Springboard Data Science Career Track
Capstone Project

Inferring Restaurant Ratings: San Francisco Restaurant Reviews

Darren Lyles (2019)

Table of Contents

I. Introduction

Have you ever looked up the ratings and reviews of nearby restaurants to see what is good?
Have you ever looked at your own restaurant ratings and reviews to see what others think?

If you answered yes to either of these questions, please read on!

This project will focus on restaurant reviews and ratings in the city of San Francisco, California. Restaurant reviews are important because they play a major role in influencing the success of the business. They influence the reputation of the restaurant , and the opinions of current and prospective patrons. Restaurant owners and patrons alike can usually get honest reviews from websites such as Yelp, OpenTable, Tripadvisor, etc. Restaurant owners can then use this feedback to assess their strengths and weaknesses to potentially take action. Potential patrons can also use restaurant reviews to avoid having a bad experience.

II. Overview of Dataset

The dataset, which can be found at Kaggle.com, contains a collection of restaurants in the city of San Francisco, California with ratings and reviews from various TripAdvisor users. The raw data is composed as a JSON file and when initially unpacked, has 147 rows and 41 columns. Many of these columns have boolean features which is not relevant to sentiment analysis. Examples of these features include, but are not limited to: meal_breakfast, meal_lunch, meal_dinner, meal_deliver, meal_cater, gluten_free, and wifi. Such columns were dropped so that I could focus on relevant features such as the restaurant ratings and reviews. The reviews column, which is the most relevant for the scope of this project, is packed as a list of python dictionaries. During the data wrangling phase, this will be unpacked as a separate data frame and then merged with the data set such that each restaurant review will be on separate rows.

III. Data Wrangling

As mentioned earlier, the raw dataset is a JSON file which contains compact information about each restaurant in one row. The features of the restaurant, including its reviews and ratings, are stored in a dictionary data structure. The JSON file comes with two other columns named restaurant_count and total_review_count. These values broadcasted on these columns are 147 and 14700, respectively.

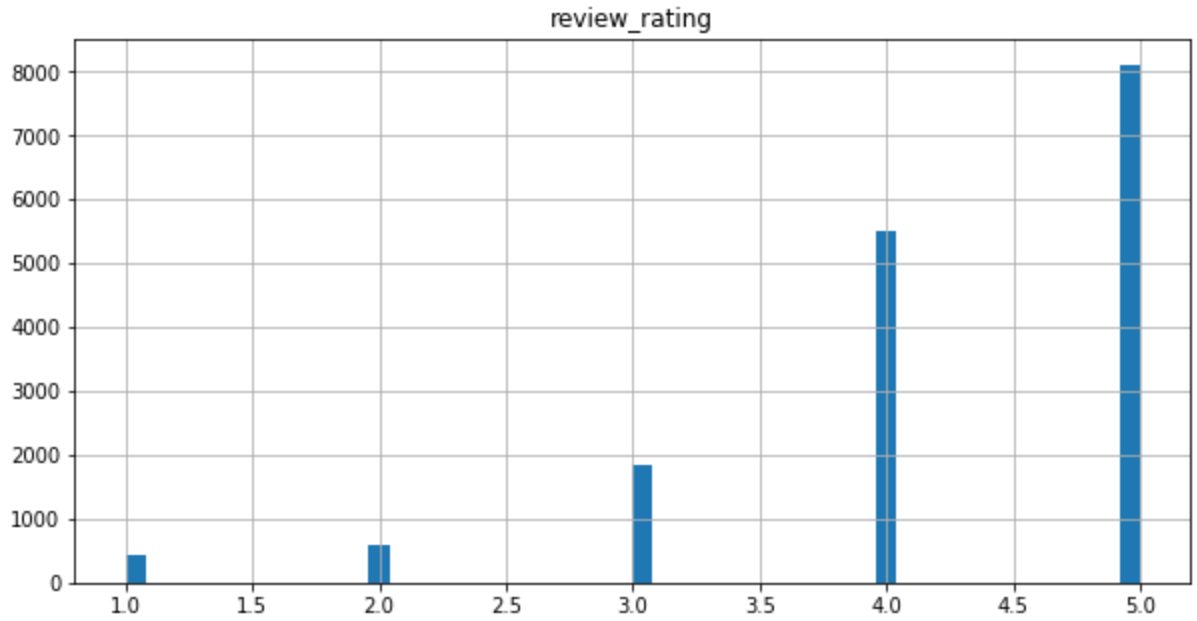
Our column of interest is the restaurant column, and so we unpack the dictionary contents of this column into a data frame. From there, we are able to easily access 147 rows of restaurants and 41 columns of features associated with each restaurant. The 41 features also include the restaurant reviews. This is the feature of interest, and so we drop most of the features but keep some such as: cuisine, address, locality, region, hours, email, tel, text, trip_advisor_url, website, latitude, longitude, price, and rating. These features are kept for informational purposes.

The reviews feature is a dictionary data structure and that must be unpacked as well. This will result in expanding the rows of the dataset such that each review will have its own row. After unpacking the reviews dictionary, we store it into a separate data frame. These features in the reveiws dataframe are: review_url, review_title, review_text, review_rating, review_date. We then expand our current data frame by merging it with the reviews data frame. The new and ready to use dataset has 16500 rows and 20 columns, which gives us 16500 restaurant reviews and review ratings to work with! There were no missing values for the restaurant reviews and review ratings.

IV. Exploratory Data Analysis

Before going into exploring the dataset, I added one more column called tokenized_review_text. The main reason for doing this is that I wanted to have a tokenized version of each restaurant review so I can apply the appropriate machine learning models to it. The process, in short, involved tokenizing each restaurant review by word, removing punctuation and stop words, and lemmatizing. Formatting the reviews in this way will make it possible to use the CountVectorizer for text feature extraction.

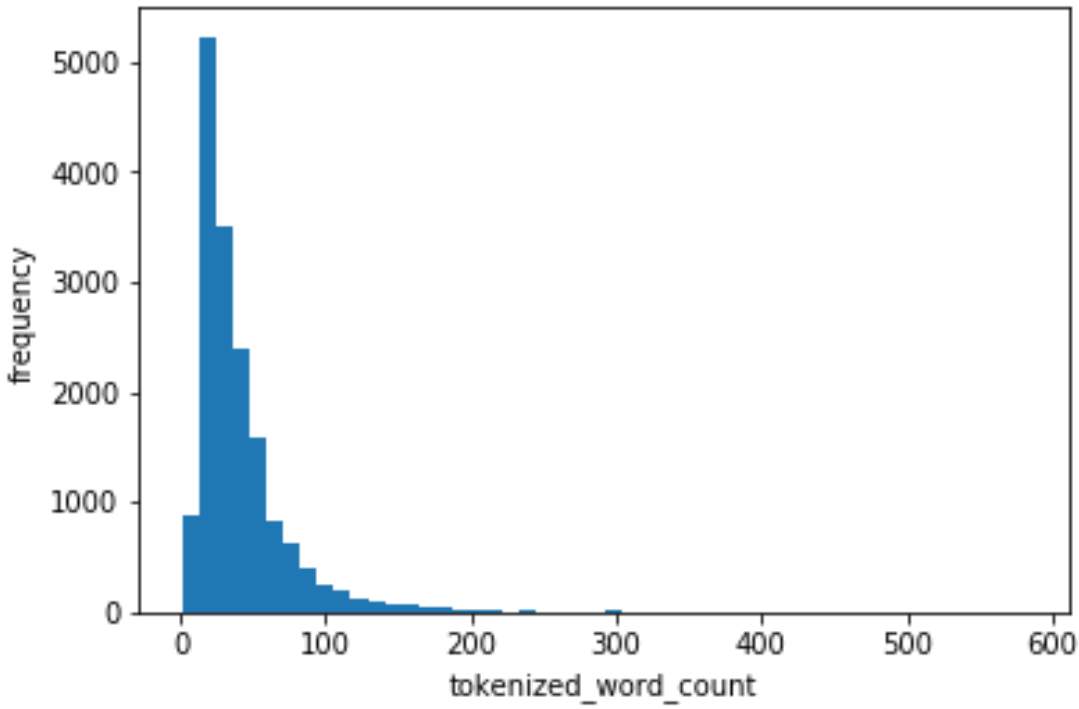
One feature of interest is the review ratings, the following graphic below shows the review rating distribution:



