CAPSTONE PROJECT

Machine learning Engineer Nanodegree **Adityi Jain**

Yelp reviews rating prediction **March 30, 2018**

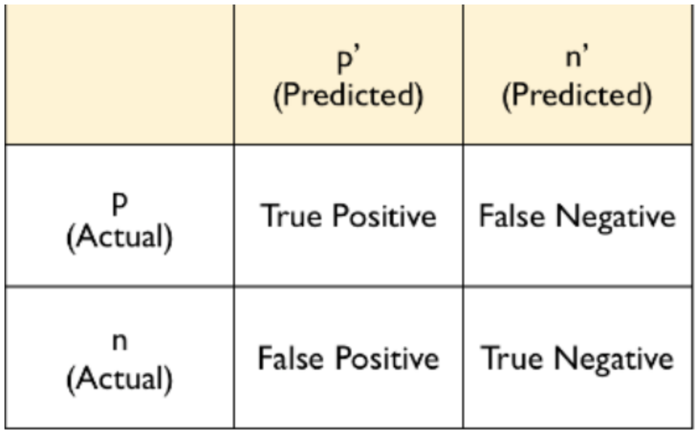
DEFINITION:

Project Overview: Due to advancement in technology and cheap availability of the same to almost everyone, the dependency of people on internet is very high, and for every suggestion or problem, we are highly used to refer to internet. This varies from asking the solution of the problem, to check the routes, to check the timings of all sort of things happening in the world and to check whether a place is good or not. YELP is similar site, where we can give and check the reviews and ratings of all types of things ranging from hotels, restaurants, Salons, doctors, food, entertainment and other type of nightlife. If someone is new to a city or a country itself (like me), then to get an idea about all above things is very important. But there are incidents when ratings and reviews don't go along, and this may cause ambiguity. Hence to resolve this issue, there should be an algorithm which can read a review (since it is not possible for a person to go through thousands of reviews) and can determine the ratings based on all those reviews, and my motive of doing this project is to solve the above issue.

Problem Statement: When the data of yelp for restaurants was checked, there were around 5.2 million reviews and related ratings. For a person, it's impossible to go through all the reviews, and the ratings could be highly misleading, for example a customer can give rating of 2/5 because food was excellent and ambience was perfect but there was no WIFI and discount, while other can also give 2/5 because the restaurant had high speed WIFI with some discounts, but food was terrible. Both these ratings are not relevant, and hence to solve this issue, we can use machine learning which will go through the text, extract important features relevant for the particular domain (food for this case) and averages the ratings based on above.

The problem is related to ratings for the restaurant, which is a discrete value with labels from 1-5, hence this is multiclass classification. The input (features) for the algorithms will be words which will be extracted from the review and output will be ratings from 1-5. Since there is a huge data available, hence machine learning will perfectly help to solve this problem. The data is updated everyday, and the same algorithm can be used for the latest ratings. There are many sites with ratings and reviews like Amazon, Zomato, Grub Hub, TripAdvisor etc, and the same algorithm with minor changes can be used for all such reviews systems.

Metrics: The F1 score will be used for the evaluation of the model. In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true. The F1 score works as: F1 = 2*precision*recall/(precision + recall)



	p' (Predicted)	n' (Predicted)
p (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

It is the harmonic mean for precision and recall where precision is number of correct positive results divided by the number of all positive results returned by the classifier (true positive / true positive false positive) while recall is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive, true positive/ true positive + false negative). The score was used with a average of 'weighted': Calculate metrics for each label, and find their average, weighted by support (the number of true instances for each label).

ANALYSIS:

Data exploration: The data has a total of 5261668 reviews, and there are total 9 columns associated with each review. The columns are as:

