Task 2: Predicting the star ratings from the Yelp reviews for Restaurants in Las Vegas

Chhavi Sharma[1]*,Prahasan Gadugu [2]*, Pranamya Vadlamani[3]*

Abstract
Yelp connects people to great local businesses. In this project, comprising Natural Language Processing and Text Analytics, we focus on the reviews for restaurants. We aim to predict the rating for a restaurant from previous information, such as the review text, the user's review histories, as well as the restaurant's statistic. We investigate the data set provided by Yelp Dataset Challenge and predict the star(rating) of a review. We performed feature extraction using Bag of Words and TF-IDF models and thereafter implement machine learning multiclass classifiers like Naive Bayes, Logistic Regression, Random Forests, Support Vector Machines and Convolutional Neural Networks to perform a comparative study in order to determine the most promising model to gauge the star rating of a review. Our approach of using TF-IDF alongwith Logistic Regression gave us the best accuracy of 60 percent.

Keywords
Sentiment Analysis — Feature Extraction — Classifier—Natural Language Processing — NLTK — TF-IDF — Bag-of-Words — Linear SVC — Naive Bayes — Random Forest — K-Fold — Cross Validation — Pipeline — Kaggle — Yelp — Lemmatization — Logistic Regression

[1] *Masters in Data Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[2] *Masters in Data Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[3] *Masters in Computer Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
***Course Instructor**: Xiaozhong Liu
***Associate Instructor**: Zheng Gao

Contents

Introduction

In the current social media age, online expression has become a virtual measure to rate products and services. This makes Sentiment analysis one of the most upcoming areas in the field of machine learning in order to understand and deal with this proliferation of data in the form of reviews, feedback and recommendations. Sentiment analysis deals with Natural Language processing and text analytics in order to draw two kinds of inferences out of textual data- classification on the basis of subjectivity or polarity. Subjectivity classifies whether a review is subjective or objective, i.e.if the text is manually rated and two different people do the labeling, based on their interpretation, the result of the model and its accuracy will tend to change. On the other hand, polarity refers to classifying the polarity of the data e.g. classifying whether the data is positive, negative or neutral. However, it can become daunting since human language is immensely complex. It can be a herculean task to make a machine analyze and understand grammar, sarcasm, sentence negation, slangs and other such language nuances. Advancements in natural language

processing and machine learning make it somewhat possible to analyze such huge amounts of text data and identify user opinions from them.

In a Yelp search, a star rating is arguably the first influence on a user's judgment. Located directly beneath business' names, the 5 star meter is a critical determinant of whether the user will click to find out more, or scroll on. In fact, economic research has shown that star ratings are so central to the Yelp experience that an extra half star allows restaurants to sell out 19% more frequently. Thus, we focus on the reviews for restaurants. We aim to predict the rating for a restaurant from its review text. This project report begins with the background bringing what has already been done in this field to light. Thereafter we introduce the dataset, and proceed towards the exploratory data analysis and data pre-processing techniques applied in order to gain better insight into the data. We also discuss about the various models and algorithms we investigated and implemented in order to make better predictions. In the end, we present and evaluate our accuracy results and draw comparisons among various models used.

1. Data

The dataset used for this project is taken from Yelp. Firstly, we visualized the distribution of businesses(Fig 1), as a result of which the one with most number of businesses- "Las Vegas" was chosen. Out of all the businesses, only restaurants have been considered.

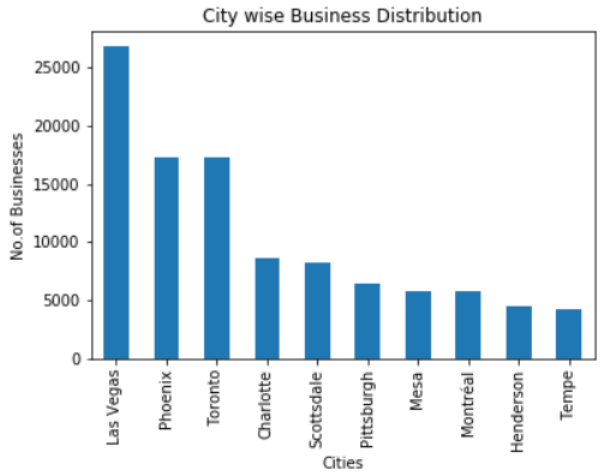


Figure 1. City wise distribution

We have taken the review text alone on which further feature extraction will be performed and used to predict the star rating given to the restaurant. The number of restaurants in Las Vegas are 5902, with an overall review count of 929,636. Since we are trying to predict the user rating based on the review, the labels here are the star ratings.