The corresponding frequency table is as follows:

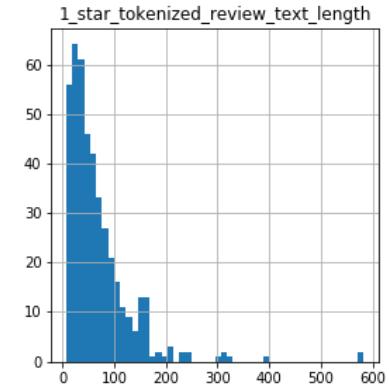
Review Ratings	Frequency
1-Star	436
2-Star	601
3-Star	1833
4-Star	5521
5-Star	8109

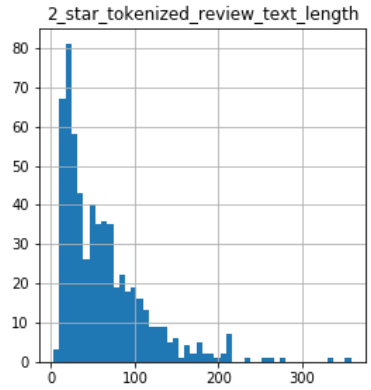
As we can see from above, we have a disproportionate amount of 5-star, 4-star, and 3-star reviews compared to 2-star and 1-star reviews. This class imbalance may cause our machine learning models to classify 5-star, 4-star and 3-star restaurant reviews with better precision. Since the 1-star and 2-star review ratings are much less in quantity, the model may have a lower precision metric, consequently lowering the overall accuracy score of the model.
To get an idea of how long the restaurant reviews in the dataset are, excluding stopwords, we can look at the distribution below:

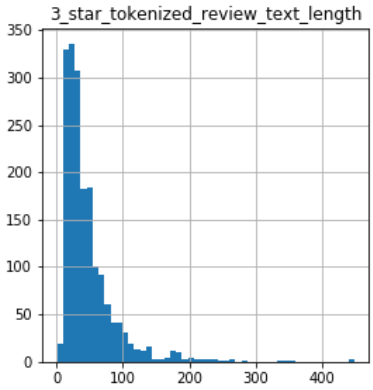


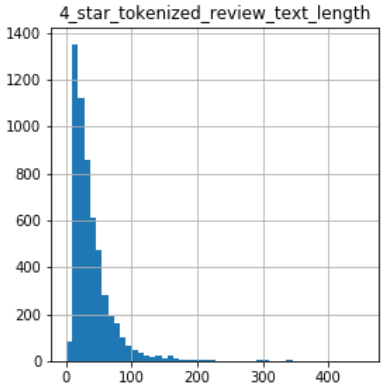
This distribution takes the tokenized review text and plots the frequency of the tokenized word counts. The distribution has a tokenized minimum word count of 1 and a maximum of 583. The mean, median, and standard deviation are 41, 31, and 35, respectively. We see a spike at about 20-30 words with a frequency of well over 5000 words. This distribution also skewed to the right, which means a small yet significant number of patrons had quite a bit to say in their review.

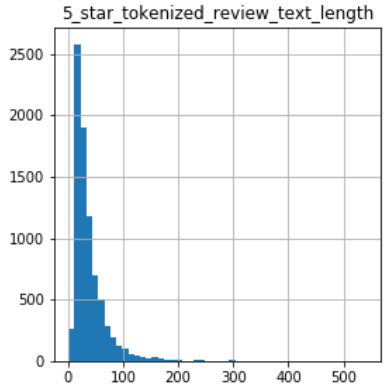
We can further break down the tokenized word counts by review rating as shown below:





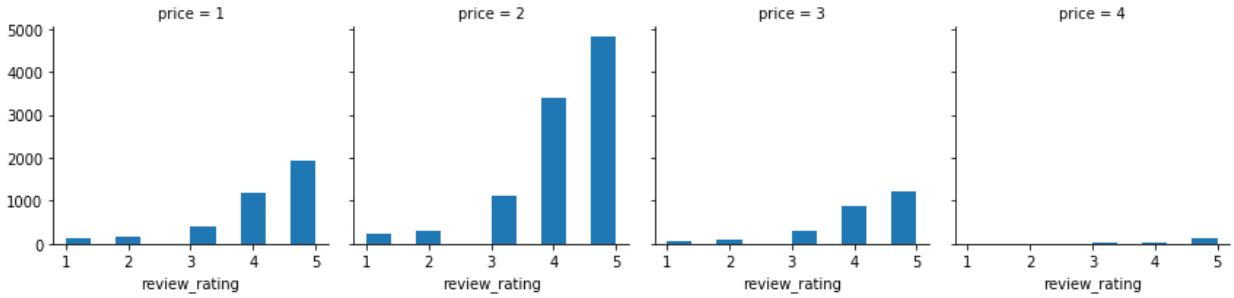






We can see that irrespective of the restaurant review ratings, the distribution of word counts are skewed to the right and the peak frequency is around 20-30 words for each review rating category.

The following plots shows the number of ratings with respect to the price. It seems that restaurant reviewers in San Francisco tend to go to restaurants with two dollar signs. Not only that, but two dollar sign restaurants have the highest number of 5 star ratings than any other restaurants. This should not however, imply that the more 5 star restaurant ratings, the better. This is an example of voluntary response bias since not every person rates and reviews a restaurant they visited, and the frequency of restaurant reviewers for each restaurant is not uniform. This means the number of reviewers for each restaurant is not the same in general.



V. Machine Learning

That fed into the machine learning models for training and testing will be the restaurant reviews (raw or tokenized) and the restaurant review ratings. This is a classification problem in which a model will take the review text as input, and classify it in one of the five categories of ratings: 1-star, 2-star, 3-star, 4-star, and 5-star.

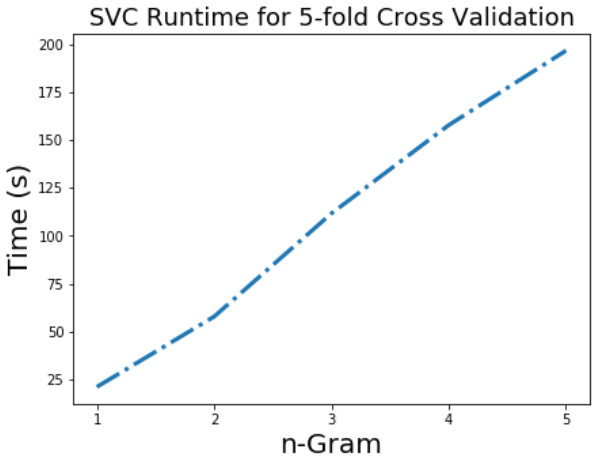
For this project, four different classifiers were used: LinearSVC, MultinomialNB (Naive Bayes), RandomForestClassifier, and XGBoost (Extreme Gradient Boosting). Each classifier was tested with varying n_gram parameters and with the CountVectorizer and TfIdfVectorizer.

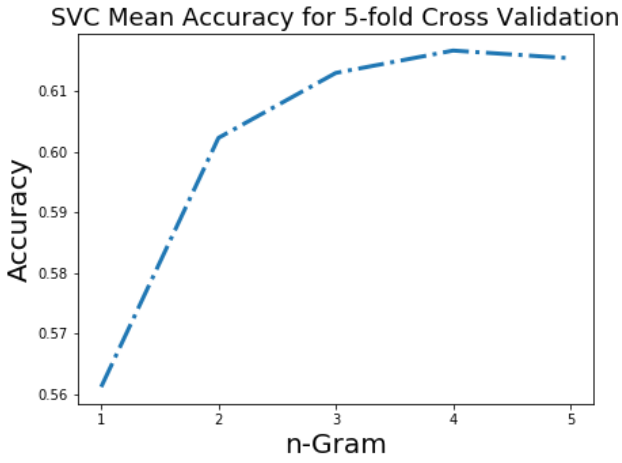
> **-n_gram** refers to the number of adjacent words that the model will treat as one unit. An example is if n_gram=(1,3), the model will evaluate with single words, a pair of adjacent words, and a group of three adjacent words.
>
> **-CountVectorizer** uses the bag of words concept and creates a sparse matrix for each feature (word or group of adjacent words) and its corresponding frequency in the review corpus.
>
> **-TfIdfVectorizer** uses the term frequency-inverse document frequency concept and creates a sparse matrix for each feature and its corresponding TF-IDF score.
> High scores correspond to frequency of feature in a rating but infrequent in the corpus, and low scores correspond to frequency in a feature across all ratings in the corpus.

