**Rating Prediction based on User Reviews**

****

**Intro to Data Mining Project Report**

**Team Members:**

* **Aniketh Sukhtankar (UF ID 7819 9584)**
* **Harika Bukkapatanam (UF ID 3683 6895)**
* **Shikha Mehta (UF ID 4851 9256)**
* **Suyog Daga (UF ID 5773 5749)**
* **Raheen Mazgaonkar (UF ID 4714 4316)**
* **Rameshwari Obla (UF ID 1616 1302)**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **1.** | Abstract | 3 |
| **2.** | Introduction | 3 |
| **3.** | Prior Work | 4 |
| **4.** | Data Collection | 5 |
| **5.** | Dataset Preparation | 7 |
| **6.** | Algorithms | 8 |
| **7.** | Experimental Results | 14 |
| **8.** | Conclusion | 19 |
| **9.** | Limitation | 20 |
| **10.** | Future Improvements | 21 |

1. **Abstract**

Review websites, such as TripAdvisor and Yelp, allow users to post online reviews for various businesses, products and services, and have been recently shown to have a considerable influence on consumer shopping behavior. An online review typically consists of free-form text and a star rating out of 5. The problem of predicting a user's star rating for a product, given the user's text review for that product, is called Review Rating Prediction and has lately become a popular, albeit hard, problem in machine learning. In this project, we implement an approach which involves a combination of topic modeling and sentiment analysis to achieve this objective by treating Review Rating Prediction as a multi-class classification problem, and building different prediction models by using Latent Dirichlet Allocation as the underlying feature extraction method with three machine learning algorithms, (i) K Nearest Neighbors, (ii) Multinomial Naive Bayes and (iii) Random Forest. We analyze the performance of each of these models to come up with the best model for predicting the ratings from reviews. We use the dataset provided by Yelp for training and testing the models.

1. **Introduction**

User reviews are an integral part of web services like TripAdvisor, Amazon, Epinions and Yelp, where users can post their opinions about businesses, products and services through reviews consisting of free-form text and a numeric star rating, usually out of 5. These online reviews function as the `online word-of-mouth' and a criterion for consumers to choose between similar products. Studies show that they have a significant impact on consumer purchase decisions as well as on product sales and business revenues. On famous websites like Amazon and Yelp, many products and businesses receive tens or hundreds of reviews, making it impossible for readers to read all of them. Generally, readers prefer to look at the star ratings only and ignore the text. However, the relationship between the text and the rating is not obvious. Several questions may be asked: why exactly did this reviewer give the restaurant 3/5 stars? In addition to the quality of food, variety, size and service time, what other features of the restaurant did the user implicitly consider, and what was the relative importance given to each of them? How does this relationship change if we consider a different user's rating and text review? The process of predicting this relationship for a generic user (but for a specific product/business) is called Review Rating Prediction. Concretely, given the set S = {(r1; s1)… (rN; sN) for a product P, where ri is the i'th user's text review of P and si is the i'th user's numeric rating for P, the goal is to learn the best mapping from a word vector r to a numeric rating s. Review Rating Prediction is a useful problem to solve, because it can help us decide whether it is enough to look at the star ratings of a product and ignore its textual reviews. Moreover, some review websites allow users to write text reviews without specifying a star rating. In these cases, Review Rating Prediction comes in handy. However, it is a hard problem because two users who give a product the same rating, may have very different reasons for doing so. User A may give a restaurant 2/5 stars because it does not have free WIFI and free parking, even though the food is good. User B may give the same restaurant a rating of 2/5 because he does not care about the wifi and parking, and thinks that the food is below average. Therefore, the main challenge in building a good predictor is to effectively extract useful features of the product from the text reviews and to then quantify their relative importance with respect to the rating.

In this project, we implement an approach which involves a combination of topic modeling and sentiment analysis to achieve this objective by treating Review Rating Prediction as a multi-class classification problem, and building different prediction models by using Latent Dirichlet Allocation as the underlying feature extraction method with three machine learning algorithms, (i) K Nearest Neighbors, (ii) Multinomial Naive Bayes and (iii) Random Forest. We analyze the performance of each of these models to come up with the best model for predicting the ratings from reviews. We use the dataset provided by Yelp for training and testing the models.

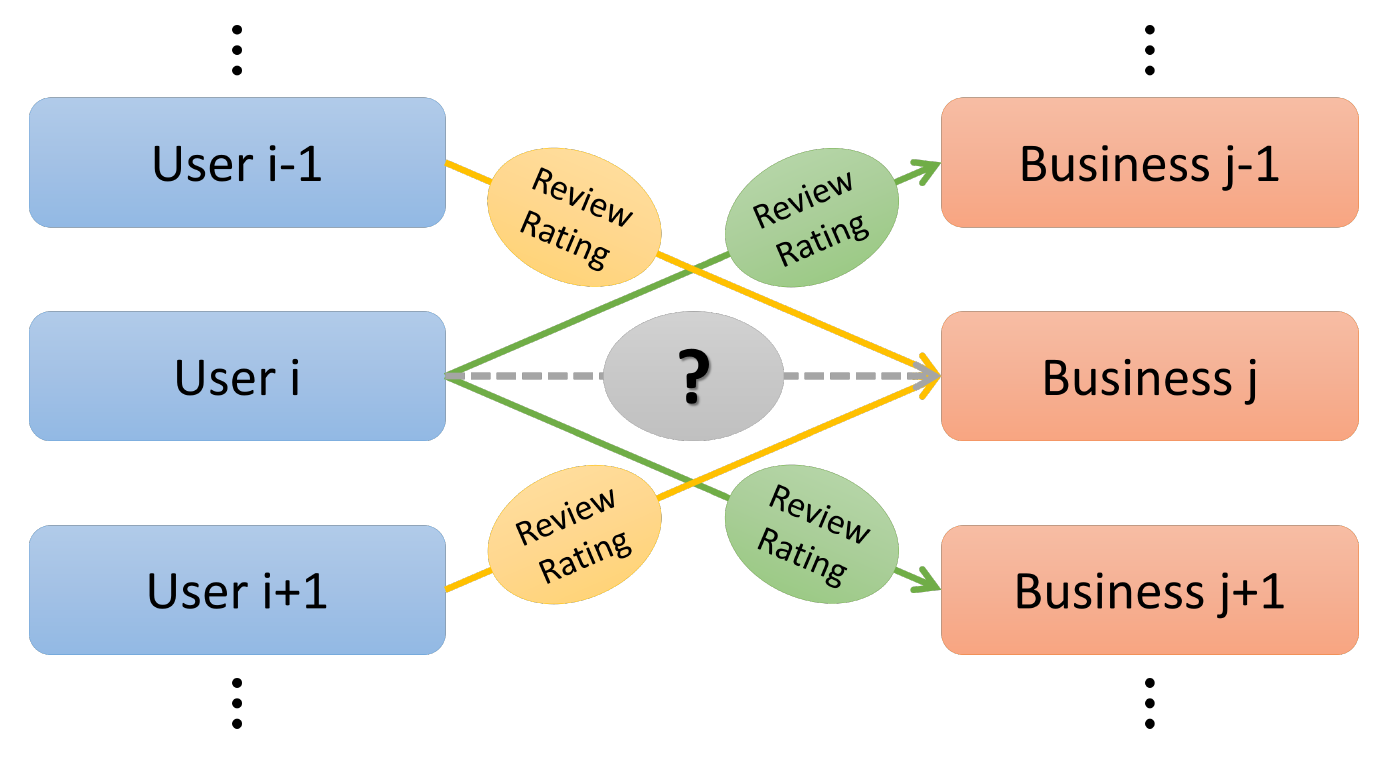
1. **Prior Work**

The key aspect of our project involves topic modeling and we rely on a state-of-the-art technique to achieve this task. The technique we used is Latent Dirichlet Allocation (LDA) proposed by Blei in 2003. In Blei et al, the author proposes a novel approach for topic modeling which improves upon the previous state-of-the-art technique called Probabilistic Latent Semantic Indexing (PLSI). The topics are assumed to be drawn out of a Dirichlet distribution and it solves the problem of over-fitting faced by PLSI. There also has been significant research in sentiment analysis. A successful application of machine learning techniques to perform sentiment analysis can be observed in Pang et al. In our project we used the Naive Bayes classifier to determine the sentiment.

