel correctly rates a restaurant and using the model tries to predict rating of each user who has provided a review. However using this method is not without its fair share of problems. Firstly it is difficult to derive a method which understands the entire context of a review and provides a rating using a predictive model. Natural Language is most of the times embedded with incorrect language usage, spelling mistakes, expressions, sarcasm or use of incorrect words which may not make any sense. To create a predictive model out of this for so many different reviews of different users is extremely difficult if not impossible. Many have even tried to just look at the overall sentiment of the review and provide a rating but it becomes difficult when the user has described a problem he faced in more detail than something which he liked which may result in the overall sentiment being negative instead of neutral. Also trying to predict sentiment is too much of a performance intensive

computation which is not possible to perform in real life for each review. Thus out of the 2 methods of predictive model and sentiment analysis, creating a model seems more viable option.

For our project we have decided to use the predictive model approach as well. We treat this problem as a classification problem where we have 5 class labels {1, 2, 3, 4 and 5} which are ratings of a review. We aim to use 3 classification techniques of SVM, XG-Boost and Neural Networks to create a model which will analyze the pool of high quality review-rating data and create a model which will predict the rating of the remaining reviews.

- **Previous Work**

We have tried to tackle this problem in past for one of our academic projects during last semester where we tried the predictive model generation method to see if we can solve this problem. We referred one of the research works carried out by Nabiha Asghar[1] and using related methods we tried creating models using Linear Regression, SVM for Regression, Random Forest and Classification and Regression Techniques (CART - ANOVA) and compared the results of each method. Unfortunately, these methods could not perform well due to the fact that data we had was very sparse and class was imbalanced. We feel that our approach was right but the tools were wrong and instead of dealing this as a Regression problem we could use Classification.

We did some more background work and realized that it may be better to treat this problem as classification problem rather than regression since our classes were discrete. We also studied Sasank Channapragada and Ruchika Shivaswamy's paper on 'Prediction of rating based on review text of Yelp reviews' [2] while studying the methodologies for tackling this problem. We found the paper on similar lines of what we plan to achieve with SVM being the common approach to this problem. They take the counts of most frequent words, most frequent stemmed words and adjectives and use them to train their SVM and Naïve Bayes model to predict ratings.

Looking at their approach we wanted to try few variations with the method of creating model and data. Firstly we wanted to try XGBoost and Neural Networks and see if we can get a better performance than SVM. Secondly we feel it may be a better approach to not use all words but rather nouns and adjectives since they are the only meaningful words in the review which may help us predict the ratings better. Considering all words does not make sense and since they are

only taking count, its possible words which are used more frequently but not relevant may make it difficult to predict rating.

What makes our work interesting is the use of methods considering the issues of the previous models. Firstly instead of taking counts of words, we are taking the TF-IDF approach of words in reviews which will give us more informative data to work on since it will not include words which are used a lot but are not meaningful when predicting ratings. Next we cover a different variety of algorithms like Tree based (XGBoost), Linear (SVM) and Deep Learning (Neural Networks). We also have processed data to tackle some of the problems which are embedded in data like high dimensionality, class imbalance, different scales which have been handled by understanding the data and experimenting with different methods.

III. Methods

- **Data Source**

The data was released by Yelp as a part of the Yelp Dataset Challenge which is organized every year. Below is the entire structure of the data released.