We assumed the star ratings to convey the following polarities:

1. 1- most negative
2. 2-somewhat negative
3. 3-neutral
4. 4-somewhat positive
5. 5-most positive

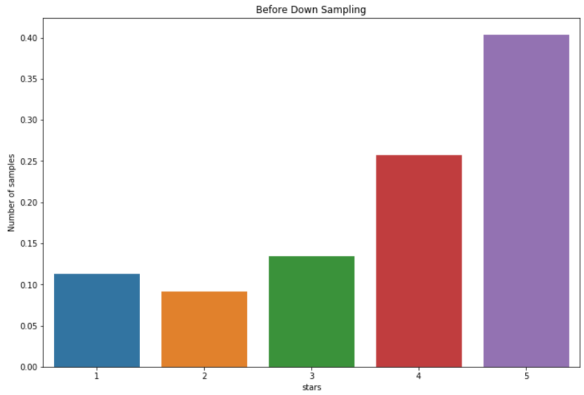
We observed that the data we chose is imbalanced since around 50 percent of the samples attribute to 5 i.e., five star rating(most positive). Therefore, we will have to balance our data before the classification.

2. Exploratory Data Analysis

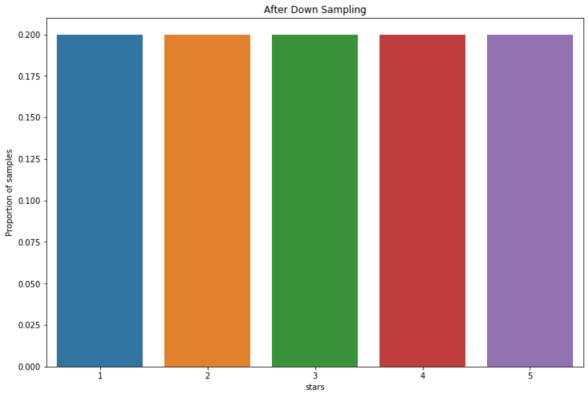
We need to analyse and pre-process the data in order to prepare it for running classification models. This comes under exploratory data analysis.

2.1 Handling the Imbalanced Data

Firstly, on analysing the given data, we found that the train data is biased towards sentiment 2 i.e. neutral(Refer histogram below) since almost 50 percent of the whole data evaluates to 'neutral'.



Therefore, we down-sampled the majority class, re-sampled each rating based on reviews into equal samples of 50K. The distribution after down-sampling is shown below,

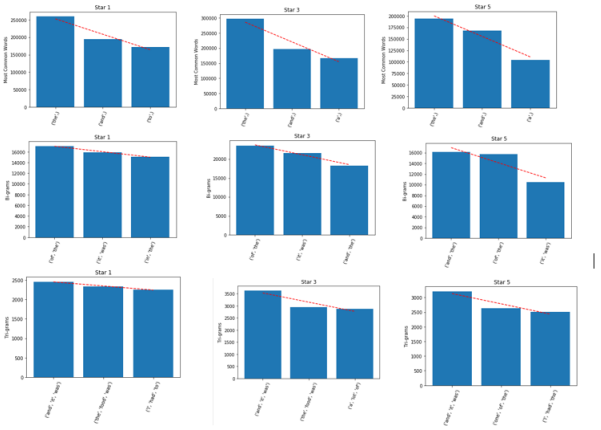


After re-sampling, the data reduced to 250K reviews with each rating having 50K reviews associated with it.

2.2 Data Exploration

Furthermore, in order to gain better insight into the training data, we performed the following steps to analyze the data and calculate the average word length of reviews in the corpus which came out to be 122.8.

Later, we performed a pre-cleaning analysis of data by plotting the most common words, bi-grams and tri-grams rating-wise and came up with the following plots,



We notice that all of them contain a lot of stop words like 'and','the', 'a' etc. which may lead to prediction of false rating for the review since similar words have high counts in more than one rating. Therefore, we need to clean the data in order to get better results.

2.3 Data Cleaning

In order to resolve these issues, we performed the following pre-processing steps on the data:

1. Lemmatization- Firstly, we grouped together all inflexions of a word in order to analyse them as a single item during class prediction.

2. Changed the entire corpus to lowercase in order to make our predictions case-insensitive by not differentiating between the same word in different cases.

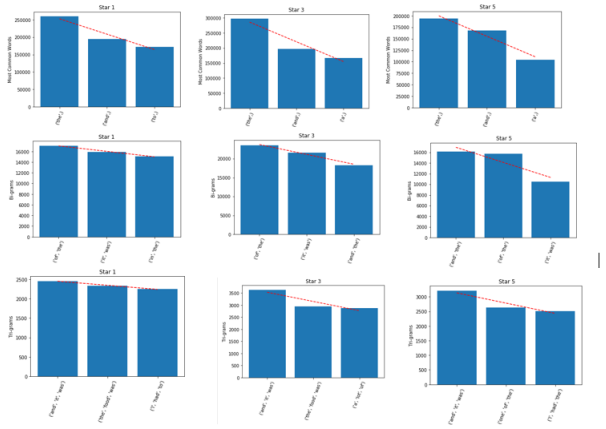
3. Removed all stop words like prepositions, articles, conjunctions etc. since they do not add much value to our predictions and are fairly common to all reviews.