1. **Data Collection**

The dataset used for this task was obtained from the Yelp dataset challenge, which consists of 4.7M reviews and 1,000,000 tips by 1,100,000 users for 156K businesses. It has 1.2M business attributes such as hours, parking availability and ambience. This data is in JSON format.

The dataset consists of five files, business, review, user, check-in and tip. Business and review files are primarily used for this predictive task. The business data file is a json file that comprises of attributes of each business listed on Yelp, and the review file consists of those of a review. The text of a review is often overlooked in such predictive tasks in favor of features such as the user's and business' previous rating history. However, if the sentiment of the text of a user's review can be estimated suitably, it would be the best indicator of the rating. Ultimately, a review is what the opinion of the user is about the business in his own words and not a mathematical predictive task. Thus, it is essential to be able to predict what the user feels about a business from the review text and this is why the task of rating prediction from review text was chosen by our team.

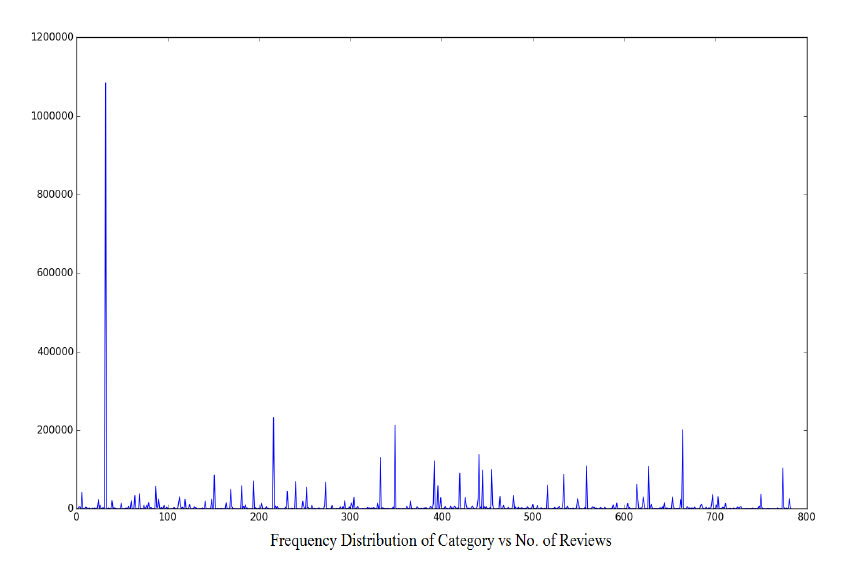


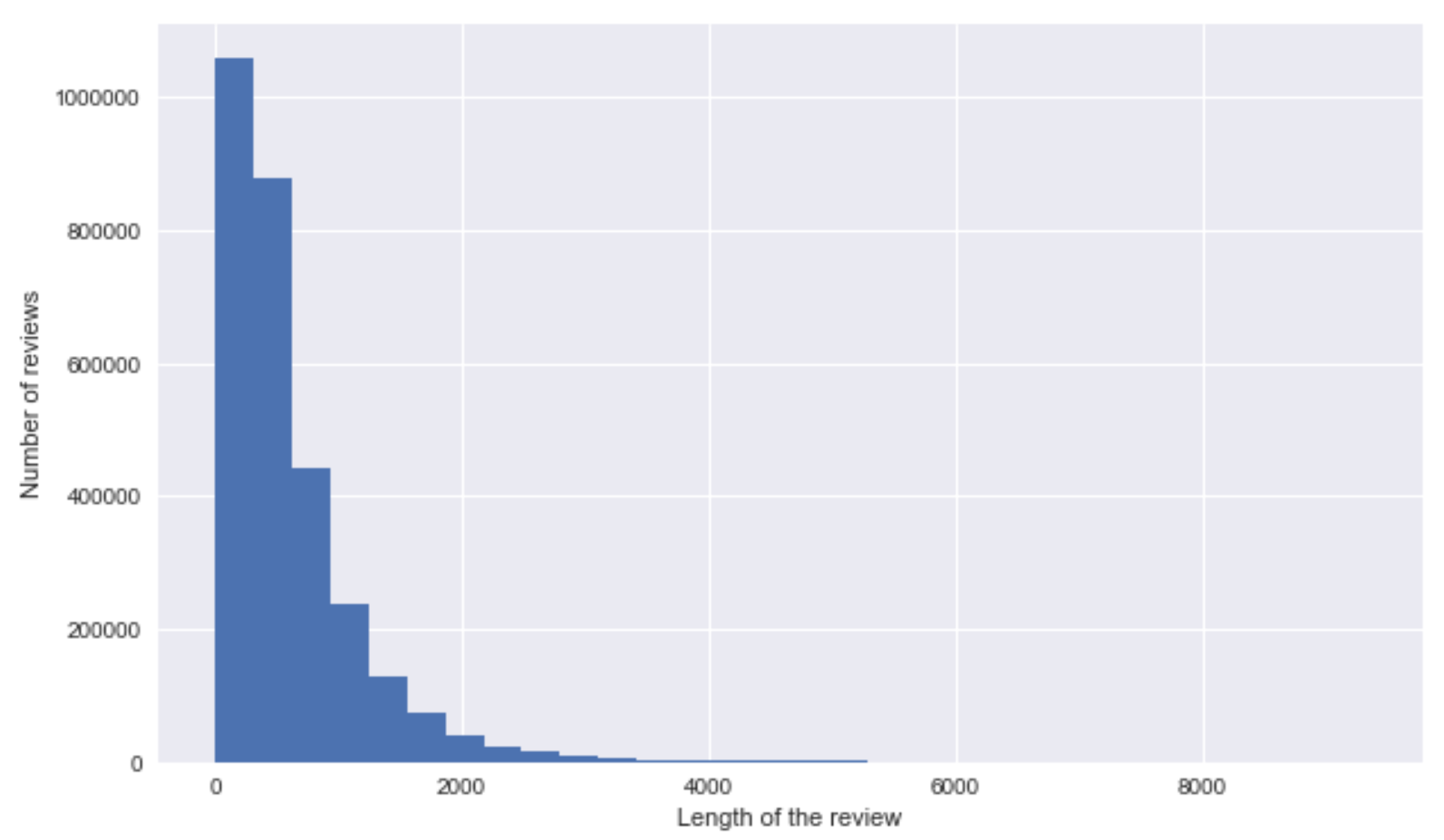
User & Business Connections

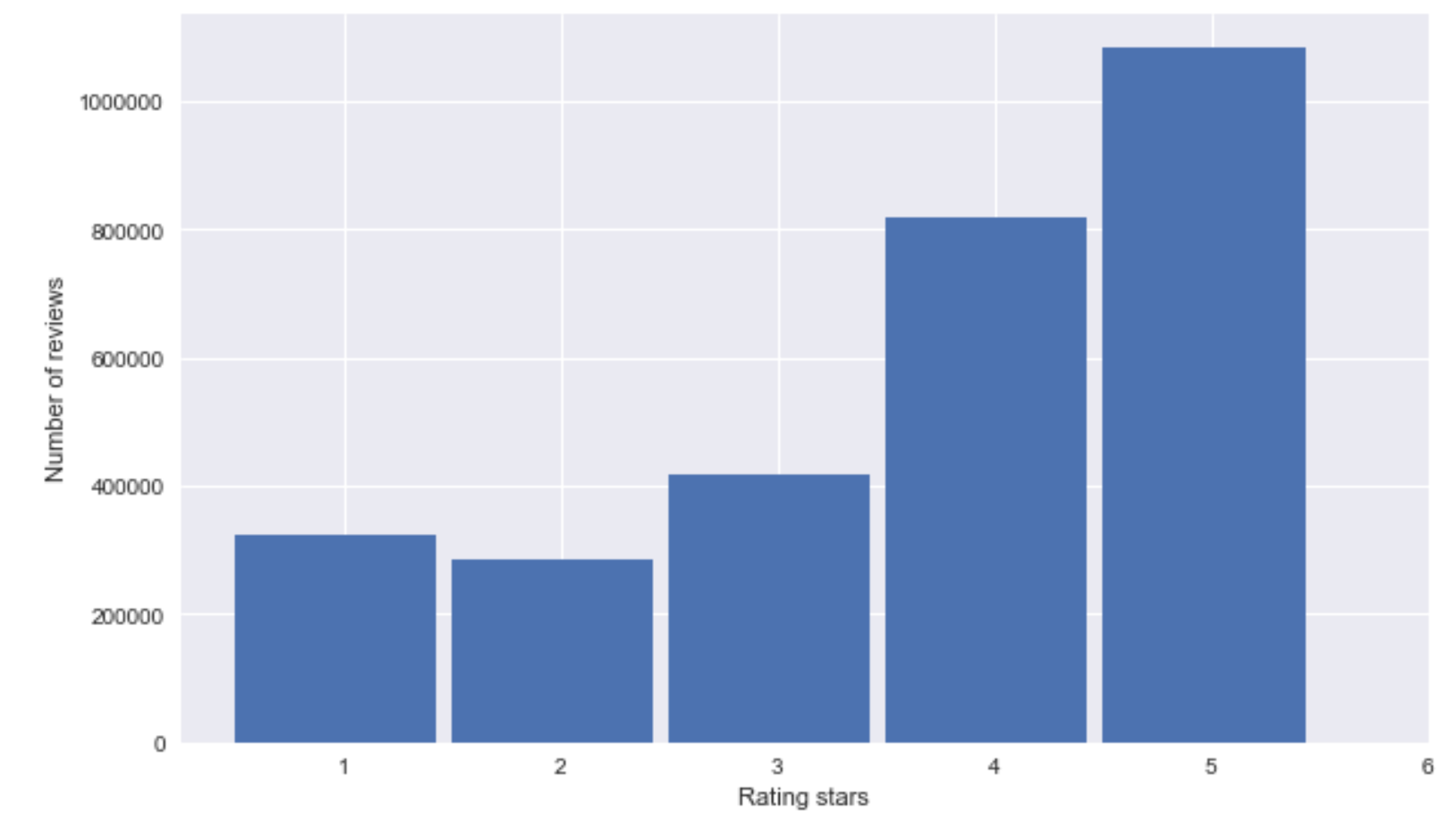
The dataset generated is 19.2 GB in size and contains the following information:

|  |  |
| --- | --- |
| **Name** | **Attributes** |
| Business | Business Name, Id, Category, Location, etc. |
| User | Name, Review Count, Friends, Votes, etc. |
| Review | Date, Business, Stars, Text, etc. |

From the plot of categories vs the number of reviews for each category as shown below, it was observed that the number of reviews for the category 'Restaurants' (69.05% of the total data), was by far the highest. Since impact of a word on rating will vary drastically across categories, considering all categories in the same text mining model would lead to unsuitable results. It was therefore, concluded that considering reviews of only Restaurants would result in a more accurate model. It should be noted that length of the most reviews is within 2000. Also, most of the ratings are 4 or 5 stars as seen below.







1. **Dataset Preparation**

Before we can began working on the classification algorithms, it is important to prepare our data. The following section outline two ways in which we achieved this: by preprocessing the data and by extracting meaningful features from the data.

Preprocessing

We first write some basic Python scripts to separate the restaurants from the business.json file, and to separate the restaurant reviews from the review.json file. We then preprocess the text reviews as follows. Yelp allows users to write text reviews in free form. This means that a user may excessively use capital letters and punctuation marks (to express his/her intense dislike, for example) and slang words within a review. Therefore, it is necessary to preprocess the reviews in order to extract meaningful content from each of them. To do this, we use standard Python libraries to remove capitalizations and punctuations. Stopwords, that is, words with no information value that appear too frequently in a language, are also removed according to a list from nltk.corpus.

Further to assess the model’s performance, the data set was divided into two parts: a training set and a test set. The first was used to train the system, while the second is used to evaluate the learned or trained system. According to the requirements the data set was split into two disjoint sets where 80% of the original data set was used as the training set, while the 20% that remains composed the test set.

Feature Extraction

As a basis for next step, we proceeded to model the topics of the review text using Latent Dirichlet Allocation(LDA). Using LDA we fit 15 latent topics from the review text in our training set. Some of the key words identified by the Latent Dirichlet Allocation model of sklearn are as shown below:

(1) Topic 2: bar beer drink selection drinks enjoy great place house tried We can observe that Topic 28 talks about pubs and bar’s