Each model was tested using 5-fold cross validation. The model with the highest average score was selected and used to train and predict on the dataset for further analysis.

Linear Support Vector Machine Classifier

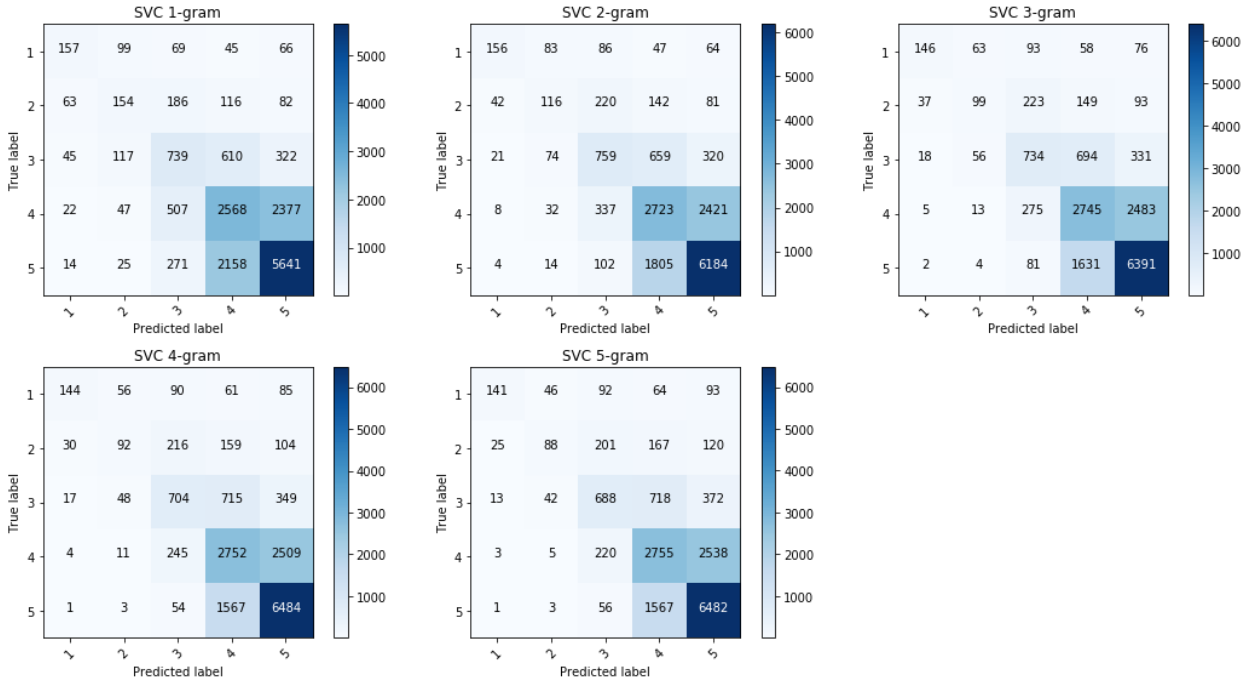
The first classifier I tested was the LinearSVC classifier. The LinearSVC classifier is essentially a special case of the support vector machine such that the vectors in the model will separate classes linearly. This essentially means that there is no curvature in the decision boundaries. The first set of tests used the ***CountVectorizer*** and here are the results:





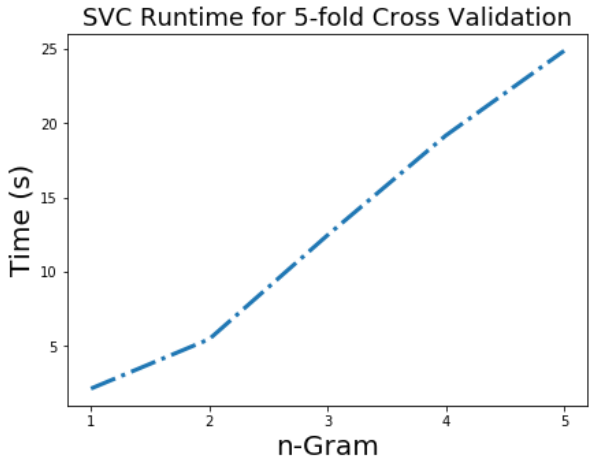
	Runtime (s)	Accuracy
1-gram	21.07	0.561149
2-gram	57.78	0.602302
3-gram	111.58	0.613030
4-gram	157.68	0.616727
5-gram	196.56	0.615456

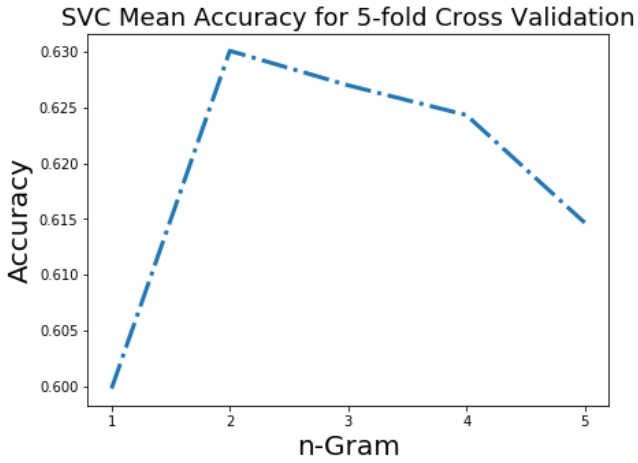
As we increase the number of grams that each model can take on, we see an initial significant rise in the accuracy as shown by the plot and table. However when comparing models that have 3-gram, 4-gram, and 5-gram features, the accuracy plateaus with 4-gram having the best overall accuracy score (61.67%). What we observe here is that the more features we add by setting the upper limit of n-grams, the slower the model takes to run. The runtime with respect to n-grams approximates linear behavior.



We will use confusion matrices to go into more detail for each classification model. For each model tested, we are looking at the accuracy of predicting the restaurant review ratings ranging from 1 to 5. Thanks to the EDA that was done prior, we know that there are significantly more 5-star reviews than 1-star reviews, this is a definite factor as to why predicting 1-star restaurants has a significantly lower precision. The small quantity of low ratings may well be the reason as to why our accuracy scores are relatively low. The SVC with 4-grams, which has the highest overall accuracy, also has the highest precision for 5-star ratings. This could well mean that the 5-star ratings are the dominant component in measuring the overall model accuracy.

