Executive Summary

Introduction:

The website Yelp allows consumers to determine where they want to take their business with an approach that is one part qualitative and one part quantitative. Qualitatively, a restaurant-goer is able to voice their opinion in a review that is up to 5000 characters long. Quantitatively, the reviewer is able to pair this review with a number rating, on a scale of 1 to 5. Although it is more challenging to write a model to predict the sentiment of an individual review than it is to predict manually, the scalability and efficiency that computers offer is unmatched. The goal for this project was to create a model that could allow a computer to predict the star rating given by Yelp users based on their respective text review. Not only should the model give accurate predictions of the star rating given the text, but it should also balance its predictive power with simplicity and interpretability.

Data Background:

The data that we trained our model on was limited to reviews in English that had been written about restaurants in Wisconsin. We received 85,543 reviews, about 50,000 of which were used in our training set.

Data Cleaning:

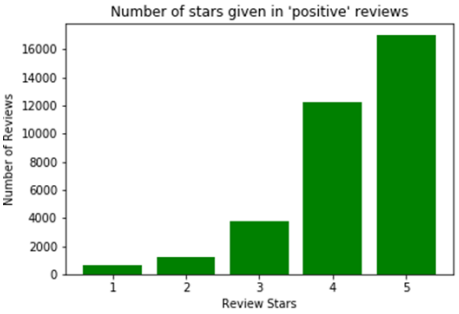
Our dataset did not contain any missing values, so we spent most of our cleaning process checking the validity of our data. One easy thing to check was whether all of the ratings in the training data set were within the range of 1 to 5 stars. Yelp only allows integer ratings from 1 to 5, so anything outside of these boundaries would be incorrect. Upon inspection, the maximum rating in our training dataset was 5 and the minimum value was 1 so there were not any outliers in the response variable. Another thing we considered checking was whether there were any outliers in the number of characters in a review. The maximum number of characters that Yelp allows is 5000, so we know that anything over this bound would be an outlier value. Upon inspection, the maximum number of characters in the training set was 5000 so there were not any outliers in this predictor variable.

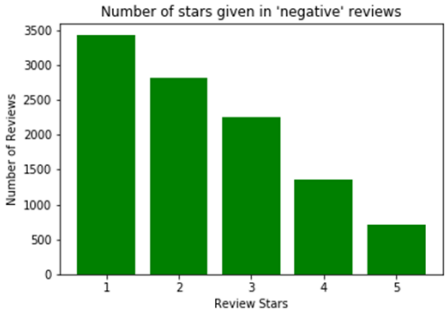
Feature Creation:

While we were given a number of features to use in the training data, we found that these could be improved with some engineering. First, instead of making each provided word its own predictor, we made a list of “positive” and “negative” words and counted the total of each in every review. We considered classifying all words using code as well as classifying words by hand through intuition. Ultimately we decided to use a “Combined Intelligence” approach where we went through by hand to pick out words we thought were positive or negative and then ran code to calculate the correlation between that word and reviews. If the sign that we assigned the word and the sign that the computer assigned the word were different we examined the word to

check whether it was worth keeping. The motivation behind doing this partly by hand was the fear that because there were so many individual word features, some neutral words may have been in many positive reviews in the training set by chance, and would be given a faulty weight.

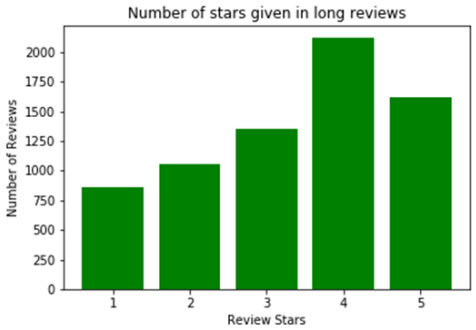
The positive and negative word counts were used in three of the four features we ended up including in our model. One of these features classified a review based on whether it had more "positive" words than "negative" words. Reviews with more "positive" words are classified as "positive" reviews, and reviews with more "negative" words are classified as "negative" reviews. On the right is a comparison between the distribution of ratings for reviews classified as positive (above) vs. ratings for reviews classified as negative (below).





Another feature used was simply the total number of negative words in a review. The third feature created was the total number of positive words in a review divided by the total number of words in a review. Intuitively, shorter reviews with lots of positive words are more likely to be 5 stars. When a restaurant experience is positive, people often don't feel a need to provide much justification for why it was positive. This also normalizes our count of positive words.

This thought on the length of a review ties into our next idea which was testing the correlation of review length with review rating. We calculated the mean rating for long reviews (>200 characters) in the training set to be 3.37 and the mean rating for short reviews (<20 characters) in the training set to be 4.13. From this information and plots of the distribution of short reviews vs. long reviews (below) we decided that length was worth including. It ended up being included in the aforementioned feature of positive words divided by total words.



The last feature we added was constructed based off the idea that restaurants which had been well-reviewed in the past would likely be well-reviewed in the future. While Yelp provides the average rating for different restaurants on its website, this data was not readily available to us in the dataset we were provided. We ended up working around this by calculating the average rating for each restaurant listed in the training set. For each review we encountered in the test set,

we would look up the average rating that restaurant had received in the training set and use that as a predictor. We considered that our model may give this predictor more weight than it is due because it is closely related to the response variable, but upon cross-validation testing it improved our RMSE from .971 to .935.