(2) Topic 8: lunch salad sandwich menu items options day tried tasty definitely. This topic models reviews pertaining to restaurants with lunch options.

(3) Topic 12: amazing love fish portions place tried taste great service did. This topic models general positive qualities in the reviews.

(4) Topic 13: vegas sushi atmosphere special definitely place worth roll best come. This topic seems to model the reviews of sushi/Japanese restaurants.

1. **Algorithms**

**K Nearest Neighbors Algorithm**

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor. Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.

Implementation:

Parameters:

In our algorithm, k=5 nearest neighbours are considered for class assignment.

Model Fitting:

We modeled the training examples as vectors in a multidimensional feature space with a class label. The training phase of our algorithm consists only of storing the feature vectors and class labels of the training samples.

Sample Class Prediction:

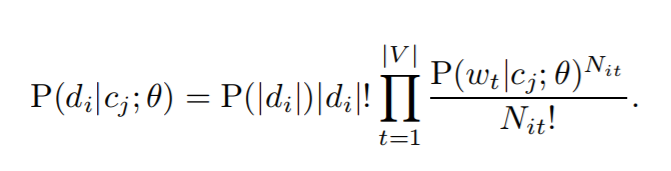
In the classification phase, we classify an unlabeled vector by assigning the label which is most frequent among the 5 training samples nearest to that query point. We make use of the Euclidean distance metric to calculate the distance between points.

Notes:

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest neighbors due to their large number. One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its k nearest neighbors. The class of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point.

**Multinomial Naïve Bayes Algorithm**

Multinomial Naive Bayes is a specialized version of Naive Bayes that is designed more for text documents. Whereas simple naive Bayes would model a document as the presence and absence of particular words, multinomial naive bayes explicitly models the word counts and adjusts the underlying calculations to deal with it. In contrast to the multi-variate Bernoulli event model, the multinomial model captures word frequency information in documents. Consider, for example, the occurrence of numbers in the Reuters newswire articles; tokenization maps all strings of digits to a common token. Since every news article is dated, and thus has a number, the number token in the multi-variate Bernoulli event model is uninformative. However, news articles about earnings tend to have a lot of numbers compared to general news articles. Thus, capturing frequency information of this token can help classification. In the multinomial model, a document is an ordered sequence of word events, drawn from the same vocabulary V. We assume that the lengths of documents are independent of class. We again make a similar naive Bayes assumption: that the probability of each word event in a document is independent of the word’s context and position in the document. Thus, each document di is drawn from a multinomial distribution of words with as many independent trials as the length of di. This yields the familiar “bag of words” representation for documents. Define Nit to be the count of the number of times word wt occurs in document di. Then, the probability of a document given its class from Equation 1 is simply the multinomial distribution:



Implementation:

Parameters:

We chose 3 parameters namely: alpha set to 1.0 which is the additive smoothing parameter to avoid the zero-conditional probability, fit\_prior set to True to allow calculation of all the class prior probabilities based on the dataset at hand and class\_prior set to None to not assume any default prior probabilities for classes.

Model Fitting:

We first obtain number of samples contained in each class and number of feature samples contained in each class. Then we adjust and calculate the smoothed empirical log probability for each class using alpha and the empirical log probability of the relevant features in the model input given a class.

Sample Class Prediction:

When a sample is received for prediction, we first calculate the feature log probabilities for the features contained in the sample with the help of dot product with the overall available feature log probabilities. The obtained feature log probabilities are then multiplied with the class log prior probabilities. Finally, the class resulting in the maximum joint log probability value is selected as the prediction for the sample.