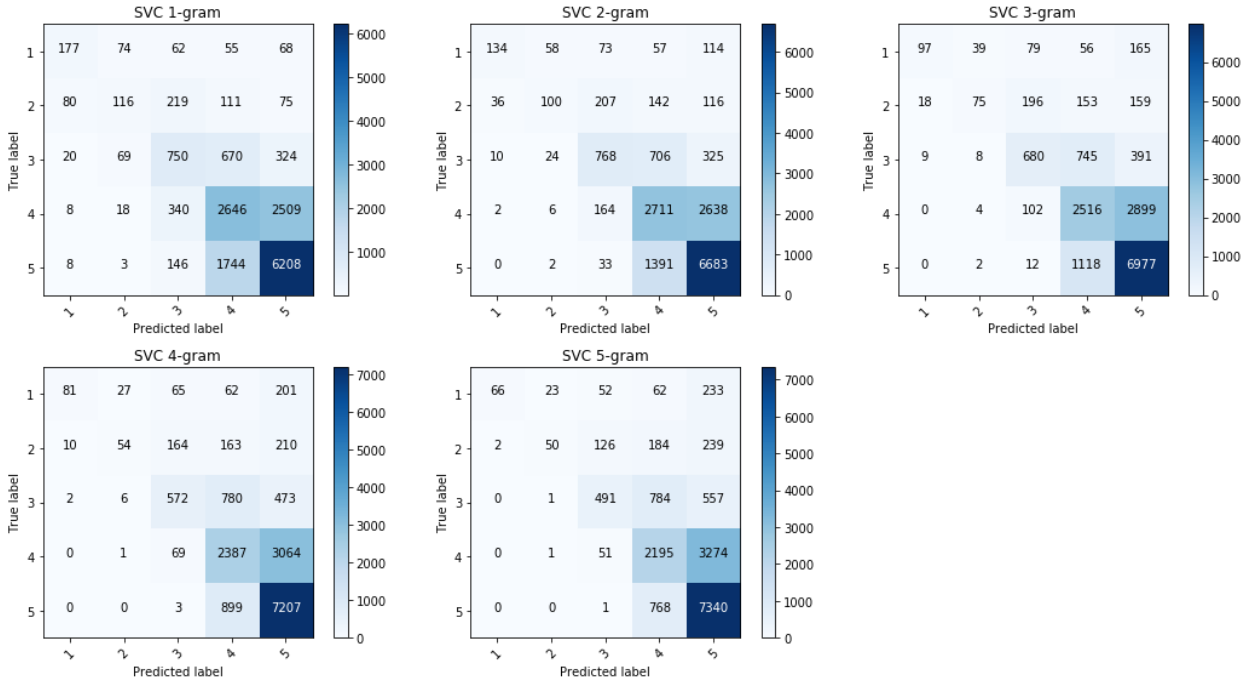
The next set of tests for the LinearSVC uses the ***TfIdfVectorizer***. The results are shown below:





	Runtime (s)	Accuracy
1-gram	2.15	0.599814
2-gram	5.48	0.630060
3-gram	12.48	0.626970
4-gram	19.19	0.624304
5-gram	24.89	0.614668

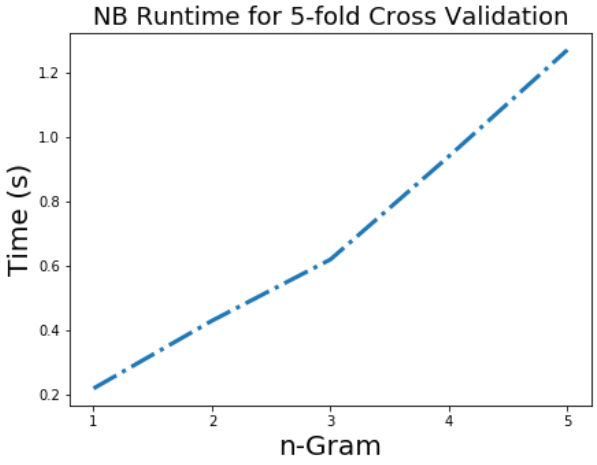
The 2-gram model in this set of tests has a average accuracy score of 63% and a runtime of about 5.5 seconds. As with the the previous set of models, the LinearSVC with the TfidfVectorizer has a roughly linear runtime behavior with respect to n-grams.

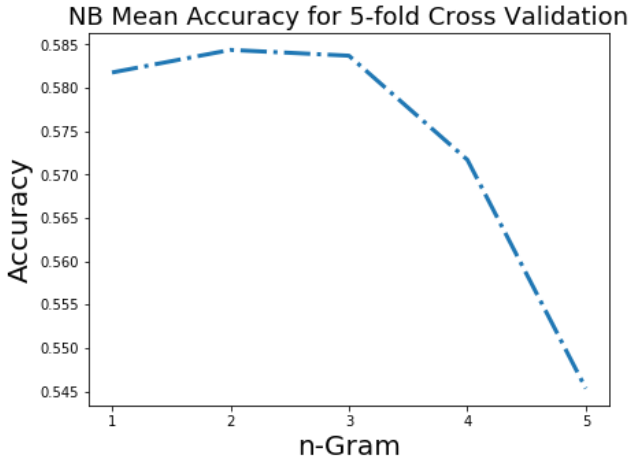


The confusion matrices above provide details into the accuracy of each model using the TfIdfVectorizer, we can see that it is able to classify 5-star restaurants better than the models using the CountVectorizer, however in the case of 3-gram, 4-gram, and 5-gram features, the models are poorer classifiers for 1-star restaurants.

Multinomial Naive Bayes Classifier

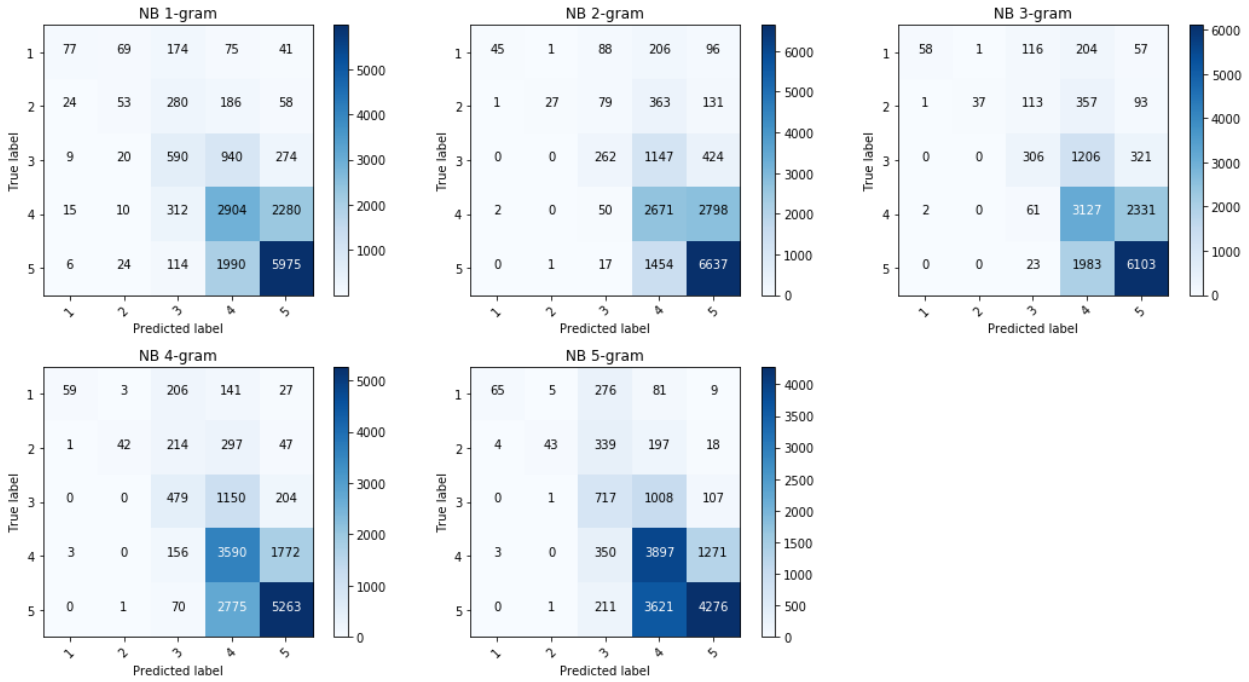
The second classifier I tested was the MultinomialNB classifier. This classifier is based off of conditional probability and computes the membership probability of a data point for each class. As with the LinearSVC classifier, we will look at the MultinomialNB classifier using the ***CountVectorizer***.