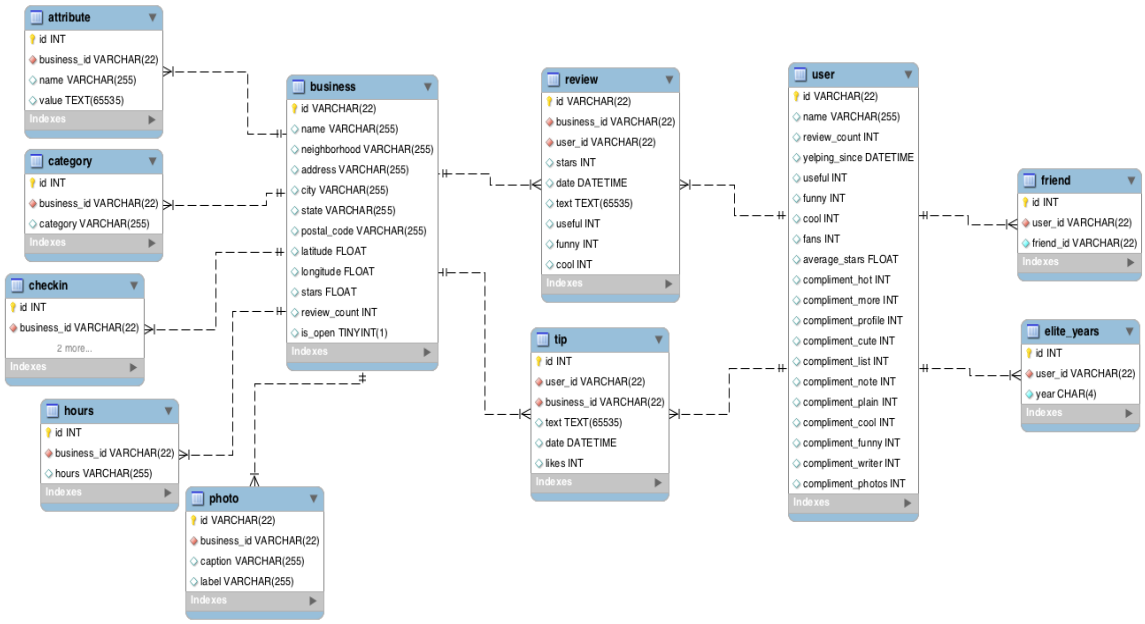


Figure 1 : Original Data Structure from Yelp [7]

The data is provided in JSON and SQL format from which we downloaded the SQL file and loaded the data in MySQL server. After loading the data we explored to see what data we can use for our project. Some of the statistics of the data is given below:

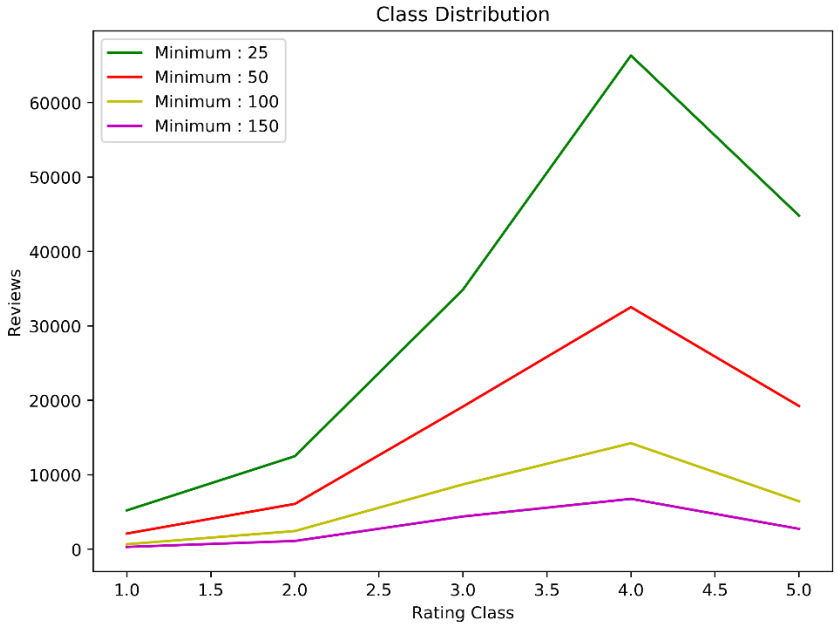
- Businesses : 174k
- Users : 1.3M
- Reviews : 5.2M
- Business of category Restaurant : 24k
- Reviews of Restaurants: 991k

From all the data we extracted the businesses which belong to the category Restaurants and then segregated reviews of those businesses using SQL queries. After this we at first thought that we will be running our project on data from only 1 city with maximum reviews which was Las Vegas in this case. But the total count of reviews for Las Vegas was just 268k out of which it was difficult to extract reviews of users with minimum count of 150. The entire base of evaluation strategy is

to train and test the data on reviews from reliable users. We believe people who are more frequent to writing reviews are better critics of restaurants and are generally more prone to writing less biased reviews. Thus we have tested our data on reviews of users with minimum of 150 reviews, 100 reviews, 50 reviews and 25 reviews. For Las Vegas, the number of reviews of users with minimum 10 reviews was also just around 30k which was too less for any the higher minimum values. Thus we decided to go for reviews from all the restaurants instead. Another problem with this data is class imbalance. Below is a table of all the class vs the minimum count of 150,100, 50 and 25.