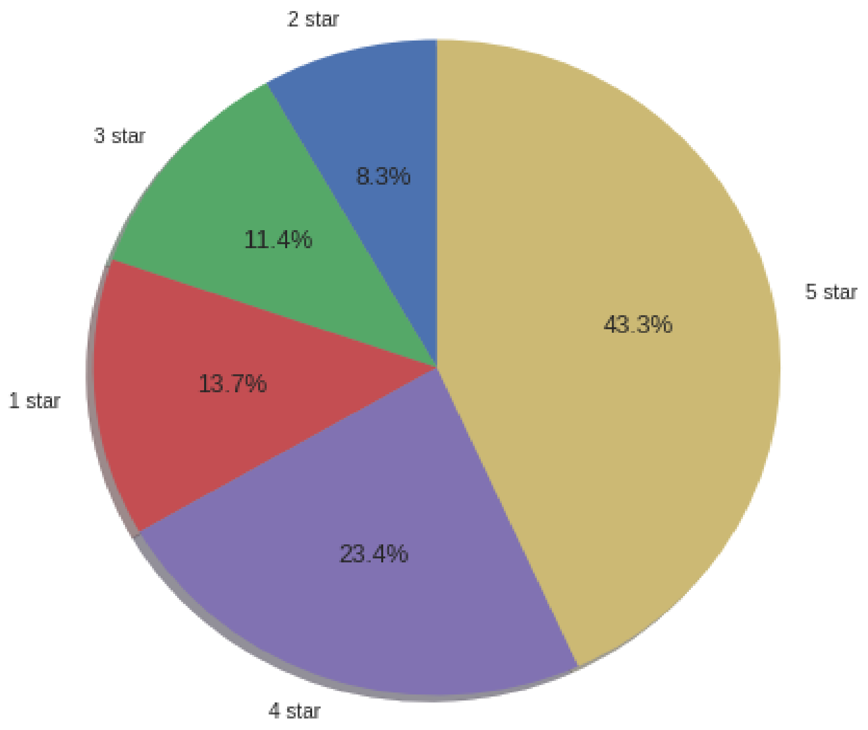
- Review_id
- User_id
- Business_id
- **Stars:** The Ratings given by the user
- Date: Date when the rating was given
- **Text:** This is the review given by customer
- Useful
- Funny
- Cool

Each review had a detailed description/ experience shared by the customer as text, and stars which ranged from 1 to 5. This two fields were relevant for this project and was considered for using in the project. Now since this all data has almost 5.2 million reviews, this was a huge data to be handled, hence 200,000 reviews were used for all algorithms except for Support vector machines, where 25000 reviews were used due to complexity of this algorithm and super slow running.

The data was taken from Kaggle (https://www.kaggle.com/yelp-dataset/yelp-dataset/data).

Exploratory visualization: When the distribution of the reviews was seen, it was noticed that the data as highly skewed, with higher number of reviews for 4 and 5 stars, while the number of reviews for 1 and 2 stars was very less. It can be seen as follows:

Star	Count
5	86501
4	46788
1	27375
3	22779
2	16557



It can be noticed that 5 stars and 4 stars have very high number (86501 and 46788) respectively and contribute to a total of 67% of the total reviews, while reviews for 1 and 2 stars are relatively less, i.e only almost 21%.

Algorithms and techniques: The algorithms used are as:

- Multinomial Naive Bayes: Naive Bayes Classifier algorithm performs well when the input variables are categorical. It classifier converges faster, requiring relatively little training data than other discriminative models , and the best is that it works well with multiclass predictions.
- Logistic Regression: They are very easy to perform and less complex, and don't take a huge time for training the data. Robust algorithm as the independent variables need not have equal variance or normal distribution, algorithms do not assume a linear relationship between the dependent and independent variables and hence can also handle non-linear effects.
- Grid search: For the next three algorithms, grid search Cv was used for tuning of the hyperparameters.

- Decision trees: In this I have used classification trees, which are considered as the default kind of decision trees used to separate a dataset into different classes, based on the response variable. These are generally used when the response variable is categorical in nature.
- Random forest: It uses bagging approach from subset of data to make a decision tree, several such trees are formed and then average is used for the final classification. They can be used on large datasets, have less tendency to overfit, and have high accuracy.
- Support vector machines: SVm are effectively used for text classification, but the problem is that when the data is huge, the SVM is very slow, Hence, for SVM 25000 reviews were used, preprocessing was done for these reviews and then algorithm was run.

Benchmark model: The accuracy in Yelp Dataset Challenge: Review Rating Prediction, by Nabiha Asghar was:

- 54% with Logistic Regression
- 56% for SVM.

The benchmark follows accuracy score, but for my model, I will be evaluating using F1 score. Both the above scores mentioned were with both Unigram and Bigram.

METHODOLOGY:

Data preprocessing: The data preprocessing involved following steps:

- A function (StemmedTfidfVectorizer) for stemming was written based on snowball stemmer using TF-IDF (term frequency–inverse document frequency) Vectorizer.
- Before performing the stemming and vectorization, the data was split into training and test data, in the ratio of 75:25. For this train_test_split was used from sklearn.
- For both training and testing, there were two datasets, X (which was text or review, from which the features are to be extracted) and y (corresponding rating).
- After splitting the data, features extraction was done from the training set with the help of StemmedTfidfVectorizer, using “english” as stop words. There were two types of feature

extraction done: Unigram and Unigram+Bigram. It was done using Fit_transform of TFIDF vectorizer.

- In unigram, single words were taken as features, while in Unigram+Bigram single words along with the pair of words were taken as features.
- For test data, transform was done to extract features (no fit was performed for the test data).