4. Used Regex formatting to remove punctuation marks full-stop, comma etc. since they are useless towards predicting the class label. However, we retained the exclamation mark since it conveys a strong emotion in the review text.

5. Broke the grammatical structure of the data in order to get a corpus of words that can be used as our feature set.

6. Removed null values encountered due to data cleaning (removal of stop words, punctuation etc).

After completing the data cleaning, we again tried to extract the 30 most common words, bi-grams and tri-grams for each rating. We found a much better, concise and useful list of words this time.



As we can see, the words such as food, place and service are prominent. In the positive space, the tri-gram "definetly coming back" is trending in ratings 4 and 5, whereas bi-gram "great service" is trending. On the contrary, in the negative space, the tri-gram "worst service ever" can be frequently found.

2.4 Word Cloud

The rating of a review can be predicted by exploring the words that are most likely to draw the review towards a particular rating. To analyze this, we can visualize the data by creating a word cloud using the word cloud library in python. It will give us some insight into the words and their frequencies in a review for a particular rating. We created the word clouds for the clean text data in order to avoid all stop words and focus on the unique root words.

Figure 2. All words

Figure 3. Positive Sentiment: Ratings 4 and 5

Figure 4. Negative Sentiment: Ratings 1 and 2

2.5 Splitting the data

For further model building and evaluation of the information, we divided our dataset into training and testing sets in the ratio 80:20.

3. Feature Extraction

For extracting meaningful reviews in order to train our classifier models, we used the following techniques:

3.1 Bag of Words

This is a text mining technique that builds a predefined set of words and calculates the word count for each word. This word count matrix is then used to extract different feature representations. We have a huge amount of data owing to which the number of words in the review text will be huge. Therefore, in order to get better results, we will consider the 10 topmost frequent words as our feature set and train our models on these features.

3.2 Tf-Idf (Term Frequency Inverse Document Frequency)

Tf-Idf (Term Frequency Inverse Document Frequency) It is a text mining technique used to categorize documents. It emphasizes on words that occur frequently in a particular review, while at the same time de-emphasizing words that occur frequently in every review. This method will help us concentrate on portions which might be less frequent but have more significance for the overall polarity of the review. To limit the number of features getting created, we put the max_features parameter as 60,000 for this dataset.

4. Conventional Machine Learning Approach

4.1 Baseline Models

The basic aim of this project is to test different models to attribute a star rating for restaurant reviews. To achieve this, we started with testing several baseline multiclass classification algorithms on the bag-of-words model to find out which model is working better. Also, we need to test whether we are getting better accuracies for the original dataset or the downsampled dataset. The performance evaluation is done on the validation set. We used Sklearn implementation of the classifiers for this project.

4.1.1 Naive Bayes

It works on the principle of Bayes' theorem with the assumption of conditional independence between every feature(word or group of words). We used Multinomial Naive Bayes since we have 5 classes in total.
Accuracy: 38.5%

4.1.2 Logistic Regression

It is a predictive classifier in which there are one or more variables determining the outcome. The probabilities describing the possible outcomes of a single trial follows a sigmoid curve.
Accuracy: 39.2%

4.1.3 Random Forests

Random Forests uses the bagging approach with max-voting among the available decision trees
Accuracy: 33.6%

4.1.4 Support Vector Machines

It is a supervised learning algorithm which designates a hyperplane in an n-dimensional space where each data instance is plotted that efficiently differentiates each class. We used LinearSVC that finds the best fit hyperplane to categorize the data. Thereafter, the features can be fed into the model to find the class it belongs to.
Accuracy: 39%

4.1.5 Summary of Baseline Models

If the test accuracy of the baseline models on Bag-of-words features is to be compared, we observe that Logistic Regression gives us the best accuracy(40%) followed by Linear SVC, Naive Bayes and Random Forests. However, in reality all of them perform badly on that test data, considering the best accuracy is a mere 40%.