Minimum User Reviews	**Review Count**					
	Total	**Rating 1**	**Rating 2**	**Rating 3**	**Rating 4**	**Rating 5**
1	991953	96034	80619	125195	265209	424896
50	81130	2108	6091	19164	32524	19243
100	32518	675	2444	8711	14246	6442
150	1532	309	1111	4400	6767	2740

Figure 2 : Class Distribution



The above class distribution shows how the class distribution is left skewed and due to this we may need to oversample the data before creating model. We formatted the data into a 2 column[Rating, Review] csv file for all the 4 types of minimum values and moved to our next step of converting raw data to model format of nouns and adjectives with their TF-IDF values and also a feature of Overall Sentiment of the review to guide the model in direction. The reason for including the overall sentiment is correlation of sentiment with rating score.

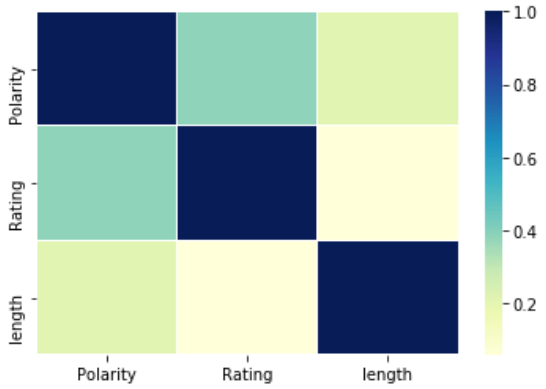


Figure 3: Heatmap of Correlation between Polarity, Review Length and Rating

If we observe the above heatmap, we can see the high correlation between a Rating of review and its Sentiment Polarity.

- **Preprocessing**

  For the purpose of learning features from the text data for predicting the class (rating) of reviews, we have to transform the 'Review' column in Document-term matrix. For this we have first created vocabulary list by considering only the top 600 words with respect to term frequencies and then used those words to transform 'Review' text in document-term matrix containing the TF-IDF score using Scikit Learn *tfidfVectorizer()*. We have also removed English stop words and converted all text to lower case while transforming the Review data.

  Once this is done, we have a sparse matrix with TF-IDF score for each word occurring in the review data. After this we have also calculated the polarity score, which is sentiment score of review, for each review using NLTK package available in python and considered this as an additional feature along with the sparse document-term matrix. This forms our pre-processed data which will be used as input by SVM classifier as well as XGBoost.

We have considered two different kinds of preprocessed data by modifying some steps before transforming Review data to Document-term matrix. They are,

- Only adjectives and Nouns from the review data as unigram features
- Unigram and bigram features of Review data by only removing the English stop words

For first type we have used only adjectives and nouns from reviews as unigram features as we feel that they will be more representative of the reviewer's sentiments and can be used to make better prediction of rating. For example, when a customer likes the ambience their review might contain something like "the ambience of the restaurant was relaxing". Here the noun and adjective together help us to understand the customer's sentiment towards the restaurant. Also, we wanted to see whether what we think about adjectives and nouns from reviews make sense in real world scenario. For second type, we have considered unigram as well as bigram features of Review data by removing stop words and created Document-Term matrix. Rest of the steps in preprocessing are same as mentioned earlier.

The final structure of the data after preprocessing is as below

Nouns + Adjectives Features — Review Sentiment — Actual Rating

	good	place	food	coffee	time	actual	job	Polarity	Rating
1									
2	0.099838	0	0.127646	0	0.193125	0	0	0.9966	5
3	0.082706	0	0	0.12046	0	0	0	0.9374	4
4	0.063538	0.139594	0	0.092542	0	0	0	0.9776	3
5	0	0.06369	0	0	0	0	0	0.9958	4
6	0	0.075771	0.088188	0	0.088951	0	0	0.9949	5
7	0.024384	0	0	0.071029	0.031445	0	0	0.9986	5
8	0	0	0	0	0	0	0	0.8884	4
9	0.067651	0.148631	0.172987	0	0	0	0	0.9847	3
10	0.125709	0	0	0	0	0	0	0.8234	4

Reviews

Figure 4: Processed Data Matrix

- **Methodology**

After completing the preprocessing part and creating the data into the above format, we can now use this to train our models. For our project we are using the below 3 methods for creating predictive models and trying to predict ratings.

Support Vector Machine (SVM): SVM is supervised machine algorithm which finds hyperplane that divides data into two classes. It is mainly used for classification and regression task. In SVM,

input space is converted into feature space and a hyperplane is found which separates the data as per margin conditions.