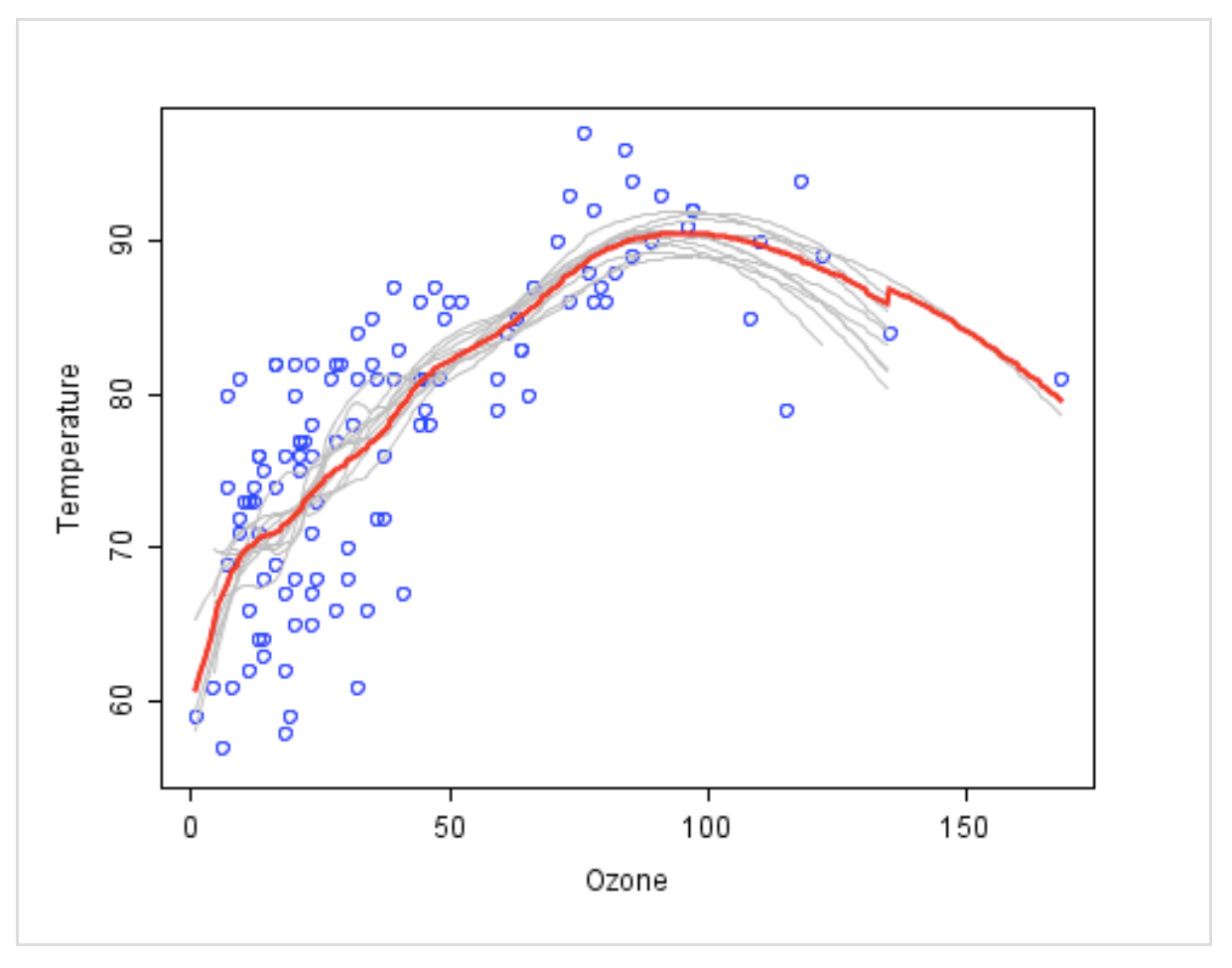
Notes:

During calculation of joint log probabilities, multiple feature log conditional probabilities are multiplied, one for each position $1 \leq \tcposindex \leq n_d$. This can result in a floating-point underflow. It is therefore better to perform the computation by adding logarithms of probabilities instead of multiplying probabilities. The class with the highest log probability score is still the most probable; $\log (xy) = \log (x)
+ \log (y)$ and the logarithm function is monotonic. Hence, the maximization that is actually done in most implementations of Naïve Bayes is:

|  |
| --- |
| $\displaystyle c_{map} = \argmax_{\tcjclass \in \mathbb{C}} \ [ \log \hat{P}(\tc... ...{1 \leq \tcposindex \leq n_d} \log \hat{P}(\tcword_\tcposindex\vert\tcjclass)].$ |

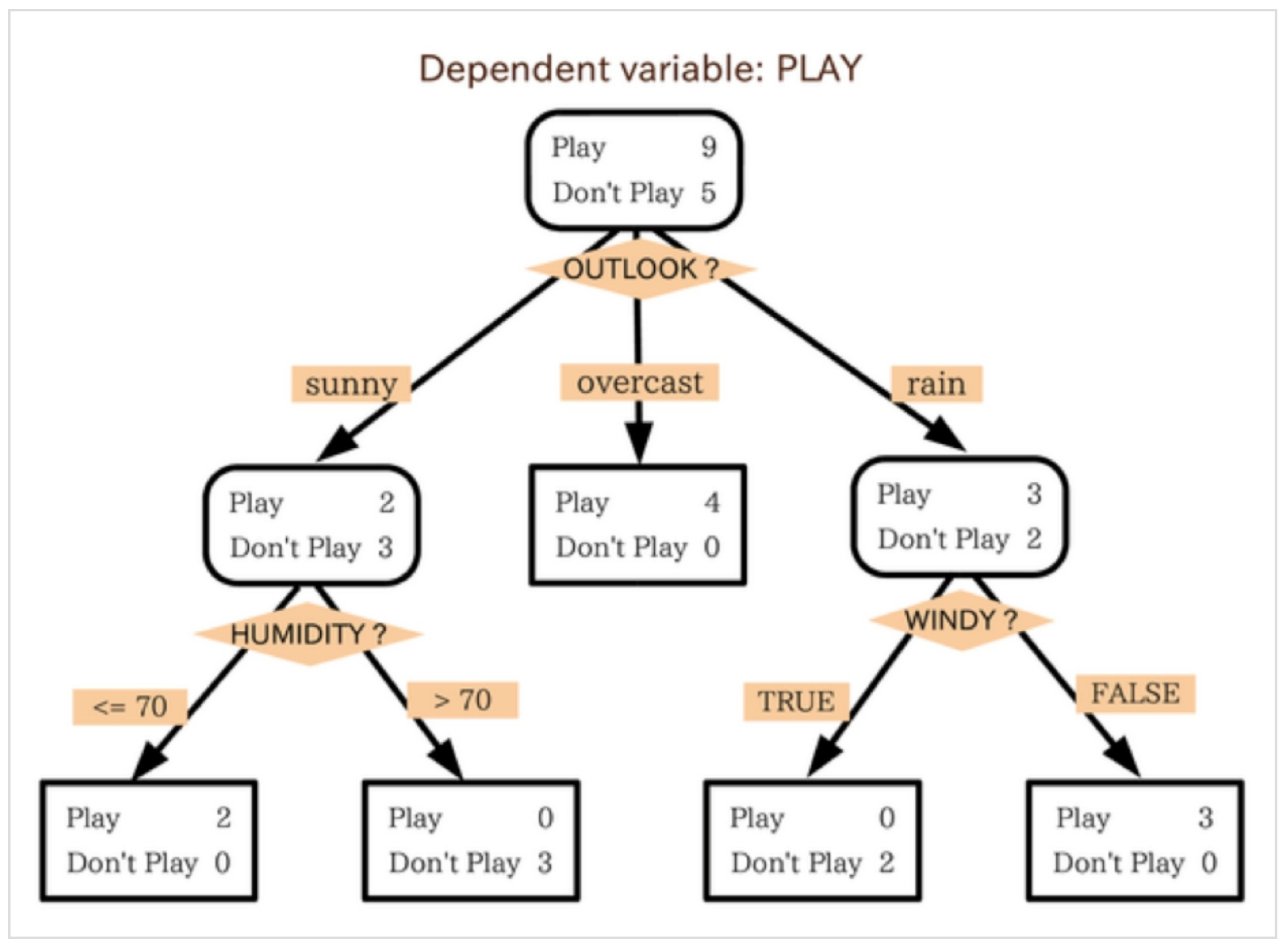
**Random Forest Algorithm**

The random forest that we implemented as our third classification algorithm is an ensemble approach that can also be thought of as a form of nearest neighbor predictor. Ensembles are a divide-and-conquer approach used to improve performance. The main principle behind ensemble methods is that a group of “weak learners” can come together to form a “strong learner”. The figure below (taken from [here](http://en.wikipedia.org/wiki/Bootstrap_aggregating)) provides an example. Each classifier, individually, is a “weak learner,” while all the classifiers taken together are a “strong learner”. The data to be modeled are the blue circles. We assume that they represent some underlying function plus noise. Each individual learner is shown as a gray curve. Each gray curve (a weak learner) is a fair approximation to the underlying data. The red curve (the ensemble “strong learner”) can be seen to be a much better approximation to the underlying data.