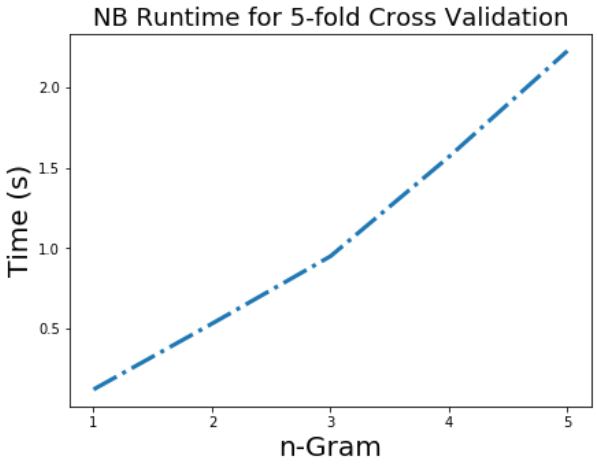
	Runtime (s)	Accuracy
1-gram	0.22	0.581759
2-gram	0.43	0.584364
3-gram	0.62	0.583698
4-gram	0.94	0.571700
5-gram	1.27	0.545338

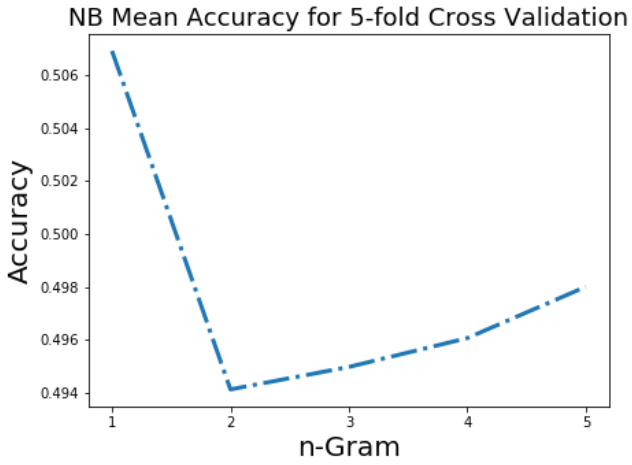
The model with the best accuracy is at 58% with a runtime of 0.43 seconds. As we increase the number of n for n-grams afterwards, the accuracy starts to lower, however since we have more input features with increasing n the runtime also increases and it shows roughly linear behavior.



As we can see in the confusion matrices, the best classifier among this group is determined by how many 5-star reviews it is able to correctly classify. Interestingly enough, it has the lowest precision amongst the other models for 1-star ratings. This classifier is great for predicting whether a restaurant review is a 5-star review, however it will have a high chance of missing a poor restaurant review by incorrectly classifying it.

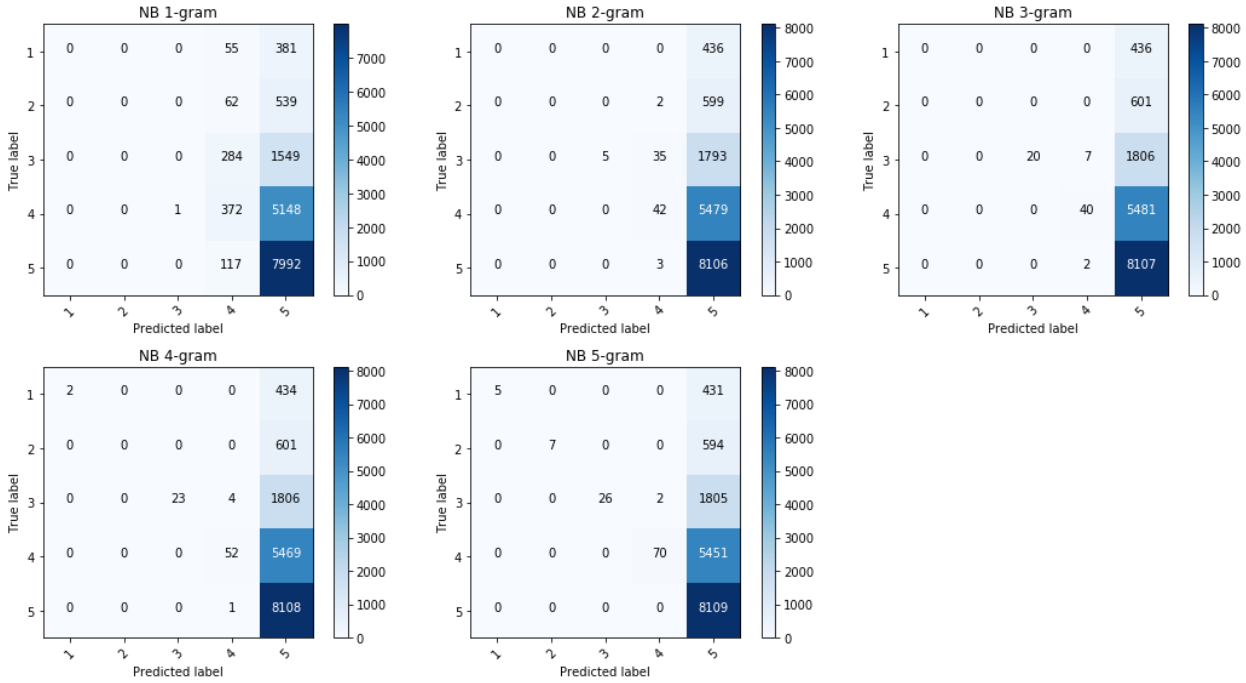
The next set of tests for the MultinomialNB uses the ***TfIdfVectorizer***. The results are shown below:





	Runtime (s)	Accuracy
1-gram	0.12	0.506910
2-gram	0.53	0.494122
3-gram	0.95	0.494970
4-gram	1.57	0.496061
5-gram	2.23	0.498000

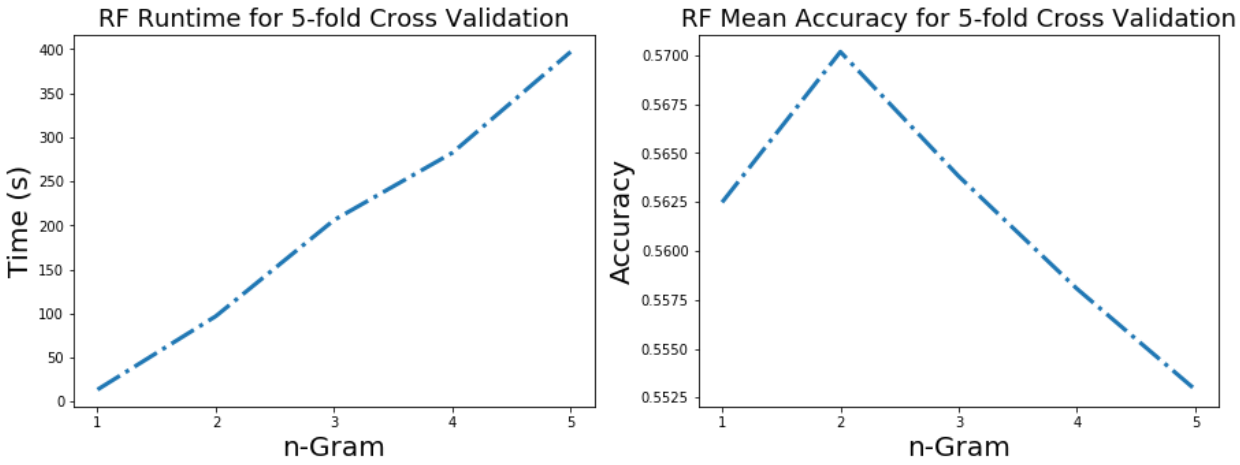
The model with the best accuracy in this set of tests is the naive bayes classifier using only unigrams. It has an accuracy of 51% and a runtime of 0.12 seconds. The runtime across all the models tested still exhibits linear behavior.



As we can see in the confusion matrices, the multinomial naive bayes classifier is an extremely poor classifier. A vast majority of the restaurant reviews are misclassified as being 5-star, whereas we can see that in the models taking 1-gram, 2-gram, and 3-gram features, they have no precision on classifying 1-star and 2-star reviews. The naive bayes classifier with 5-gram features is the only model in which has a diagonal with non-zero entries. Overall, these models are really poor classifiers.