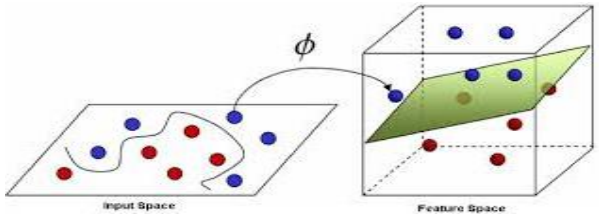


Figure 5 : SVM Concept [14]

Some of the advantages of SVM are that it is able to handle high dimensional data, memory efficient since it uses subset of training points in the support vectors and we can use different kernel functions. SVM is mainly used for text classification problem and since we are working on text classification problem, we have used SVM. While doing literature survey, we came across that SVM is used for tackling such problem by others as well and so we decided to use SVM as benchmark. We tried to use different kernel such as 'rgb' and 'linear', but as dataset is in high dimension, linear kernel performs better.

Extreme Gradient Boosting (XGBoost): XGBoost is a gradient boosting method which follows the principles of ensemble learning. It works on the concept of minimizing errors in predictive models (decision trees) by creating a new model from week models. This method of targeting the errors in previous model is what makes XGBoost a reliable method for machine learning. Since we already have a linear method of SVM, we have used XGBoost which offers decision tree concepts to see how it scales against SVM.

Neural Networks: Recently it has been seen that the state-of-art TF-IDF vectorized data doesn't perform that well in Neural Networks when compared to word embedding. So as to test this difference in performance, we will use word embedding here which is a class of methods which recently achieved fame for classification and regression task using natural language data. One of the main reason for that is word embedding produces dense representation of text data as compared to TF-IDF which produces sparse representation, this is because the vocabularies with TF-IDF method are vast and given review will be represented by a large vector comprised mostly of zero values. Also, TF-IDF which assumes bag-of words doesn't consider the semantic relation between

words which word embedding does. In an embedding, the position of word within the vector space is learned from text and is based on the words that surround the word when it is used.

We will train a type of Recurrent Neural Network called Long-short term memory (LSTM) using the real valued features obtained from word embedding. The reason to use LSTM for this task is that as the reviews can be multi-sentenced, we want to preserve long-term dependencies/information within words from different sentences which can be done using LSTM.

The model we are using is a five-layered network with embedding layer, convolution layer, max pooling layer, LSTM layer and an output layer. We have used Keras as framework to implement this model and the model was trained using Big Red 2. Neural Network wants data to be in numeric form, for that we are using Keras tokenizer to integer encode the text data. The embedding layer is initialized to random weights and will learn an embedding for all the words in the training dataset.

Some of the problems which we expected to be present in data are:

High Dimensionality - In our data matrix we have selected around 600 words of nouns and adjectives Parts of Speech which are most frequently used along with bigram combinations where the combination of most frequently noun-noun, noun-adjective or adjective-adjective forms a bigram. But due to this high dimensionality it results in increased processing time along with sparse matrix which makes creating model a difficult task. As shown below the matrix becomes very sparse at the tail end of the matrix.

scottsdale	beverage	floor	dough	cherry	sides	game	east	types	mexican	heat	cozy	wings	higher	actual	job
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0.177823	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0.069359	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.14596	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0.138871	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.18939	0
0	0	0	0.247415	0	0	0	0	0	0	0	0.118728	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.170366	0	0

Figure 6: Curse of Dimensionality

For this reason we have used the concept of Principle Component Analysis (PCA) which reduces dimensionality of the matrix. It is used to reduce the dimensionality of data to a given component size by using Singular Value Decomposition of the data to project a low dimension data matrix. We have used the method of PCA in scikit-learn to reduce the dimensionality of data. For component size we feel a matrix of size 50 is sufficient and easier to process.

Class Imbalance – It is a general trend we have observed in this data that the classes are not all perfectly balanced. The review samples of class 4 and 5 are more than class 1 and 2. Below is the distribution of classes in all the data:

1 – 96034 | 2 - 80619 | 3 - 125195 | 4 - 265209 | 5 – 424896

This is one of the main issues we have tried to solve in this project which has not been addressed before. When we tried to evaluate our models, we observed that we got better results if we did not handle the class imbalance, but it was a result of the model only predicting the class 4 (which had sufficient data in all the samples) was the most accurately predicted class and thus caused less error. But the predictions of other classes were horrible. Below is a sample confusion matrix if we do not handle imbalance.

