Yelp is an online review site where users can write reviews for restaurants, home services, and auto services. Yelp is a well known name and many people rely on it to determine where they are going to spend their money. This makes the reviews more important for the restaurants because having a higher rating could mean more income or just staying in business. Yelp releases data on the reviews posted on their site. The goal of this paper is to show a linear model indicating how the variables given by Yelp, and newly constructed variables contribute to the star rating left by an individual, and to indicate why the model is useful.

Three models were selected for examination. The multi-linear , Ridge, and Lasso regression models. These models were selected for their capability in handling large amounts of variables. James worked on the multi-linear regression model. Zhelong worked on the Lasso model and also tried to tune alpha which gives the elastic net model, but there is no significant difference. Xiaosheng worked on the Ridge model and also tried the multinomial model which gives a good rmse for training data but a bad rmse for test data, indicating overfitting.

Finally, we choose the LASSO for our model. Because we generate more than 1000 word-predictors, it is impossible to use some selection method provided in R due to computational limitations. Instead, LASSO can help us to choose useful predictors and turn the useless predictors’ coefficients to zero.

After we try all the models, we realize that it is more important to select the correct subset of predictors than choosing a better model. Our first approach is to use python to break all the texts into words and count their appearances, choosing 1000-1500 words with top frequencies as our predictors and generate the word vector of these selected predictors for each text. It turns out that we can get small rmse for training data, for example, up to 0.78 with multinomial and 0.82 with lasso. However, the model totally collapse when we try to predict test data.

In another preprocessing approach, we use the R code that the professor provided to generate a really large matrix, more than 40k columns, counting all the words that appear in the whole train data.

With the huge matrix containing all information of the text, we first try to select predictors using small p-value with the function that the professor provided, and also calculate the correlation of each column with the response column and select columns with extreme correlation, either really positive or really negative. We surprisingly found out both approaches give really similar subset of variables. Nevertheless, the rmse score using these variables is not ideal.

1. new\_words <- colnames(new\_X)

2. colnames(new\_X) <- new\_words

3. new\_pvals <- rep(0,length(new\_words))

4. names(new\_pvals) <- new\_words

5. **for** (i **in** 1:length(new\_words)){

6. ctable <- table(yelp$stars, new\_X[,i])

7. new\_pvals[i] <- fisher.test(ctable, simulate.p.value = T)$p.value

8. }

9. #pick those with smaller p-values

10. length(new\_pvals[which(new\_pvals < 0.0005)])

Then we change to another approach with the same huge matrix. We use two dictionary packages Bing and AFINN to select columns from that huge word count matrix. For words in AFINN we choose words with extreme values, less than -2 or greater than 2, and choose from the matrix that match the words in AFINN dictionary. We also choose words that match the words in Bing. Furthermore, we sum the columns and drop the columns with small total count, we choose word count total above 10 for words from AFINN and word count total above 50 for words from Bing, since we think extreme words in AFINN is likely to be more indicative than just “positive” and “negative” category words in Bing.

1. # Lasso penalty, choosing lambda by cross-validation

2. election.lasso.cv <- cv.glmnet(Xmat, Ymat, nfold=5)

3. slope\_lasso <- coef(election.lasso.cv, s = "lambda.min")[-1]

4. # we pick up those predictors with nonzero coefficients

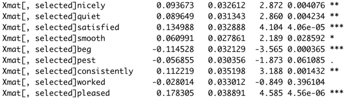
5. selected <- which(slope\_lasso!=0)

6. colnames(Xmat)[selected]

7. # inorder to see the estimation and inference of relevant parameters

8. # we change the model in to MLR and use summary to check the inference

9. summary(lm(Ymat~Xmat[,selected]))



Coefficient: These are some of the predictors, we can see that most of the absolute value of coefficients is between 0.02 and 0.2.

P-values: we can say predictors are useless with p-values more than 0.03.

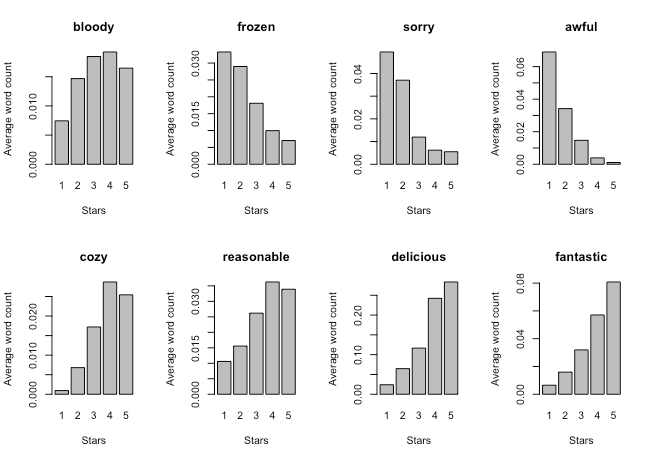
; standard errors is 0.8517;

1. #confidence intervals by taking two-sided 95%

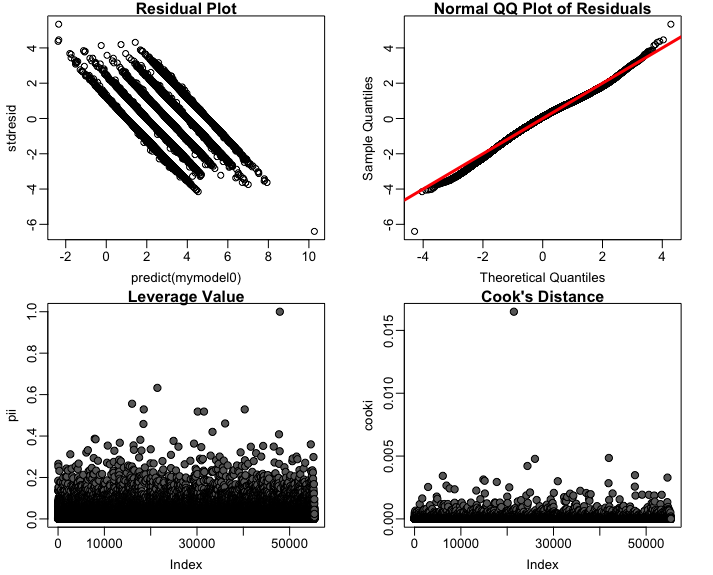
2. confint(lm(Ymat~Xmat[,selected]))



Above are just three samples of confidence intervals of predictor coefficients.



These are histogram plots of 8 sample predictors in our model, which have small p-values and large absolute values of coefficients.



The residuals plot takes the predictions from the multi-linear regression model and plots it against the residuals of the predictions. These points are centered around zero and therefore the linearity assumption is not violated.

There are five parallel lines in the residual plot. For each line, the variance of the residual doesn’t change against ŷ. To some extent, the constant variance assumption is not violated. Intuitively because the outcome variable stars only has five levels, so there are five lines. But this model couldn’t be counted as a perfect linear model, which is also a weakness of using LASSO for prediction. However, using linear model to fit this data is still helpful.

The QQ-plot plots theoretical quantiles against sample quantiles. A line was added to indicate the shape the points should take. Since the points stick close to the line it is concluded that the normality assumption is not violated.

The Cook’s distance plot calculated the Cook’s distance for each data point. While there is one point that has a value much higher than the others it is still below one and so there are no outliers indicated from this.