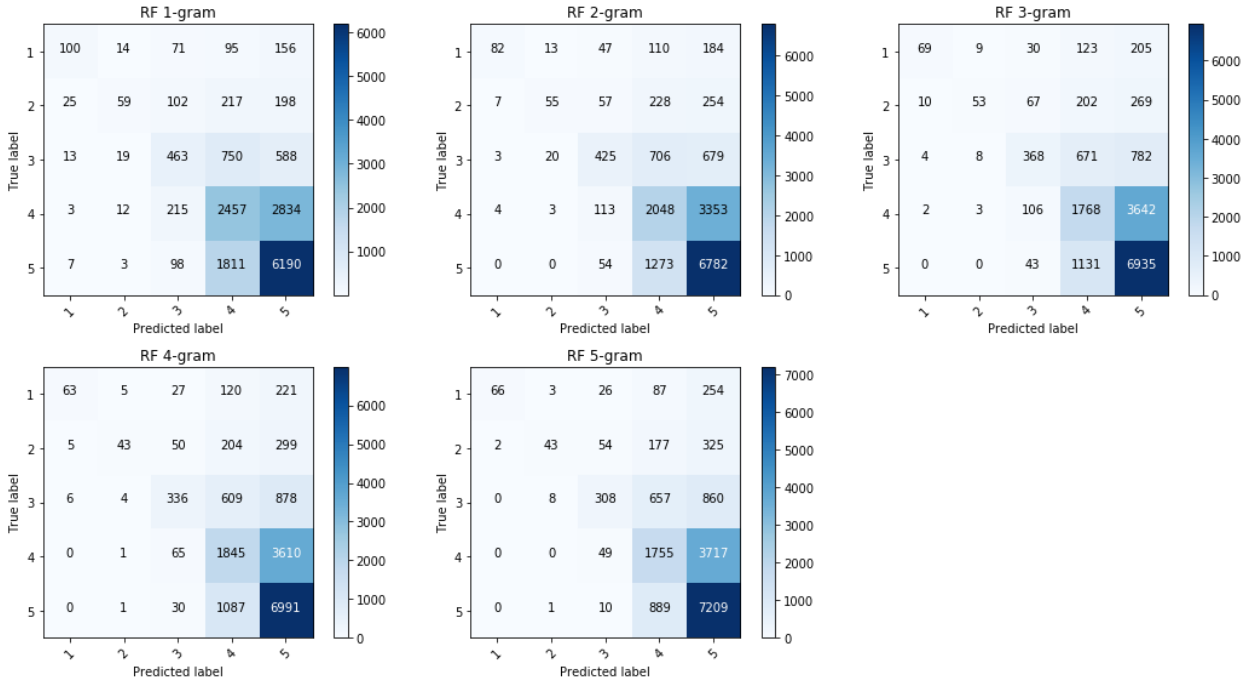
Random Forest Classifier

The third classifier that was tested was the RandomForestClassifier. This classifier is an ensemble method which uses multiple decision trees to classify data into a group of categories. We will now look at the results of the RandomForestClassifier using the ***CountVectorizer***:



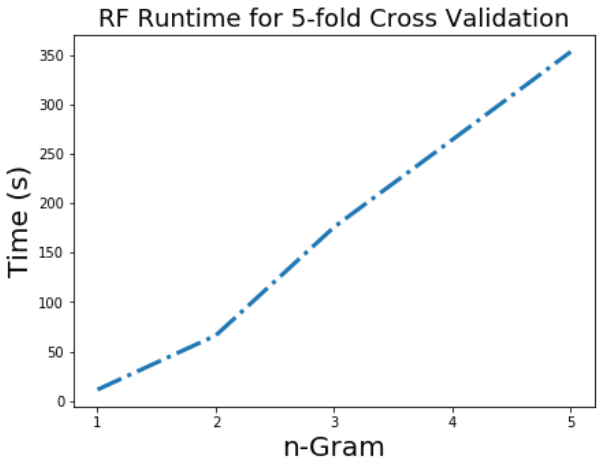
	Runtime (s)	Accuracy
1-gram	13.67	0.562485
2-gram	96.96	0.570182
3-gram	206.09	0.563819
4-gram	282.43	0.558064
5-gram	396.99	0.552908

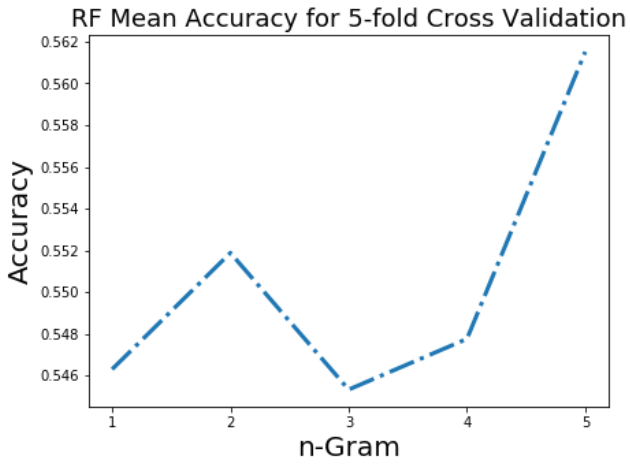
The best RandomForestClassifier using CountVectorizer has an accuracy of 57% and a runtime of 96.96 seconds. It contains bigram features. Just like in the previous models, the runtime behavior with respect to n-gram features is approximately linear. However, the more more features you include after 2-gram, the accuracy of the model decreases down to 2%.



The RandomForestClassifier using CountVectorizer also is a strong classifier for 5-star ratings, but very week for 1-star ratings just as the previous models. We can clearly see a pattern here where we lack strong precision in classifying low review ratings. We can expect to see the same behavior in the RandomForestClassifier which uses the TfIdfVectorizer.

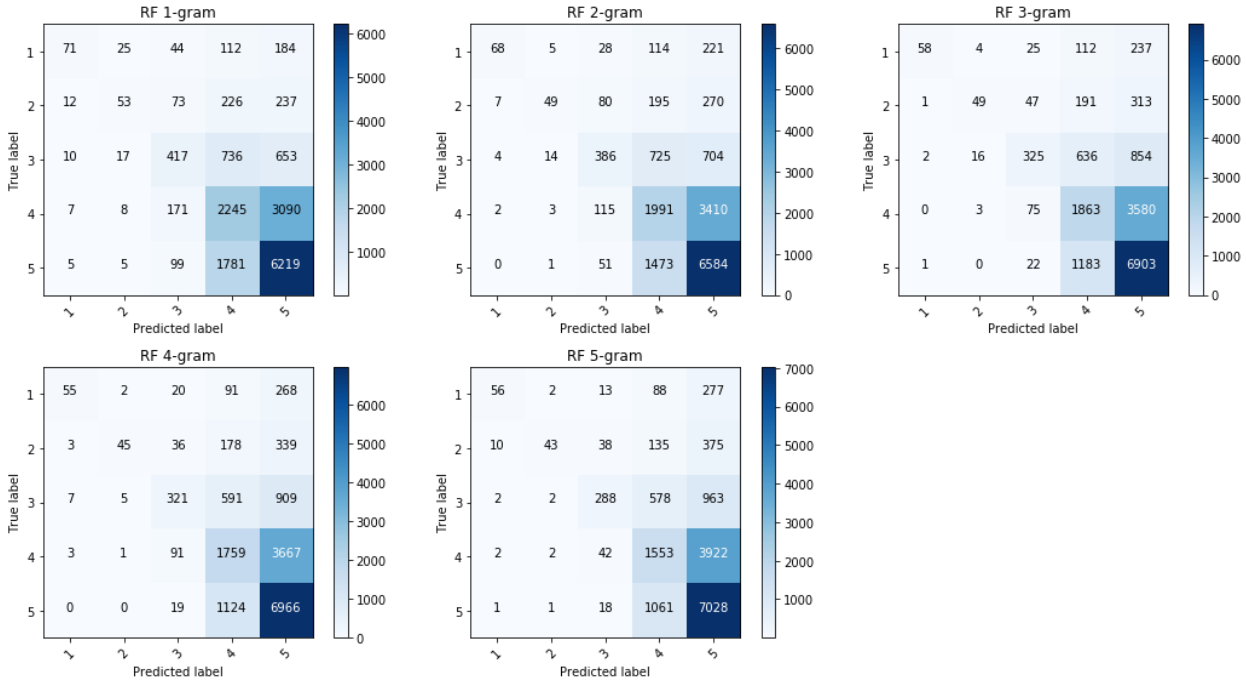
The next set of tests for the RandomForestClassifier uses the ***TfIdfVectorizer***. The results are shown below:





	Runtime (s)	**Accuracy**
1-gram	11.37	0.546306
2-gram	66.16	0.551882
3-gram	175.88	0.545334
4-gram	264.30	0.547763
5-gram	353.24	0.561514

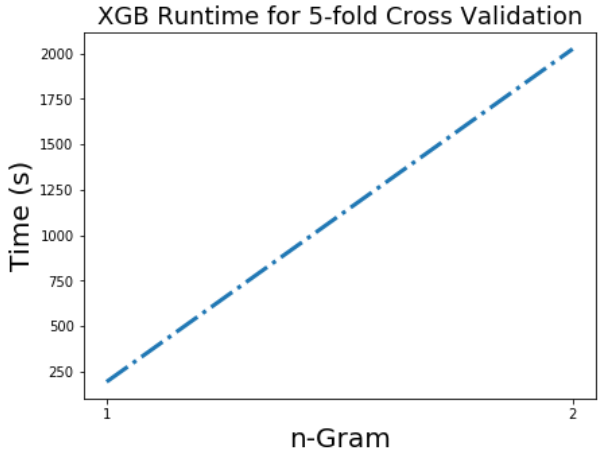
The best classifier in this group contains 5-gram features and an accuracy score of 56%. Its runtime is 353.24. Just as the other models, the runtime with respect to n-grams is roughly linear.