Confusion Matrix

	1	2	3	4	5	Correct Prediction %
1	3	7	15	4	0	10.344
2	5	16	64	35	0	13.333
3	1	5	158	273	2	35.990
4	0	1	82	585	14	85.777
5	1	0	18	228	16	6.0836

Figure 7 : Class Imbalanced Results

For this reason we have tried to solve this issue by Oversampling and Undersampling the data so all classes have similar number of samples. We use the ‘imblearn’ package methods SMOTEENN and SMOTETomek to handle this issue. SMOTEENN is used to perform oversampling and Undersampling using SMOTE and cleaning using Edited Nearest Neighbors while SMOTETomek class is used to perform over-sampling using SMOTE and cleaning using Tomek links. Based on the results we have observed SMOTETomek performs much better than SMOTEENN. Even though the results are less successful with oversampling, we feel the results without handling this

issue will always be biased. If we look at the confusion matrix below we can actually see the difference in predictions across classes.

Confusion Matrix

	1	2	3	4	5	Correct Prediction %
1	16	9	3	1	0	55.172
2	29	44	24	7	16	36.666
3	38	72	132	119	78	30.068
4	21	56	113	230	262	33.724
5	6	10	21	70	156	59.315

Figure 8: Class Balanced Results

We can observe here that for class 1 and 2 we are getting majority results in 1 and 2 range like a right skewed graph. For 3 we are getting majority of 3 predictions like a normal distribution graph and for 4 and 5 we get left skewed distribution. We can see that we are getting most of the values in the correct range if we divide the reviews as 1, 2 being negative, 3 being neutral and 4 and 5 being positive.

Unscaled Data – Using unscaled data is another issue since the TF-IDF of words and polarity of review will be on different scale on the same matrix which can cause issues. Therefore we may need to somehow normalize this data. For this purpose we tried 3 methods.

Normalizer – Normalizer() is the method of normalizing data using the max value as unit and the remaining values having relative value to max value.

Robust Scaler – Robust Scaler is a method which acts as same way as Normalizer but is immune to outliers unlike Normalizer.

Quantile Transformer – Quantile Transformer is a nonlinear transformer which shrinks the distances between marginal outliers and shrinks the inliers.

- **Evaluation Strategy**

As our task of predicting the rating from review text is complex and finding exact mapping of combination of words with the rating largely depends on the quality of reviews considered for training, we are more interested in finding out how many times our models predicts incorrectly and when it predicts incorrectly what is the magnitude of incorrect prediction, as our class labels are ordered. For example, if the true label (rating) of a review is ‘4’ and if classifier A predicts ‘2’

and classifier B predicts ‘5’ then we can see that classifier B predicts somewhat better than classifier A. So, we have to consider evaluation metrics which considers such distinction.

Thus, we are considering Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Both these metrics measures the average magnitude of the errors which we are interested in.

Mean Absolute Error (MAE) - MAE measures the average magnitude of deviation between the actual and the predicted observation.

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n} |y_j - \bar{y}_j|$$

Root Mean Square Error (RMSE) - RMSE is the square root of the average of squared difference between predicted and actual observation.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \bar{y}_j)^2}$$

Both the metrics are negatively oriented score which means lower the value better is the model. One key difference between both of them is that RMSE gives relatively higher weight to large errors. This is something we are particularly interested in. We want our model to predict the rating as close to the actual rating as possible. Even MAE is does similar thing, but it doesn’t punish model as heavily as RMSE when prediction errors are large. Ultimately, it depends on the person who is evaluating as to which one to consider based on his/her liking. We are trying compare models based both of these matrices.

IV. Results

For the results below please follow the terminology of

M150B – Bigram Dataset with Minimum 150 reviews per user

M150 – Dataset with Minimum 150 reviews per user

M100 – Dataset with Minimum 100 reviews per user

M50 – Dataset with Minimum 50 reviews per user

Since there were so many variants available for XGBoost and SVM, we have tested the dataset M150 with all the preprocessing variations and the preprocessing techniques which give the best results have been used on all other datasets to find the Minimum review threshold.