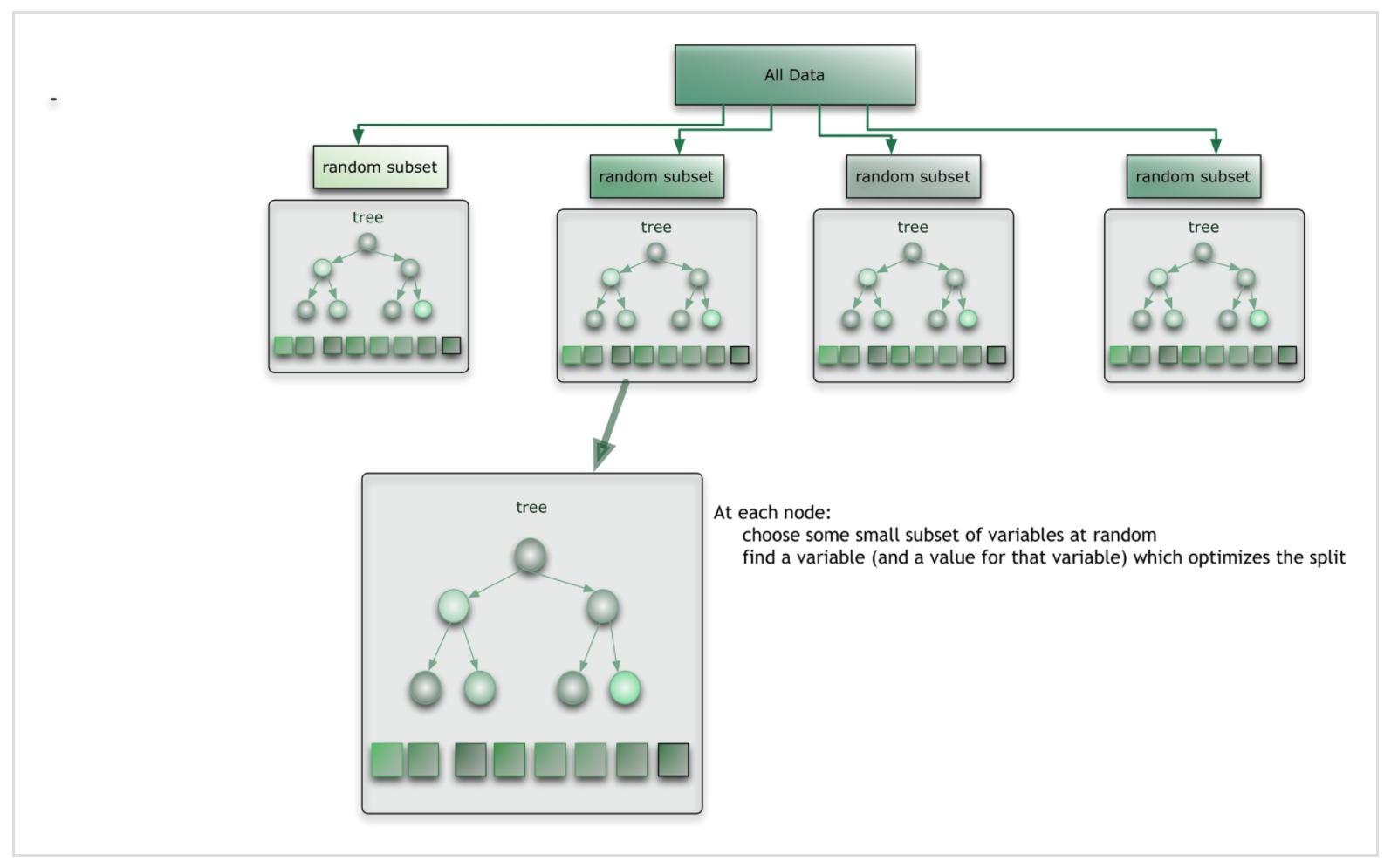


We use a standard machine learning technique called a “decision tree” from the Scikit package which, in ensemble terms, corresponds to our weak learner. In a decision tree, an input is entered at the top and as it traverses down the tree the data gets bucketed into smaller and smaller sets. For details see here, from which the figure below is taken.

In this example, the tree advises us, based upon weather conditions, whether to play ball. For example, if the outlook is sunny and the humidity is less than or equal to 70, then it’s probably OK to play.



The random forest (see figure below) takes this notion to the next level by combining trees with the notion of an ensemble. Thus, in ensemble terms, the trees are weak learners and the random forest is a strong learner.



Implementation:

1. In the general approach, first *N* cases are sampled at random with replacement to create a subset of the data (see top layer of figure above). The subset is recommended to be at least 66% of the total set. For our implementation, we consider all the data points available as N.
2. At each node:
   1. We select m=3 variables at random from all the predictor variables.
   2. Then, we use the predictor variable that provides the best split, according to the Gini index, to do a binary split on that node.
   3. At the next node, we choose another set of *m* variables random from all predictor variables and the same process is repeated.

Parameters:

Apart from the parameters specified above, we also used 100 decision tree predictors in our algorithm to create a random forest for prediction.

Methods:

1. get\_predictions: The get\_predictions method returns the predicted class of an input sample which is determined by a vote by the trees in the forest, weighted by their probability estimates. That is, the predicted class is the one with highest mean probability estimate across the trees. It takes as input an array-like or sparse matrix of shape = predictor variables and return as output an array of predicted classes.

2. get\_model: The get\_model method builds a forest of trees from the training set (X, y). It takes as input an array-like or sparse matrix of shape = predictor variables X and an array of class labels or target values y.

Sample Class Prediction:

When a new input is entered into the system, we run it down all the trees. The result is selected based on a voting majority.

Notes:

* With many predictors, the eligible predictor set is quite different from node to node.
* The greater the inter-tree correlation, the greater the random forest error rate, so one pressure on the model was to have the trees as uncorrelated as possible.
* As *m* goes down, both inter-tree correlation and the strength of individual trees go down. So some optimal value of *m* had to be discovered. The √*m* that we chose works efficiently for our dataset.

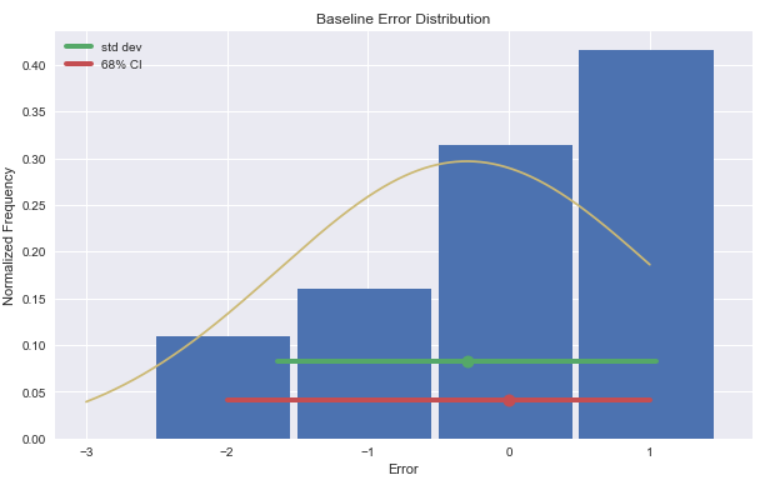
*Strengths and weaknesses*. Our Random forest implementation is quite fast, and it is also able to deal with unbalanced and missing data. The weaknesses of our implementation is that it may over-fit our Yelp data set if the amount of noise in the data increases. Overall we feel our algorithm works well on our data set.