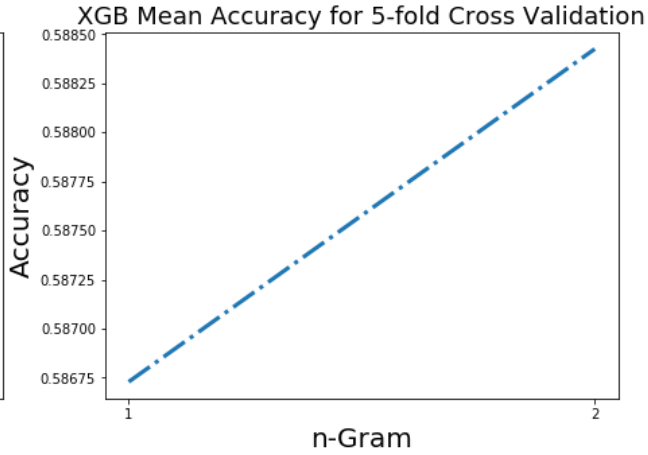


As with the RandomForestClassifier with CountVectorizer, we see roughly the same behavior. However the overall accuracy of the best model here is slightly lower by about 1%.

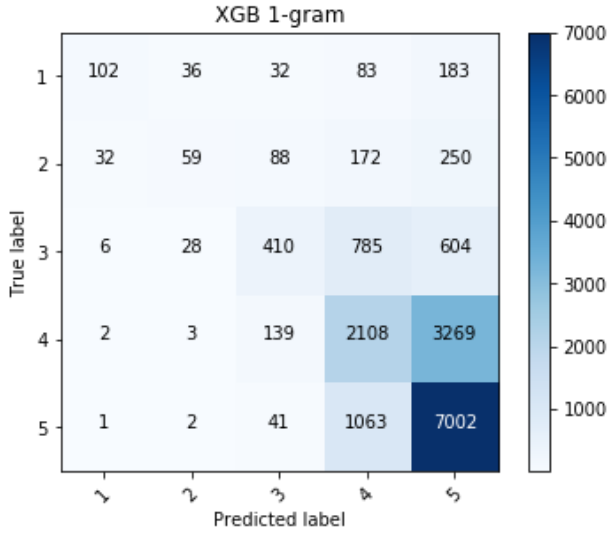
Extreme Gradient Boosting

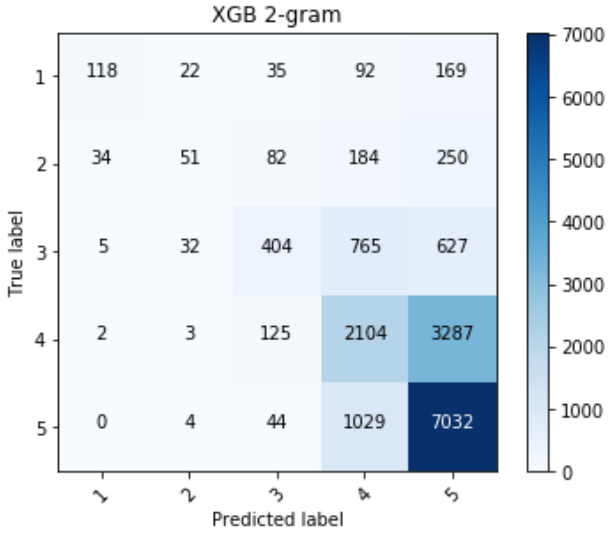
The final classifier that I tested is the extreme gradient boosting classifier, or XGBClassifier. It uses gradient boosted decision trees to significantly improve classification accuracy.





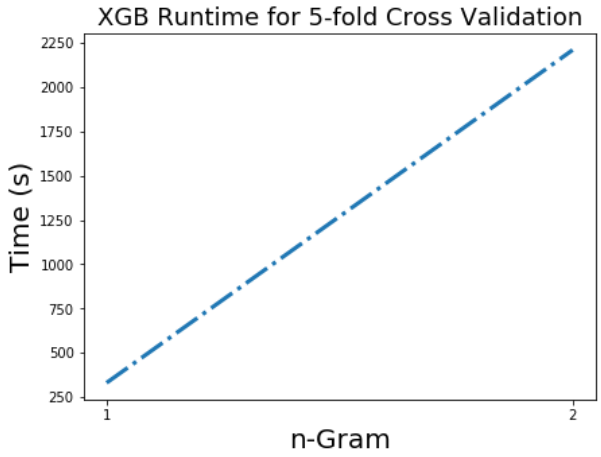
	Runtime (s)	Accuracy
1-gram	195.12	0.586729
2-gram	2023.43	0.588425

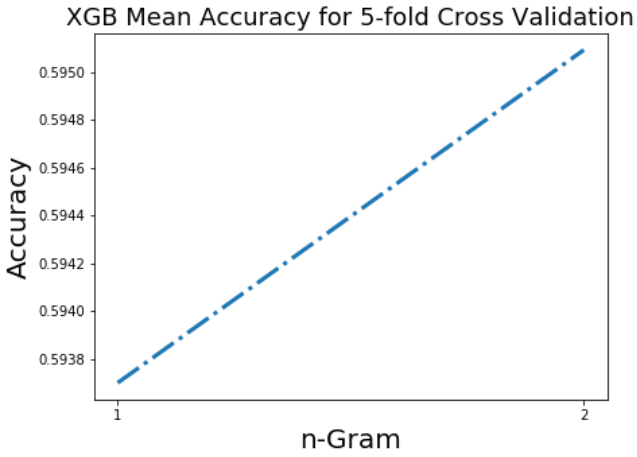




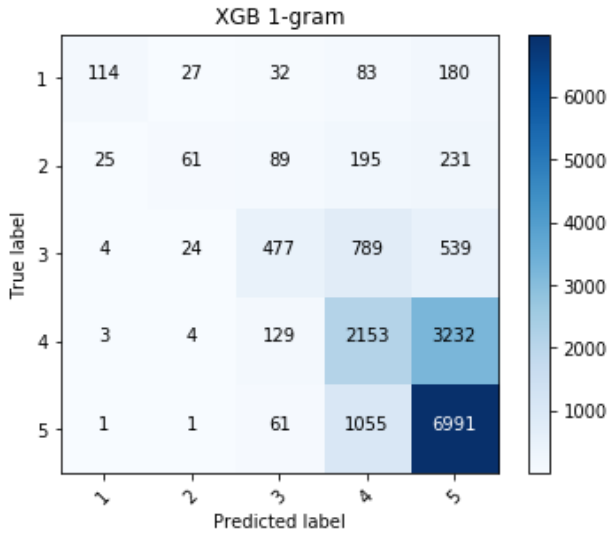
I have only tested unigrams and bigrams for the sake of runtime, as can be seen, the runtime just for running the bigram model is over 2000 seconds. Unfortunately the XGBoost classifier has a lower accuracy than expected due to it being knows as a ground breaking machine learning algorithm. The best model, which includes bigrams, has an accuracy of 58.8% and a runtime of 2023.43 seconds. Like previous models it classifies 1-star ratings poorly and 5-star ratings well.