XGBoost Results

PCA Applied	Normalization Method	Sampling	RMSE	MAE
No	Normalizer	None	0.80	0.61
		SMOTEENN	2.42	1.26
		SMOTTomek	1.67	0.94
	Robust Scaler	None	0.70	0.56
		SMOTEENN	1.83	1.09
		SMOTTomek	1.63	0.99
	Quantile Transformer	None	0.70	0.56
		SMOTEENN	1.99	1.13
		SMOTTomek	1.31	0.81
Yes	Normalizer	None	0.79	0.61
		SMOTEENN	2.46	1.28
		SMOTTomek	1.77	0.98
	Robust Scaler	None	0.83	0.62
		SMOTEENN	2.47	1.28
		SMOTTomek	1.35	0.85
	Quantile Transformer	None	0.81	0.61
		SMOTEENN	2.40	1.26
		SMOTTomek	1.39	0.84

Figure 9: Preprocessing Results of XGBoost with M150 dataset

We notice that out of the above results we get the best results when we don't sample but since we know that these results tend to be biased, we don't consider them. The next best result we get is RMSE = 1.31 and MAE = 0.81 is with Quantile transformer and Oversampling method

SMOTTomek without applying PCA. So we take this as threshold and continue with our experimental results on other datasets.

Dataset	RMSE	MAE
M150B	1.33	0.81
M150	1.31	0.81
M100	1.37	0.83
M50	1.48	0.86

Figure 10: XGBoost | PCA – No | Quantile Transformer | SMOTTomek

The best results are for dataset M150 with RMSE = 1.33 and MAE = 0.81.

SVM Results

PCA Applied	Normalization Method	Sampling	RMSE	MAE
No	Normalizer	None	0.74	0.59
		SMOTEENN	2.55	1.29
		SMOTTomek	2.04	1.06
	Robust Scaler	None	0.70	0.55
		SMOTEENN	2.05	1.14
		SMOTTomek	1.65	0.97
	Quantile Transformer	None	0.75	0.58
		SMOTEENN	2.38	1.25
		SMOTTomek	1.84	1.00
Yes	Normalizer	None	0.74	0.59
		SMOTEENN	2.55	1.29
		SMOTTomek	2.04	1.06
	Robust Scaler	None	0.77	0.60
		SMOTEENN	2.53	1.29
		SMOTTomek	1.71	0.98
	Quantile Transformer	None	0.74	0.59
		SMOTEENN	2.51	1.28
		SMOTTomek	1.94	1.02

Figure 11: Preprocessing Results of SVM with M150 dataset

Even in this case we get the best results when we don't oversample but since we know that these results tend to be biased, we don't consider them. The next best result we get is RMSE = 1.65 and MAE = 0.97 is with Robust transformer and Oversampling method SMOTTomek without

applying PCA. So we take this as threshold and continue with our experimental results on other datasets.

Dataset	RMSE	MAE
M150B	1.64	0.94
M150	1.65	0.97
M100	1.69	0.94
M50	1.60	0.90

Figure 12: SVM | PCA – No | Robust Transformer | SMOTTomek

The best results are for dataset M50 with RMSE = 1.60 and MAE = 0.90.

Neural Network Results

We have trained a LSTM model using Big Red 2 and fine-tuned the hyper-parameters to get the best performance. We have tried different combinations of 10,15,20,25,30 epochs with learning rate of 0.01, 0.001, 0.004, 0.0001 with decay factor 1e-04, 1e-08 and finally set the hyper-parameters as learning rate 0.001, epochs 10, decay factor 1e-04, batch size 128.

In this case we are considering class weights instead of Undersampling or Oversampling so as to avoid the issue of class imbalance. We have used sklearn *compute_class_weight()* method to assign weights to classes which analyzes the data and assigns more weightage to underrepresented classes.

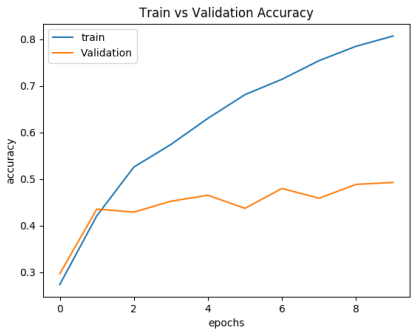
The results from LSTM using word embedding for different dataset are as mentioned below:

Dataset	RMSE	MAE	Execution time (in Hrs)
M50	0.86	0.60	~2
M100	0.75	0.58	~1
M150	0.79	0.59	~0.5

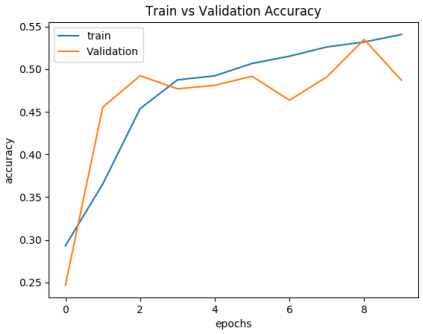
Figure 13: LSTM Results

These results are better than what we had found earlier with SVM and XGBoost. LSTM with M100 dataset performs the best with RMSE value of 0.75 and MAE of 0.58 which is quite impressive as earlier best RMSE value we obtained was 1.33 for XGBoost.