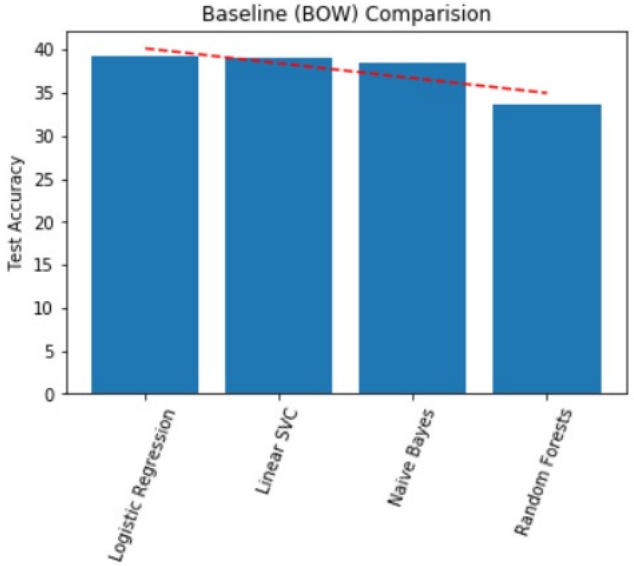


Figure 5. Baseline Models Accuracy Summary

Since this project deals with sentiment analysis of textual

data, we realized that bag of words concentrates more on word count rather than the logical construct of data. It is quite possible that a word may occur less frequently but is clearly indicative of a particular class. To overcome this shortcoming, we used TF-IDF model for feature extraction in the subsequent models for better accuracy.

4.2 Pipeline Models

Pipeline sequentially applies a list of transforms to estimate the final class. For better accuracy, based on the baseline models, we extracted features using TFIDF, hypertuned the model parameters and used pipeline over these models to get better results. We started by applying hyper parameter tuning for TF-IDF vectorizer by supplying a range of maximum features(max_features) and built the pipelines for the respective models.

We constructed confusion matrices summarizing the correct and incorrect predictions for each class. Models implemented:

4.2.1 Logistic Regression using TF-IDF

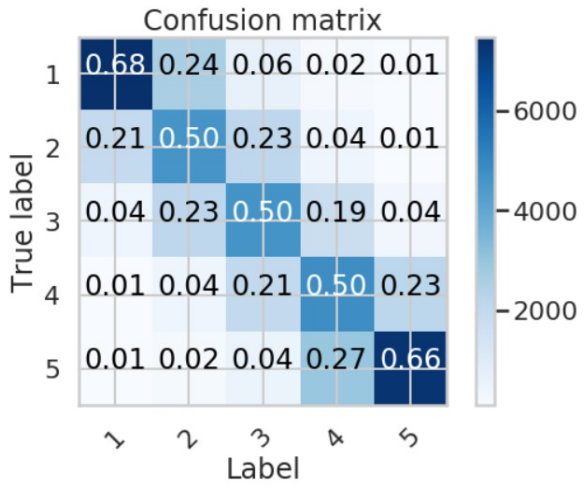


Figure 6. Confusion Matrix for Logistic Regression

As per the confusion matrix for logistic regression, the model predicted label 1 and label 5 well, however it performed mediocre for the other sentiments.

	precision	recall	f1-score	support
1	0.77	0.69	0.73	11388
2	0.47	0.54	0.50	8674
3	0.50	0.54	0.52	9148
4	0.51	0.53	0.52	9546
5	0.76	0.67	0.71	11236
accuracy			0.60	49992
macro avg	0.60	0.59	0.60	49992
weighted avg	0.62	0.60	0.61	49992

50000 Train score 0.623267188093857
50000 Test Score 0.6020163226116179

Figure 7. Results for Logistic Regression

The test accuracy was around 60%, therefore, we tried training our data by pipelining TFIDF with Multinomial Naive Bayes to check if we get better results.