Other things we considered adding were dummy variables for restaurant names, a predictor for whether the restaurant was reviewed often, and predictors based on the average word length (characters/words) but we didn't find any of them to substantially improve the accuracy of our model. We also considered creating dummy variables from the city data and name data, but the inclusion of these features in our model did not improve our results enough for us to feel justified in trading off the simplicity of using just four features.

Model Selection:

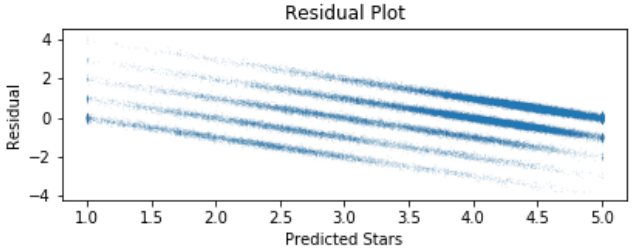
We decided to use multiple linear regression because it had the lowest root mean square error out of all the models we tested. We tested other models and found that they performed much better on the training data, but they were clearly victims of overfitting. We considered spending time adjusting parameters and re-selecting features to try to remedy this issue, but we decided that because we had already passed the benchmark for accuracy we would keep our simple model. We also considered the difficulty we would encounter in trying to produce any sort of interpretation of a neural network or other complex model.

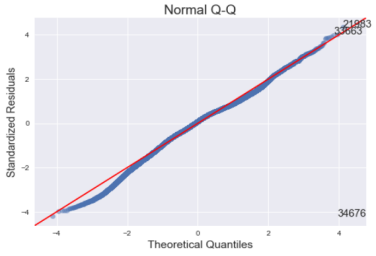
Model	Cross-Validation RMSE with Three Folds
Multiple Linear Regression	.929
Random Forest Classifier	1.23
Random Forest Regressor	1.03
K-Nearest Neighbor	1.20
Gaussian Naive Bayes	1.22
Multi-Layer Perceptron	1.10

Diagnostics and Assumptions:

Although we started out with nine variables, we removed several to both increase the simplicity of our model and to remove collinearity. For instance, we initially had one feature for whether or not a review was positive overall and one feature for the number of positive words but found them to be very collinear.

We created a scatter plot (below) of our predicted stars vs. residuals. We can use this to confirm homoscedasticity as it appears that the variance of the residual is fairly constant. It should be noted that there is a downward trend in the residual plot due to the upper and lower bounds on the number of stars received. Simply put, if our model predicts a rating of 5 stars, it is impossible for the residual for that prediction to be positive. Likewise, if our model predicts a rating of 1 star, it is impossible for the prediction to be negative. A qq-plot (below) was produced to check for normality. As shown from the qq-plot, our model is approximately normal.





Model Interpretation:

Our final model was: Star Rating =1.29+.479*(Positive Overall)+.593*(Restaurant average rating)-0.054*(Negative words)+4.66*(Total positive words/number of words)
Our rule of thumb is: Star Rating = 1+.5*(Positive Overall)+.5*(Restaurant Average Rating)-.05*(Negative words)+5*(Total positive words/number of words)

Because we ended up using a fairly simple linear regression model, the effect that the features have on a review's rating is pretty easy to ascertain. As expected, a review's classification as positive has a positive effect on its predicted score, increasing the predicted rating by .479 if the rating is positive; by .479*(-1.5) if the rating is negative; and by 0 if the rating is neutral. The coefficients for a restaurant's average rating and positive words divided by length are both positive. For every additional average star rating a restaurant has, the rating of the review will increase by .593, and for every increased value of one in positive words divided by length, the rating of the review will increase by 4.66. The negative words feature had less influence on the predicted score than we expected, with a change of -.054 in the predicted rating for each negative word that is counted, all else being equal. However, the p-value for this feature (and all other features) justified their inclusion with P > | t | being about 0.000 for all variables.

Our model had an R-squared and Adj. R-squared of .943, indicating that 94.3% of the variability in the response variable (review rating) can be explained by the model. Furthermore, our model has an F-Statistic of over 20,000, so we can reject the null hypothesis that the fit of an intercept-only model and our model are equal.

Strengths and weakness & Conclusions:

Our model only includes four variables so that means it is simple to calculate and easier to interpret. Another strength of our model is that we have a very high adjusted R^2 value.

The largest weakness of our model is that it trades predictive power for simplicity. We did not spend much time creating features from the raw text of the reviews and our model only looks at the number of occurrences of single words in the review. An example of why this causes us to lose accuracy is that our model would see the sentence "the food was not good" and increase the reviews predicted rating due to the appearance of the word "good."

Member Contributions

Christopher Kardatzke	Data Background, Data Cleaning, Feature Creation
Alex Seo	Feature Creation, Building Model, Model Selection
George Liu	Data Cleaning, Model Interpretation, Model Assumptions
Yichen Sun	Feature Creation, Model Inference