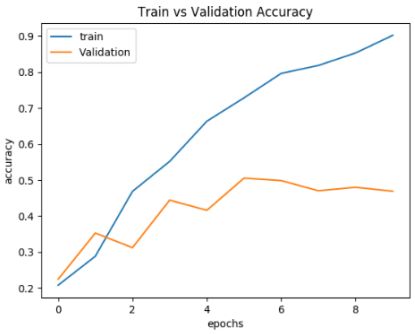
The experimental results clearly show that Neural Network is outperforming XGBoost and SVM both. We even analyzed the results of Neural Network to see how our model learned features for the three different datasets (i.e. M50, M100 and M150) by looking at train and validation accuracies at each epoch



M50



M100



M150

Looking at the above graphs, we can see that for M50 the model overfits the train data. This might be because our model is not able to learn general features for words (words present in a review) to rating mapping for this dataset as the language of reviews might be too ambiguous. For M100, we can see that the train and validation accuracies are quite similar. This implies that some general features reverent to particular class are learned properly. Even though the accuracies don't look that good, it's fairly acceptable considering the fact that learning features from natural language like English with so many ambiguities is a complex task. For M150, we can't actually infer much here as this dataset had around 15k reviews data which is quite small considering how much data is required for deep learning models to learn generalized features relevant to dataset.

By training this model we can also observe that considering reviews of only those users who have more than 100 reviews gives us better result as compared to those users who have more than 50 reviews. We cannot comment on users with more than 150 reviews as we have very few reviews for this dataset, but we feel that the higher the minimum number of reviews by a user on Yelp, the more relevant his/her reviews will be for modeling the system for predicting rating from reviews. We want more data of users who frequently posts reviews on Yelp to have a concrete proof about the statement we made above.

VII. Conclusions

From the above results we can see that LSTM model with M100 dataset gives us the best results with RMSE = 0.75 and MAE = 0.58. Let's look at the confusion-matrix to see how LSTM model predicted the rating for reviews for M100 in test dataset.

	1	2	3	4	5
1	67	28	2	1	2
2	80	162	38	10	10
3	54	296	383	161	73
4	26	93	269	704	624
5	13	22	32	196	711

Figure 14: Confusion Matrix of LSTM

Well one thing we can clearly see that for true class label as 1 and 2, the predicted class distribution is right skewed with mean as the correct class label. For true class label as 3, the predicted class distribution follows an approximate normal distribution with mean as the correct class label (i.e. 3). For class label as 3 and 4, the predicted class distribution is left skewed with mean as the correct class label. So most of the times the deviation of predicted class from the actual class is quite low. Our model gets confused mostly with the neighboring class which might have happened because of the ambiguity of language and overlapping features of neighboring classes.

This completely depends on the quality of reviews and we have no control over it. So, we cannot reduce this effect by any means and have to accept the results we are getting. This is one of the reason why working with natural language data is a complex task and we wanted to explore it.

We would conclude this project with some of the highlights of the results:

- Out of the different methods we tried, Neural Network using LSTM outperformed both XGBoost and SVM by a large margin.
- Compared to the results of the paper we studied in literature survey [3] our results outperform their best results with Random Forest with LDA and also the results which we got when we approached this topic in the previous semester.
- Some of the things we feel worked well for our method are
    - Word embedding - preserves the semantic relationship between words
    - LSTM - preserves the contextual meaning of all the sentences in the review

- Weights to Classes – helps solve the issue of class imbalance in case of LSTM
- Also for XGBoost and SVM, our assumption that Bigram model may work better proved to be incorrect and it appears that unigram models perform better.

In future we would like to see if we can get better results with more relevant users. In our project we did not have much data of users with minimum 150 reviews which was a hindrance in training models for better accuracy. It seems that with considering more reviews of better quality the approach would perform better and minimize the confusion which occurs between neighboring classes.