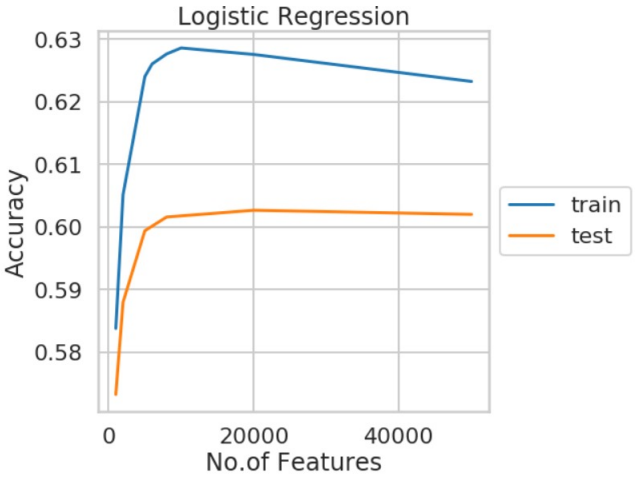


Figure 8. Logistic Regression Graph

4.2.2 Multinomial Naive Bayes using TF-IDF

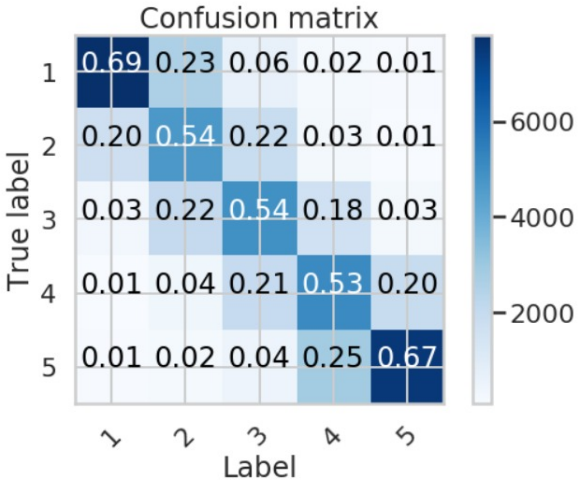


Figure 9. Confusion Matrix for Multinomial Naive Bayes

Again, this model predicted the sentiment classes better as compared to logistic regression, however, its test accuracy dropped to a little less than 58%. So, we went on to experiment the learning using Random Forest to check if it provided better results.

	precision	recall	f1-score	support
1	0.72	0.65	0.69	11134
2	0.44	0.50	0.47	8849
3	0.49	0.50	0.50	9712
4	0.52	0.52	0.52	10042
5	0.71	0.69	0.70	10255
accuracy			0.58	49992
macro avg	0.58	0.57	0.58	49992
weighted avg	0.59	0.58	0.58	49992

```
50000 Train score 0.6492368626352744
50000 Test Score  0.5786325812129941
```

Figure 10. Results for Multinomial Naive Bayes

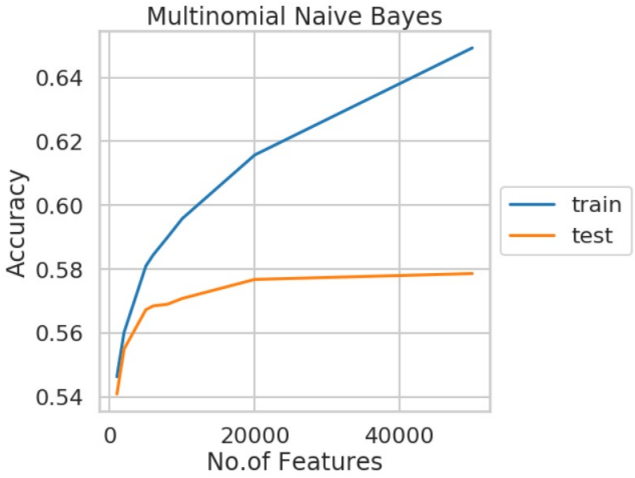


Figure 11. Multinomial Naive Bayes Graph

4.2.3 Random Forest

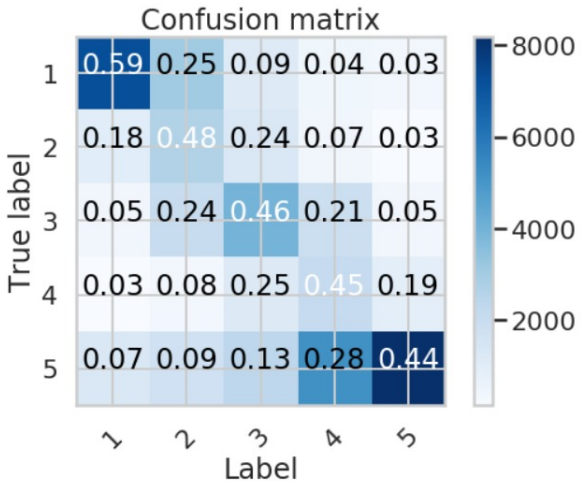


Figure 12. Confusion Matrix for Random Forest

By analyzing the confusion matrix for this model, we found that its classification accuracy for each class drops a little.

	precision	recall	f1-score	support
1	0.72	0.58	0.64	12666
2	0.26	0.48	0.34	5401
3	0.39	0.46	0.42	8590
4	0.23	0.45	0.30	5183
5	0.80	0.44	0.57	18152
accuracy			0.48	49992
macro avg	0.48	0.48	0.46	49992
weighted avg	0.60	0.48	0.51	49992

```
8000 Train score 0.48264187553759674
8000 Test Score  0.4830372859657545
```

Figure 13. Results for Random Forest

Its performance on both train and test sets also drops further down to about 48%.

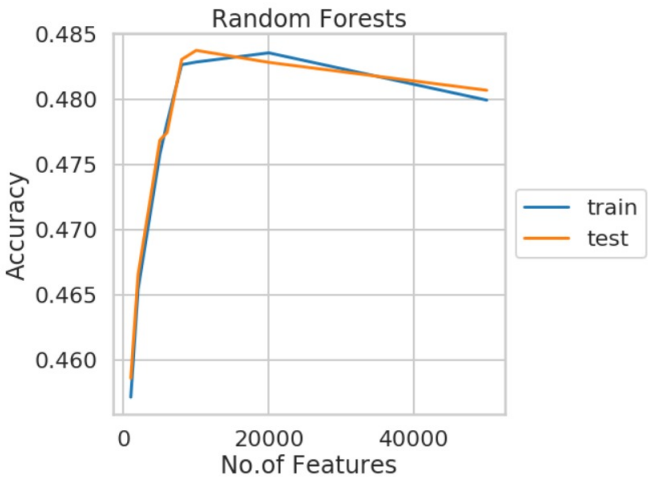


Figure 14. Random Forest Graph

Last but not the lease, we proceed to apply Linear support vector classifier in the hope to get better accuracy.