After completing the whole procedure, it was found that the number of tokens (features) for unigram was 81363 while the number of features for both unigram+bigram was 2849324.

Implementation: For naives bayes and logistic regression, the execution was direct. The algorithms were trained for both unigram and bigrams and then the ratings were predicted for the test data. The F1 score (rounded to 5th place of decimal) for both the models are as:

- Unigram Naive bayes: 0.43847
- Unigram + Bigram naive bayes: 0.28646
- Unigram Logistic Regression: 0.63806
- Unigram + Bigram Logistic Regression: 0.64862

For the next 3 algorithms, grid search was used for tuning the parameters.

Decision tree: The parameters for decision trees were tuned as:

{'max_depth': [10, 20, 40, 50],

'min_samples_leaf': [5,10,15],

'min_samples_split': [5,10,15]}

The classifier was used with class_weight="balanced". The tuning was done by fitting 3 folds for each of 36 candidates, totalling 108 fits. After the grid search, the best parameters were as:

{'max_depth': 40,

'min_samples_leaf': 5,

'min_samples_split': 5}.

With the help of these parameters, the model was trained with Unigram and Unigram + Bigram. The F1 score was as:

- Unigram: 0.45932
- Unigram + Bigram: 0.45826

Random forest classifier: The parameters tuned for random forest were as:

{'max_depth': [40, 60],

'n_estimators': [200, 400]}

For this algorithm also, class_weight was set to balanced. The tuning was done by fitting 3 folds for each of 4 candidates, totalling 12 fits. After the search, the best parameters were as:

{'max_depth': 40,

'n_estimators': 400}.

These parameters were used for training and predicting, and the resulted scores were as:

- Unigram: 0.57645
- Unigram + Bigram: 0.55981

Support vector machines: The support vector machines is a very exhaustive algorithm, and hence with the available setup, it was running endlessly with 200,000 reviews. Hence for SVM, 25,000 reviews were taken, they were preprocessed as for all above algorithms and then grid search was performed for them. The number of Unigram tokens for SVM were 29766 and Unigram + Bigram 644047. The parameters used for training were:

{'C': [1,10,50],

'kernel': ['linear','rbf'],

"gamma":[0.01, 0.1, 1]}

The tuning was done by fitting 3 folds for each of 18 candidates, totalling 54 fits. The best parameters were as:

{'C': 1,

'gamma': 1,

'kernel': 'rbf'}

With the help of these parameters, the model was trained with Unigram and Unigram + Bigram. The F1 score was as:

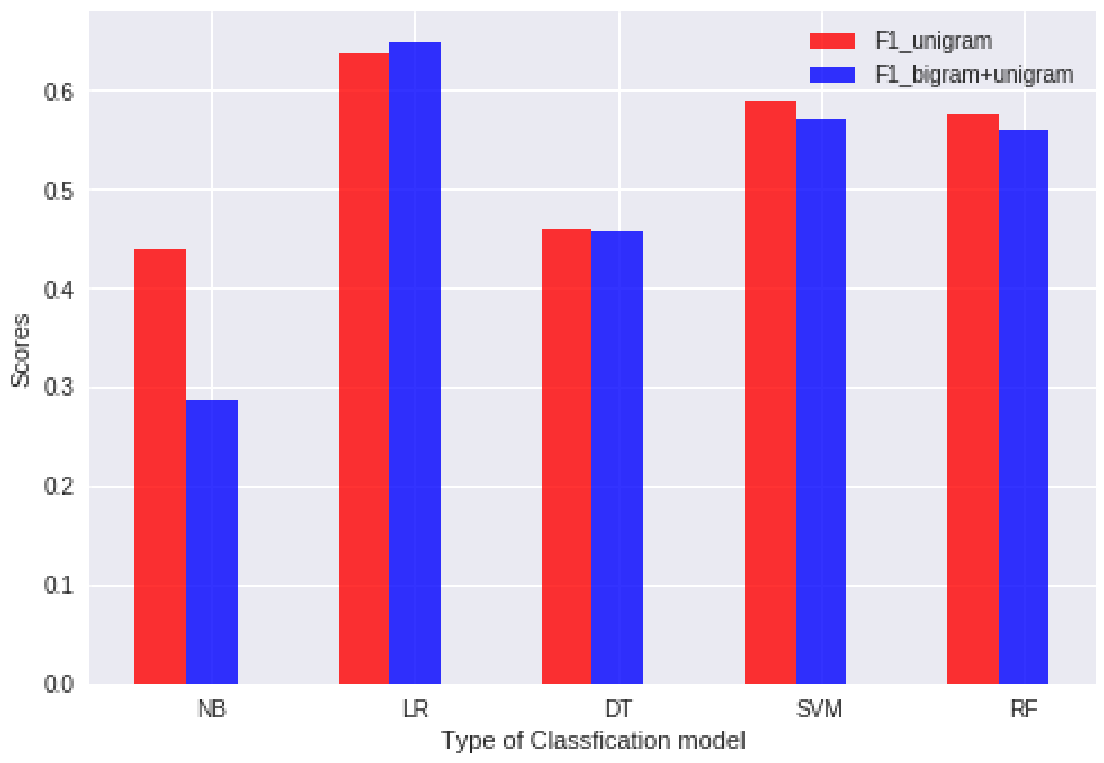
- Unigram: 0.58945
- Unigram + Bigram: 0.57215