VIII. Individual Tasks

Initially we all discussed about what went wrong last time Jigar and I were tackling this problem and discussed about basic timeline and schedule we are going to follow. After first going through our work again we divided the initial tasks among ourselves where Praneta and I were going through similar work done before and trying to see what kind of approach they followed, at the same time Jigar was extracting the required data using SQL as he has better grasp over it compared to us. Once the data was ready, we started with initial work wherein I decided to find out different way to tackle class imbalance problem, scaling problem and which tool and framework to be considered for effective execution of the models. Meanwhile Praneta and Jigar were working on preprocessing to covert the data in Document-Term matrix form. While reading articles about the effectiveness of using TF-IDF, I read about word embedding and how it is better from TF-IDF. We wanted to compare TF-IDF with word embedding so we decided to model two algorithms using TF-IDF and use word embedding with neural network. We all took ownership of one algorithm each where I worked on Deep Neural Network (DNN), Jigar on XGBoost and Praneta on SVM.

Having earlier knowledge about working with DNN, I tried to experiment with different model and finally decided to use one network which I never learned earlier, i.e. LSTM. I read few medium blogs, read a paper on LSTM and then after gathering all information I need to start implementation, I created LSTM network and started fine tuning it on Big Red 2. Faced few issues with resource allocation as Big Red 2 provides 4GB RAM which was not enough for training LSTM on entire length of reviews, so I had to take a decision and reduce the maximum length of

reviews considered. It almost took one week to fine tune the model and we got the best performance by using LSTM as compared to SVM and XGBoost.

Since we had already worked on this topic last semester for our Search (Information Retrieval) course, we would like to note down the differences we had in this project. Last semester we had posed the same problem as a regression task and used Linear Regression, SVM for Regression, Random Forest and Classification and Regression Techniques (CART - ANOVA) methods. We were not able to get satisfiable result, some of the reason might be that we neglected few problems with data like class imbalance and considered count of words in reviews. So, we decided to tackle the same problem again by considering this problem as a classification task and by taking care of class imbalance by either oversampling or under sampling the data or by using class weights. We considered TF-IDF score instead of word count to create Document-Term matrix which was used to train SVM and XGBoost classifiers, we also tried word embedding with LSTM which was not done earlier. Some of the experiments done by us includes using only noun and adjectives from each review as features, considering bigram features, using PCA for dimensionality reduction, using different normalizing techniques to see how it changes the performance of the model.

Note: Since the datasets are huge in size, we have uploaded it on google drive.

Link: https://drive.google.com/drive/folders/1ek6N0g1M8I3bk2sOAg13PSv4R6Rq2qqS

IX. References

[1] Nabiha Asghar Yelp Dataset Challenge: Review Rating Prediction https://arxiv.org/pdf/1605.05362

[2] Sasank Channapragada, Ruchika Shivaswamy Prediction of rating based on review text of Yelp reviews https://cseweb.ucsd.edu/~jmcauley/cse255/reports/fa15/031.pdf

[3] Ojas Gupta, Sriram Ravindran, Vraj Shah Text Based Rating Prediction on Yelp Dataset Text Based Rating Prediction on Yelp Dataset https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a041.pdf

[4] Jigar Madia, Akshay Naik, Saniya Ambavanekar, Akash Sheth Yelp Dataset Challenge Project https://github.com/aunaik/Yelp-Data_Challenge/blob/master/Search%20-%20Project%20Report.pdf

[5] Mean Absolute Error https://en.wikipedia.org/wiki/Mean_absolute_error

[6] Root Mean Square Error https://en.wikipedia.org/wiki/Root-mean-square_deviation

[7] Yelp Dataset Challenge 2018 https://www.yelp.com/dataset/challenge

[8] https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

[9] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[10] https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

[11] https://github.com/EmielStoelinga/CCMLWI

[12] https://www.nltk.org/_modules/nltk/sentiment/vader.html

[13] http://xgboost.readthedocs.io/en/latest/tutorials/index.html

[14]https://www.researchgate.net/figure/Figure-A15-The-non-linear-SVM-classifier-with-the-kernel-trick_fig13_260283043

[15] https://en.wikipedia.org/wiki/Support_vector_machine

[16] https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

[17] Gradient Boosting Concept https://en.wikipedia.org/wiki/Gradient_boosting

[18] XGBoost Parameter Tuning https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/