4.2.4 Linear Support Vector Classification using TF-IDF

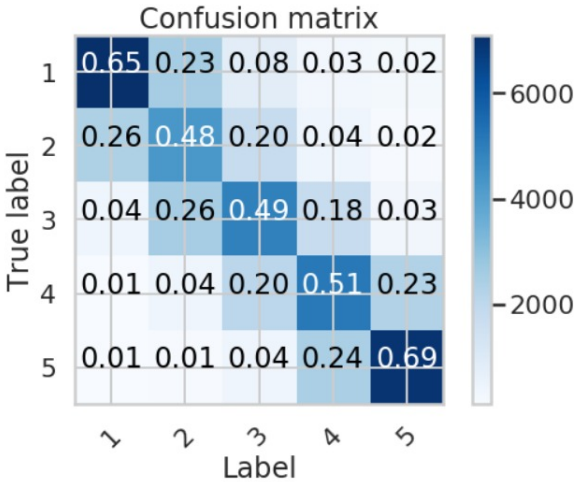


Figure 15. Confusion Matrix for Linear Support Vector Classifier

By analyzing the confusion matrix for this model, we found that it performs better than the other two models in terms of identifying labels for the train set.

	precision	recall	f1-score	support
1	0.77	0.68	0.72	11434
2	0.46	0.52	0.49	8828
3	0.47	0.52	0.49	8881
4	0.49	0.52	0.50	9378
5	0.76	0.66	0.71	11471
accuracy			0.59	49992
macro avg	0.59	0.58	0.58	49992
weighted avg	0.61	0.59	0.60	49992

6000 Train score 0.643165769838571
6000 Test Score 0.5892742838854217

Figure 16. Results for Linear Support Vector Classifier

However, it clearly overfits because the graphs for train and test sets diverge in opposite directions.