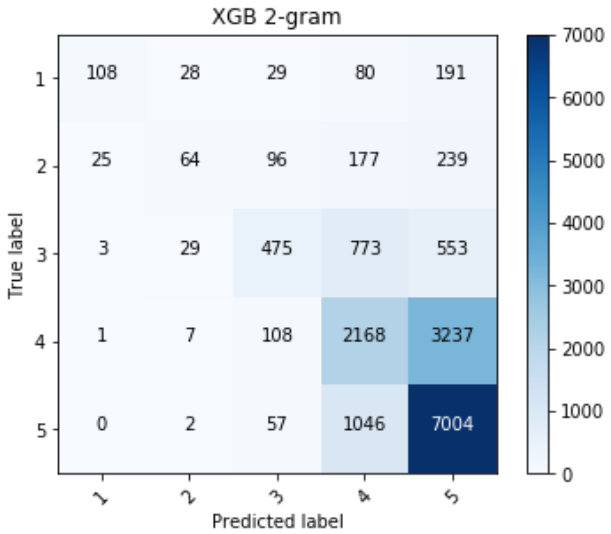
The next set of tests for the XGBoostClassifier uses the ***TfIdfVectorizer***. The results are shown below:





	Runtime (s)	Accuracy
1-gram	331.45	0.593699
2-gram	2208.28	0.595092

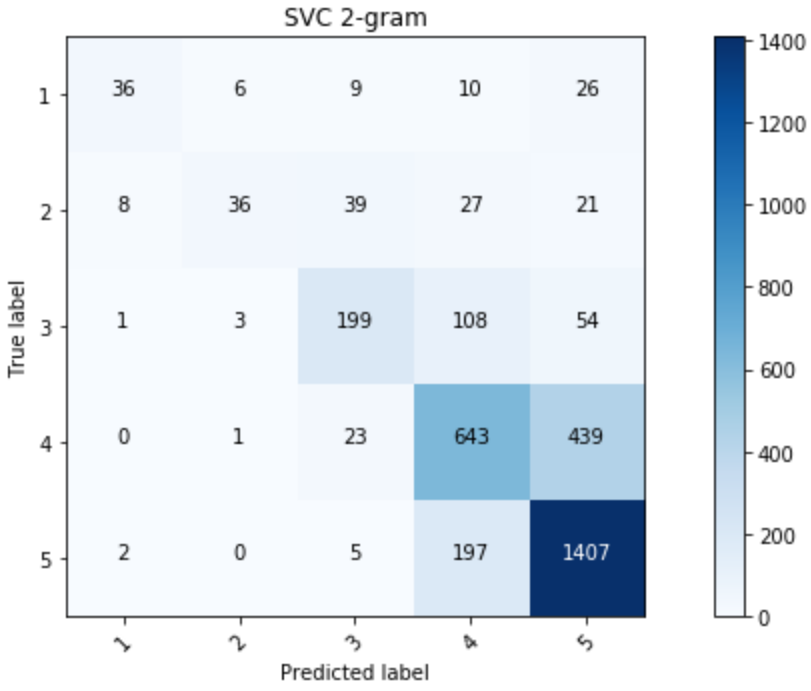




Using the TfIdfVectorizer did not make a significant difference in improving the accuracy of the model. The best model with TfIdfVectorizer has an accuracy of 59.5% and a runtime of 2208.28. Although slightly better, it has a significantly longer runtime by 200 seconds!

VI. Results and Conclusion

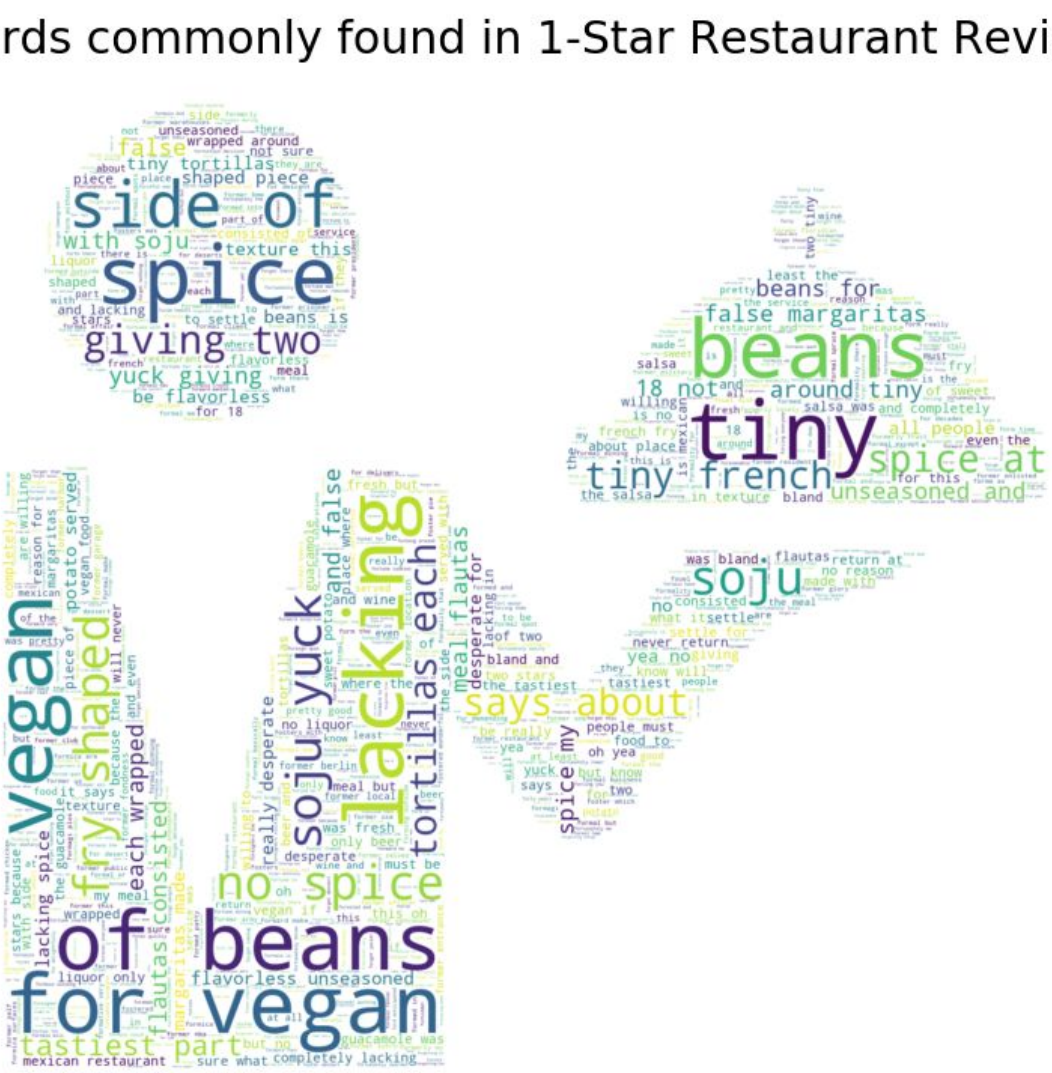
The best model to use is the SVC with bigram TF-IDF vectorization which has a mean accuracy of 63% when tested with 5-fold cross validation. However, after applying this model to the restaurant reviews dataset, it has an even better accuracy score of 70% and a runtime of 1.27s. The following confusion matrix shows how well the model classifies. As mentioned before, due to the huge class balance between 5-star ratings vs the rest, this model will naturally classify 5-star reviews best.



We can also get an idea of what content in the restaurant reviews contribute to high and low restaurant ratings. We can look at review content of the two extremes (5-star and 1-star) with the following word clouds below.

Words commonly found in 5-Star Restaurant Reviews

Some words which stand out in 5-star restaurants are: incredible, easy to, been mentioned, absolutely go, etc. As expected, these would be words for a positive review. These are typically what restaurant owners would like to see in their reviews as it helps them get more customers. Additionally positive reviews with such content written in them will lure in more customers, resulting in better business. The customer is happy and so is the owner!



Possibly due to class imbalance, the results of the 1-star word cloud may seem strange. Words such as: tiny, beans, vegan, soju yuck, etc. show up. However words that do correspond to a negative review or negative experience are somewhat present: lacking, no spice, never return, unseasoned, false margheritas. It's obvious that we need to look into the context of the reviews to understand why certain words without a negative connotation itself are in this word cloud. This is important information for the restaurant owner so he/she can see where the customer is dissatisfied and take appropriate action. Unfortunately for the restaurant, such reviews can deter potential customers from coming, so taking action is a must!