1. **Experimental Results**

**Baseline Model**

An interesting observation that resulted from a histogram plot of review ratings has led us to build a baseline model that is based on average rating. The average rating of all the restaurant reviews in our data set is 3.7 and it is rounded off to 4 and used as the baseline prediction. The accuracy of the baseline model is demonstrated in the table below.

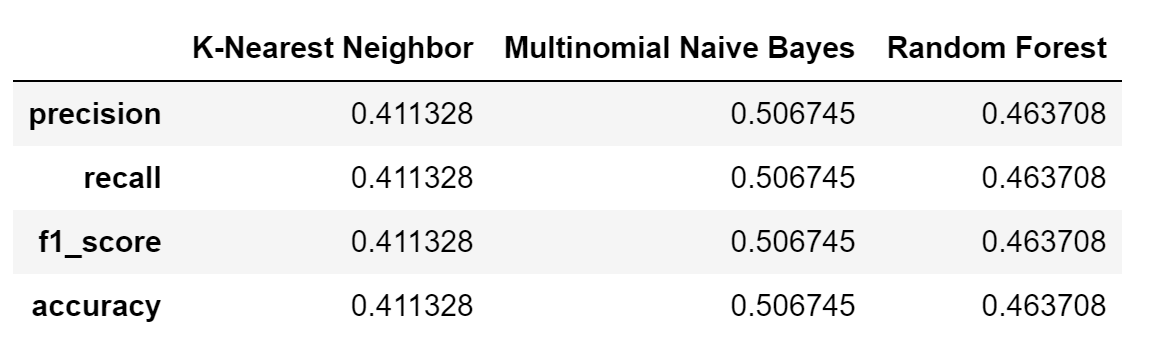


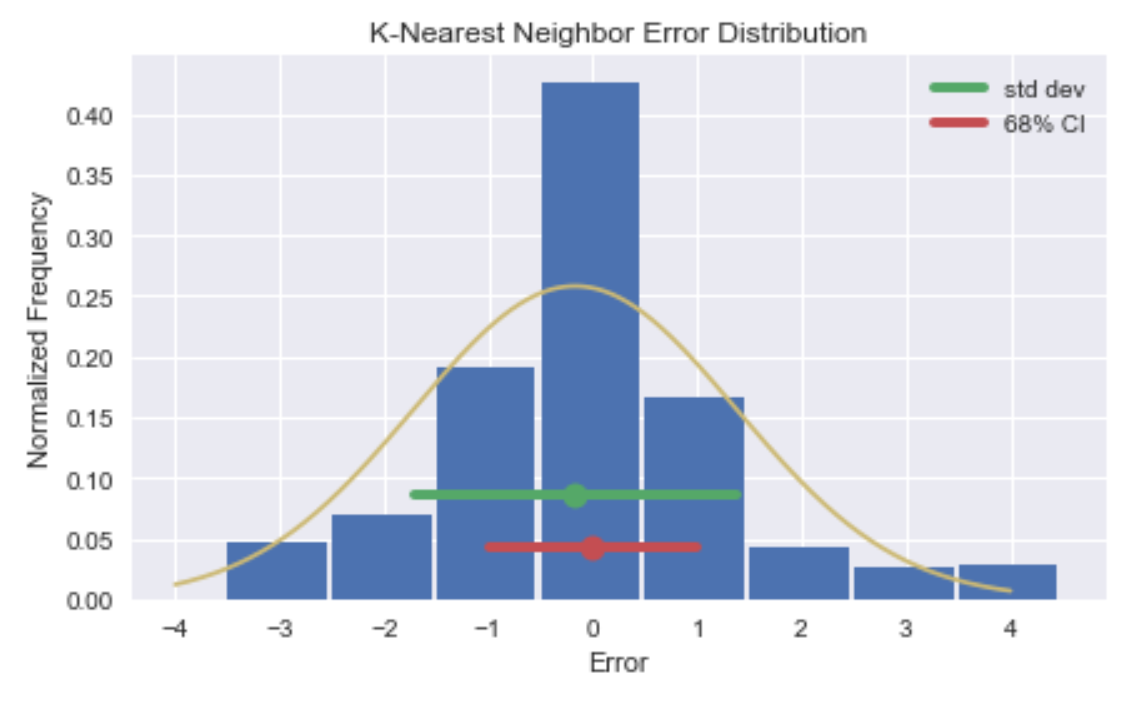


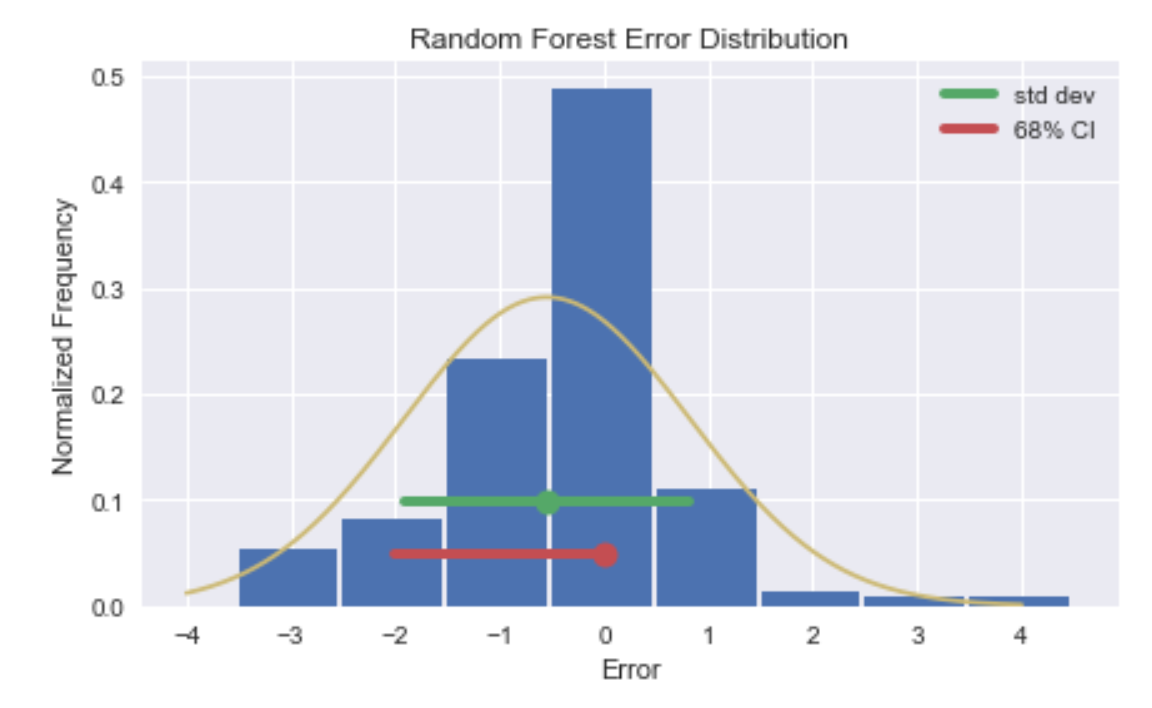
Once we had the baseline model, we built our advanced model that helped predict the rating stars. The underlying idea was extracting key features of a review to help predict the sentiment. The model used topics and was further improved upon to use topics combined with the sentiment. Latent Dirichlet Allocation (LDA) was used to extract features from the review dataset. After the feature extraction, the data is split randomly into train data (80%) and test data (20%). The models trained using these features is then evaluated on the test data.