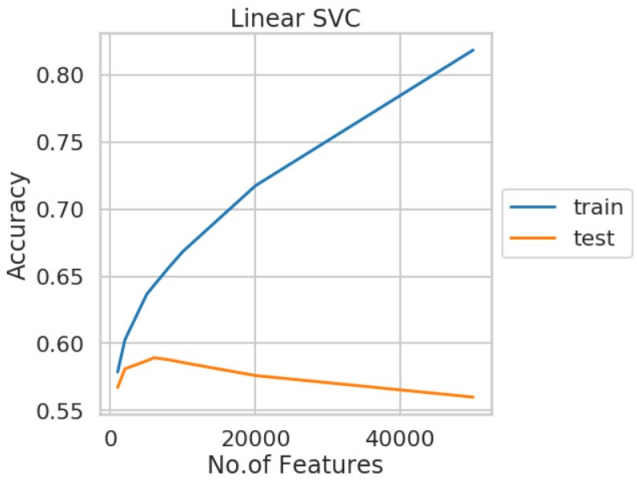


Figure 17. Linear Support Vector Classifier Graph

4.2.5 Summary of TF-IDF Pipeline Models

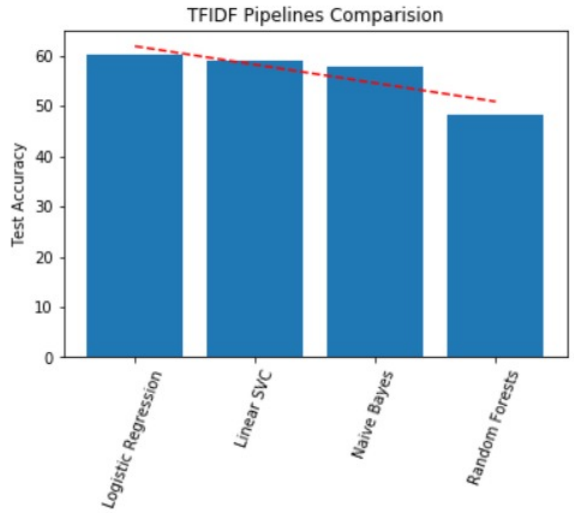


Figure 18. Test Accuracy comparison for TF-IDF pipeline models

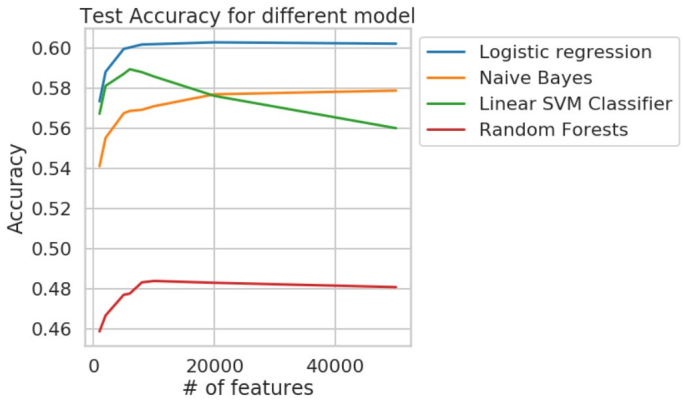


Figure 19. Test Accuracy VS Number of features for TF-IDF pipeline models

On analyzing the graphs, we can conclude that similar to Bag-of-words baseline models, Logistic regression gave us the best accuracy(60%) on the test set followed by Linear Support Vector Classifier, Naive Bayes and Random Forest. However, the accuracy for the models, in general and Logistic Regression, in particular(60%) achieved much higher accuracy when compared to baseline models using Bag-of-words feature extraction.

5. Deep Learning Approach

To apply deep learning techniques, we pre-processed the data even further by tokenizing the corpus and post padding. Thereafter, the train and test labels were integer encoded and then one hot encoded.

5.1 CNN:

Convolution neural networks are generally used in the image classification due to its channel-wise approach to the problems. For CNN, generally, CNN is used for Image Classification owing to its channel-wise approach. However, since this particular dataset presents itself as a text classification problem, single-channel CNN can be used.

Layer (type)	Output Shape	Param #
activation_1 (Activation)	(None, 3655)	0
embedding_1 (Embedding)	(None, 3655, 100)	8105600
conv1d_1 (Conv1D)	(None, 3653, 100)	30100
dropout_1 (Dropout)	(None, 3653, 100)	0
max_pooling1d_1 (MaxPooling1	(None, 1217, 100)	0
flatten_1 (Flatten)	(None, 121700)	0
dense_1 (Dense)	(None, 10)	1217010
dense_2 (Dense)	(None, 5)	55

Figure 20. CNN Architecture

We used an embedding layer initially on which a Convolutional 1D layer with Relu Activation was used with 100 filters

and kernel size 3. This generates a single dimensional matrix as an output on which maxpooling 1D can be used to pick the desired values. We added a dense layer with 10 hidden units and an output layer with 5 hidden units and Softmax activation. Since it is a multi-class classification problem, it has multiple output nodes for which 'Categorical cross entropy' function was used as the loss function. Adam optimizer has been used for optimization and "accuracy" metric has been used for evaluation of the results.

5.2 CNN with Word Embeddings

When the input to a neural network contains symbolic categorical features (e.g. features that take one of k distinct symbols, such as words from a closed vocabulary), it is common to associate each possible feature value (i.e., each word in the vocabulary) with a d-dimensional vector for some d. These vectors are then considered parameters of the model and are trained jointly with the other parameters.

	precision	recall	f1-score
star 1	0.73	0.66	0.69
star 2	0.47	0.58	0.52
star 3	0.48	0.45	0.46
star 4	0.51	0.41	0.45
star 5	0.65	0.74	0.69
accuracy			0.57

Figure 21. CNN + Word Embeddings

Using the above architecture, we achieved an accuracy of about 58 percent on the test set. A comparison of accuracy on the train and test set can be seen in (Figure 22).

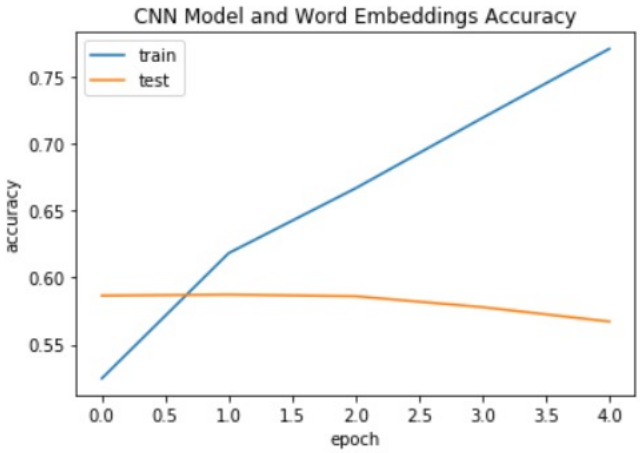


Figure 22. CNN + Word Embeddings Accuracy

5.3 CNN with GloVe

GloVe or Global Vectors is an unsupervised learning algorithm developed by Stanford for generating word embeddings by aggregating global word-word co-occurrence matrix from a corpus. This results in a linear substructure relationships between words.

	precision	recall	f1-score
star 1	0.72	0.68	0.70
star 2	0.48	0.56	0.51
star 3	0.51	0.43	0.47
star 4	0.50	0.55	0.52
star 5	0.70	0.66	0.68
accuracy			0.57

Figure 23. CNN + GloVe

Using the same CNN architecture specified above, but with the GloVe Feature extraction, we got an accuracy of 57 percent on the test set. A comparison of accuracy on the train and test set can be seen in (Figure 24).