RESULTS:

Model evaluation and Validation:

It can be seen that the highest score is for logistic regression (unigram+monogram), with the score of 0.64862, i.e 64.9%, this is followed by logistic regression only with 0.63806, 63.8%. Apart from logistic regression, the models which performed well were Support vector machine with unigram score of 58.9% followed by random forest with 0.57645/ 57.6% for unigram.

Type	**Naives bayes**	**Logistic Regression**	**Decision trees**	**Random forest**	**Support vector Machines**
Unigram	**0.43847**	**0.63806**	**0.45932**	**0.57645**	**0.58945**
Unigram + Bigram	**0.28646**	**0.64862**	**0.45826**	**0.55981**	**0.57215**



The benchmark model selected was from Yelp Dataset Challenge: Review Rating Prediction, from Nabiha Asghar. The highest accuracy from this model was 56% with Linear SVC and for logistic regression, the scores were 54% (accuracy score). For my model, the highest score (F1) was 64.9% for logistic regression (unigram+bigram) and 58.9% for SVM, unigram.

If the features are to be compared, in benchmark model the numbers of tokens for unigram were 171,846 while for Unigram + Bigram, they were 7,612,422.

In my case, they were as 81363 and 2849324.

Justification: The project was run on Google colab, due to its limitation, the data used was 200,000 for all models except for SVM, where 25,000 was used. The number of reviews used for the benchmark model was 706,646. The model is performing well when the accuracy is considered, but when all the dataset is used, which has around 5.2millions reviews, then the tendency of model to perform better might be increased.

CONCLUSION:

The problem which was taken was relevant. This type of Classification could be used for other datasets also like Amazon for customer reviews, IMDB for movies reviews etc. The ratings were working accurately with the 64.9%, and if whole data set is used, then there is a possibility for increased accuracy.

Learnings: I learnt a lot while doing this project. I learnt how to visualise the dataset, like in this dataset, the distribution of reviews was not equal for all classes, which means this was imbalanced, hence this was a new concept. I learnt that not all scores are equally applicable for classification problem, that is accuracy score, RMSE score can not be used for unbalanced class, while F1/ AUC score are good for such classification problems.

Best thing was to use stemming, features extraction, unigram and bigram concept. This was completely new concept, and I got a little bit idea of natural language processing too. The use of TF-IDF was very interesting, and I would like to dig more in all these concepts.

The tuning of parameters using grid search was very helpful, though I used this technique in one of the project in supervised learning, but this was in more depth. For tuning the parameters, I had to go through the details of each parameter and its importance, which helped to gain a in depth understanding of the algorithm.

Overall, I learnt alot, and my enthusiasm to go in machine learning has increased significantly (I am from pharma background, and this was a amazing experience for me).

Limitations and improvements: The only thing hindering full use of dataset was the machine which I have (ASUS, 8GB). This lead me to use google colab, but it also has its own limitations due to which I could not use complete dataset.

For utilizing all the algorithms thoroughly, complete dataset needs to be used. The tuning of all the hyperparameters can be done with lot of more details, with many different kind of combinations. Further for feature extraction, more methods like lemmatization, trigrams etc. can be tried. The data can be modified so as to reduce noise. This all can overall lead to a more potent model with significantly high accuracy.

References:

1. Prediction of Useful Reviews on Yelp Dataset
   https://webcache.googleusercontent.com/search?q=cache:JA_716wjevcJ:https://bcourses.berkeley.edu/files/

2. Classifying Yelp reviews into relevant categories
   http://www.ics.uci.edu/~vpsaini/

3. Yelp Dataset Challenge: Review Rating Prediction
   https://arxiv.org/abs/1605.05362

4. Predicting yelp restaurant reviews
   https://cseweb.ucsd.edu/~jmcauley/cse255/reports/wi15/Wa'el_Farhan.pdf