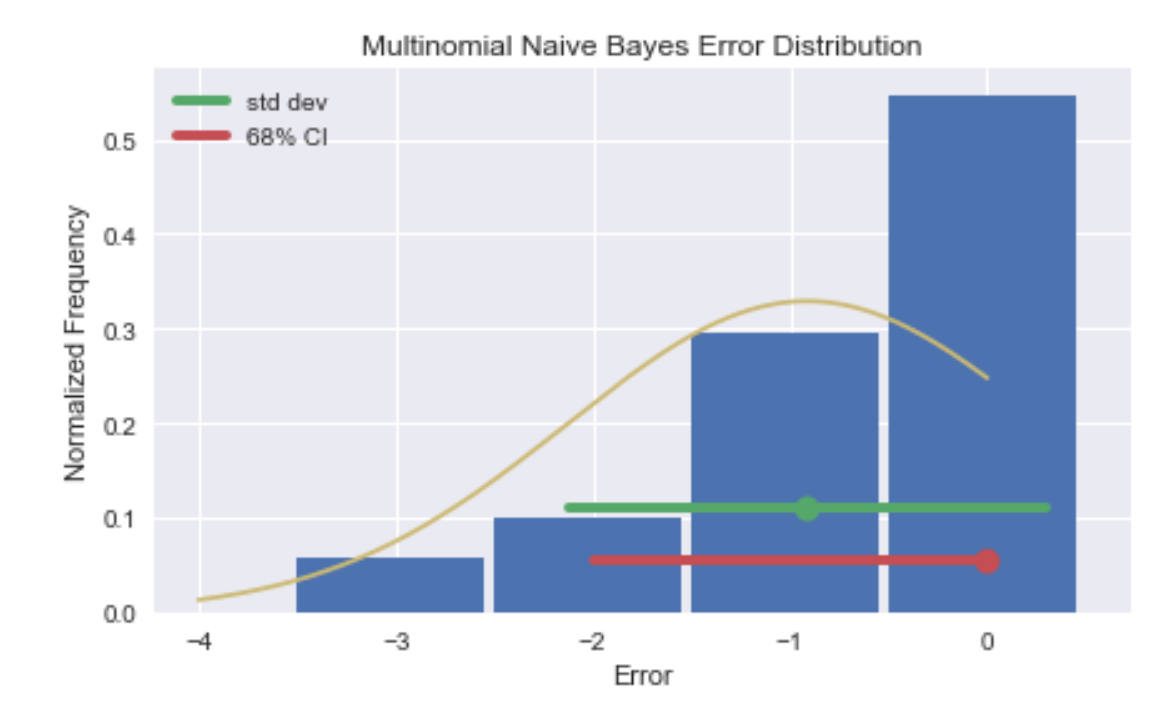
**Using Topics from LDA**

Our initial thought was to consider all the words and their frequencies as training features. However, it might be inefficient to do such a task when the data is huge and it might affect the accuracy of prediction when the features span a wide range of words. So, we extracted only key features of a review, called topics, and used them as training features. To extract topics, we used Latent Dirichlet Allocation proposed by Blei. The results of each classification algorithm with Topics are demonstrated below:



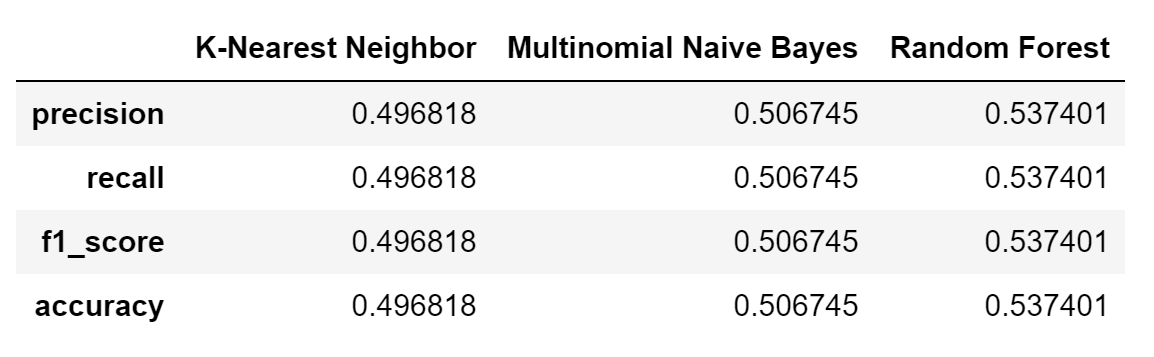


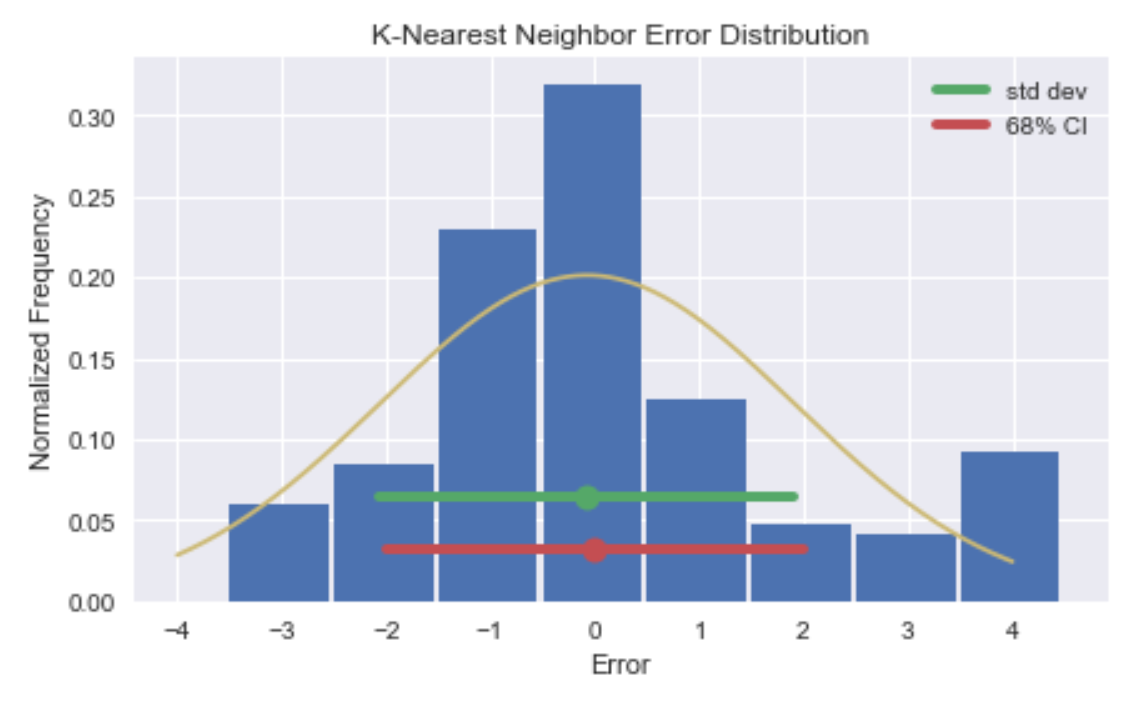




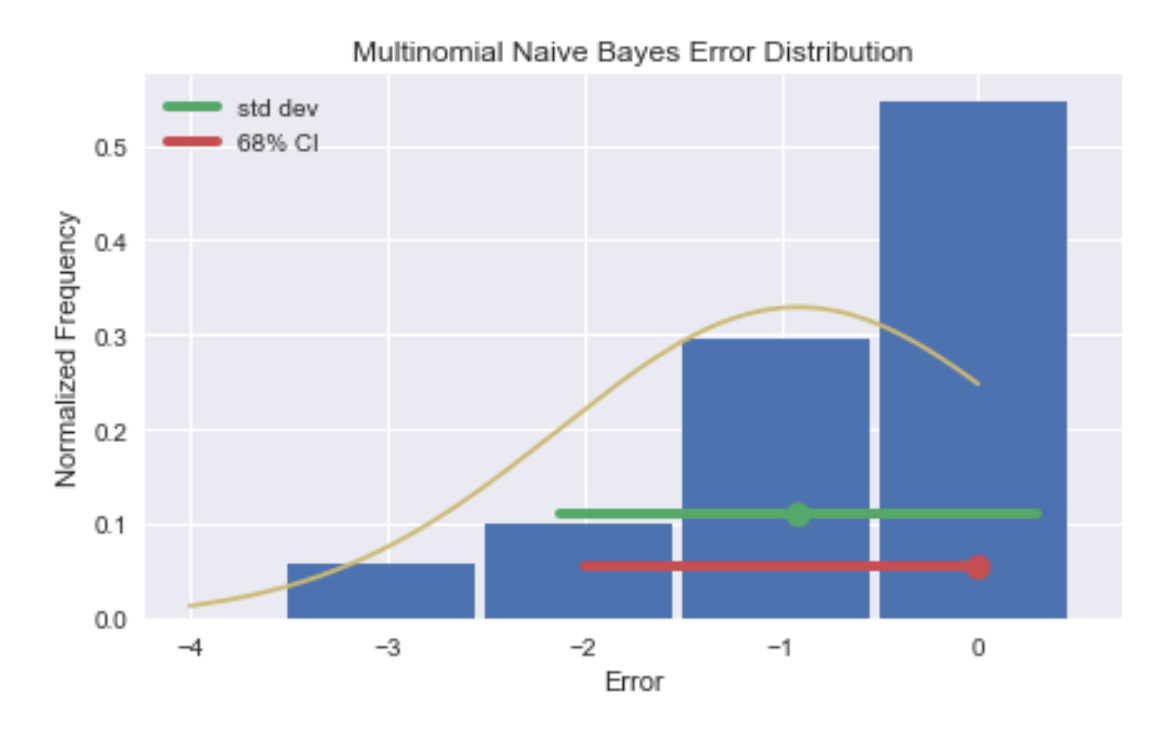
**Using LDA Topics and Sentiment Score**