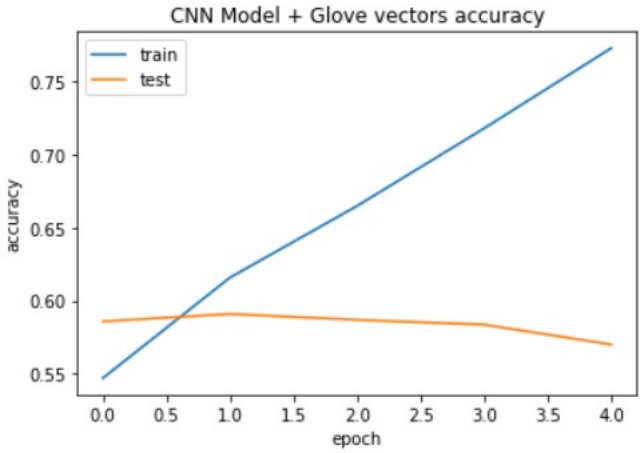


Figure 24. CNN + GloVe Accuracy

6. Summary

In a nutshell, the accuracy scores for each model over down-sampled data over all Machine Learning and deep learning approaches can be summarized in (Figure 25). Therefore, Linear SVC performed better with just 6000 features delivering an accuracy of 59 % compared to Naïve Bayes Model. However, Logistic Regression with 20000 features gave an accuracy of 60.2 % which is the highest. CNN with Word Embeddings performed better than the one with GloVe feature extraction method. To sum up, the simple Logistic Regression proved to be the best method for classifying with an accuracy of 60.02 percent.

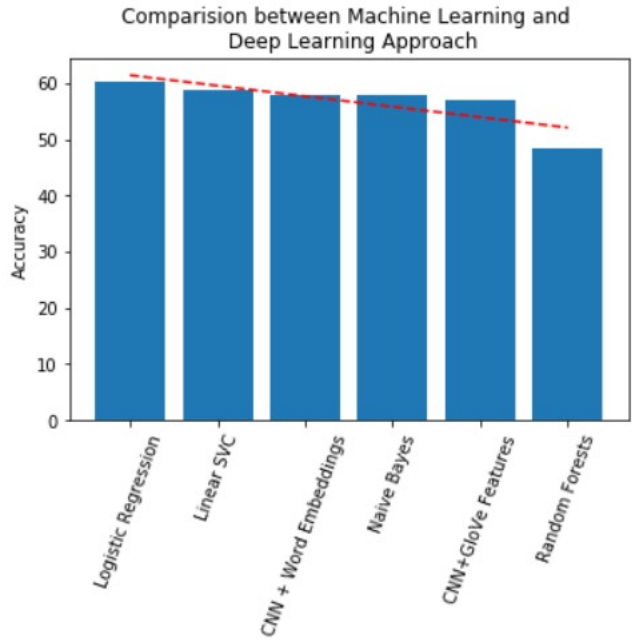


Figure 25. Model Comparison

7. Conclusion

Regarding this project, our tasks of learning the challenges associated with sentiment analysis and trying to resolve them by trying different features selection and classifier models have been met. However, we could try n-grams if we had better memory and it could have provided us with better results. Additionally, trying other feature extraction techniques such as Random Forests feature importance module or neural networks like Long short-term memory models could increase the performance over prediction of star ratings from restaurant reviews.

Contributions

Pranamya Vadlamani: Data Cleaning and Baseline Models

Chhavi Sharma: TF-IDF Pipeline models

Prahasan Gadugu: Deep Learning Approach

References

1. https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/

2. https://www.kaggle.com/adamschroeder/countvectorizer-tfidfvectorizer-predict-comments

3. https://www.kaggle.com/ynouri/rotten-tomatoes-sentiment-analysis

4. https://www.kaggle.com/artgor/movie-review-sentiment-analysis-eda-and-models EMNLP 2002, 79–86. http://www.cs.cornell.edu/home/llee/papers/sentiment.pdf

5. Rennie, J., Shih, S., Teevan, J., Karger, D. Tacking the Poor Assumptions of Naive Bayes Text Classifiers. In ICML-2003.

6. Siolas, G., d'Alche-Buc F. Support Vector Machines Based on a Semantic Kernel for Text Categorization. In IJCNN 2000. https://nlp.stanford.edu/projects/glove/

7. https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/

8. https://medium.com/coinmonks/solving-twitter-sentiment-analysis-problem-on-analytics-vidhya-ea3e51eea521