As observed in the results above, using topics as features to train the classifier resulted in a lower accuracy overall. This could be due to the fact that topics don’t have a sentiment associated with them. For example, consider the following two reviews: Review 1: “This place has great ambience.” Review 2: “This restaurant doesn’t provide valet parking.” The first review talks about ambience in a positive manner and the second review talks about parking in a negative sense but the topics extracted from these reviews would just be ‘ambience’ and ‘parking’. They don’t capture the essence of the sentiment and thereby would not serve useful as features to determine star rating. This led us to the idea of adding sentiment as a feature along with topics to train the model. We used Naive Bayes Classifier to extract the sentiment. The extracted sentiment, along with the topics, are passed to each of the classification algorithms and the results obtained are demonstrated below:









1. **Conclusion**

The motivation for this project was to implement different classification techniques to predict a review’s star rating from its review text. This has applications in tasks such as information retrieval, opinion mining, text summarization and many other problems which involve large amount of textual data.

In this report, we have discussed our approach which involved the combination of topic modeling and sentiment analysis to predict the star rating. Latent Dirichlet Allocation (LDA) was used for feature extraction. We implemented K Nearest Neighbors, Multinomial Naïve Bayes and Random Forest classification algorithms in Python and used these for predicting ratings based on user reviews. The Yelp data set that was used was large in size (approximately 19.2 GB) and difficult to handle.

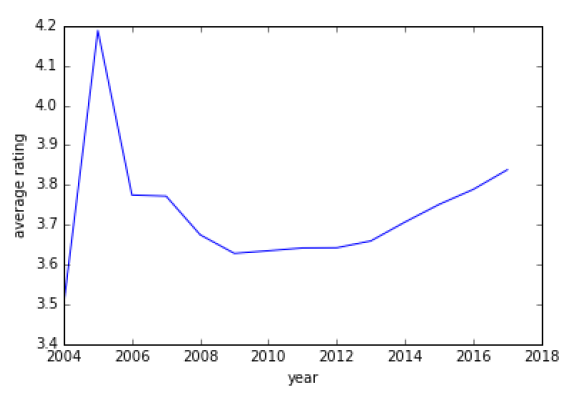
The splitting of dataset and selection of train set and test set had a significant impact on the overall accuracies of the classification algorithms we implemented. Cross-validating and tuning the parameters of our algorithms greatly helped in improving our model predictions.

The following are the cumulated results obtained for each of the algorithms. As we can see below, Random Forest with LDA and Sentiment Score was the most accurate among all the classification methods used with an accuracy of approximately 54%. It was also the faster among all the algorithms.

|  |  |  |
| --- | --- | --- |
|  | LDA | LDA + Sentiment Score |
| K Nearest Neighbors | 0.411328 | 0.496818 |
| Multinomial Naïve Bayes | 0.506745 | 0.506745 |
| Random Forest | 0.463708 | 0.537401 |

1. **Limitation**

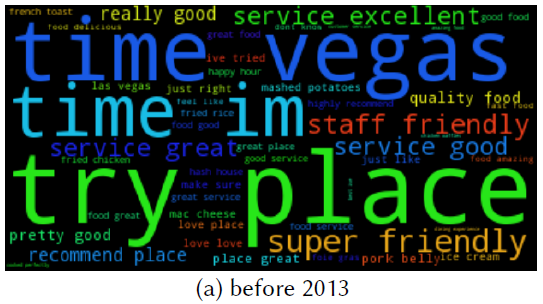
Average rating over the years



From the above figure, we can observe that average rating increases with every year from 2010 onwards. But from 2004 to 2009, the rating is kind of unstable, mainly due to sparse number of reviews. This might be due to the fact that it was during the initial phase of Yelp. To meet our dataset size requirements, we retained reviews during this period. But as more data is collected over the years, we can discard these reviews for a more consistent representation.

1. **Future Improvements**

In future work for this project, we would like to include temporal effect in our model by using different model parameters for years <= 2013 and >2013. We believe this may improve classification results because of the following analysis.



These word clouds were generated from our reviews database for restaurants in Las Vegas. We can see the difference in how users review restaurants in terms of the words they used before and after 2013. The reviews after 2013 became more focused towards the quality of the service and behavior of the staff. From the same time onwards, we can also observe that the slope for average rating (in the previous graph) increases significantly. This shows that temporal effect also determines prediction.

After pre-processing and eliminating stop words from reviews, unique terms in the corpus can be treated as unigrams. These give us a bag-of-words model of our data, which is a widely used representation in NLP and worth exploring for our problem statement. To also capture the effect of phrases such as `not delicious' or `tasty fish burger', we can further include bigrams and trigrams, and compute tf-idf features from them for classification. For our data, we would like to obtain and compare accuracies achieved using these tf-idf features set as opposed to LDA + Sentiment Score features.

Lastly, we did not get a chance to implement some other classification techniques like Logistic Regression, AdaBoost, etc. that have proven to work well on the Yelp dataset. It would be interesting to implement these classification algorithms and verify the results they produced.

**References**

* F. Li, N. Liu, H. Jin, K. Zhao, Q. Yang, and X. Zhu. Incorporating reviewer and product information for review rating prediction. In Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), pages 1820-1825, 2011.
* http://www.yelp.com/dataset challenge
* Qu, Lizhen, Georgiana Ifrim, and Gerhard Weikum. "The bag-of-opinions method for review rating prediction from sparse text patterns." Proceedings of the 23rd International Conference on Computational Linguistics. Association for Computational Linguistics, 2010.
* Scaria, Aju Thalappillil, Rose Marie Philip, and Sagar V. Mehta. "Predicting Star Ratings of Movie Review Comments."
* http://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics
* https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/
* https://en.wikipedia.org/wiki/K-nearest\_neighbors\_algorithm
* https://github.com/scikit-learn/scikit-learn
* David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet allocation. the Journal of machine Learning research, 3:993-1022, 2003.
* Ganu, Gayatree, Noemie Elhadad, and Amelie Marian. "Beyond the Stars: Improving Rating Predictions using Review Text Content."WebDB. Vol. 9. 2009.
* https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/
* https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/
* https://machinelearningmastery.com/implement-random-forest-scratch-python/
* Li, Chen, and Jin Zhang. "Prediction of Yelp Review Star Rating using Sentiment Analysis."
* Chada, Rakesh, and Chetan Naik. "Data Mining Yelp Data-Predicting rating stars from review text."
* L. Qu, G. Ifrim, and G. Weikum. The bag-of-opinions method for review rating prediction from sparse text patterns. In Proceedings of the 23rd International Conference on Computational Linguistics (COLING2010), pages 913-921, 2010.
* Mingming Fan and Maryam Khademi. Predicting a business star in yelp from its reviews text alone. arXiv preprint arXiv:1401.0864, 2014.
* Clayton J Huo and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Eighth International AAAI Conference on Weblogs and Social Media.