# An Analysis of Yelp Reviews


## Lauren Flemmer


## Main Question

Can we predict the sentiment of a review from its rating?

## Outline

- Overview of the Data
- Data Visualization
- Model Building
- Model Validation/Selection
- A Closer Look at the Results
- Conclusions

## Data

I am utilizing **2 Yelp datasets** highlighting the Phoenix, AZ metropolitan area * The first describes each business: including its name, location, type, rating, etc. * The second dataset contains each review: including the business it corresponds to, the number of stars, the number of cool/funny/useful votes it recieved, and so on.

I decided to join these two datasets by the variable "business_id" to make things easier…

```
#join the two datasets
yelp <- inner_join(review, business, by = 'business_id') %>% select((1:33))
names(yelp)
```

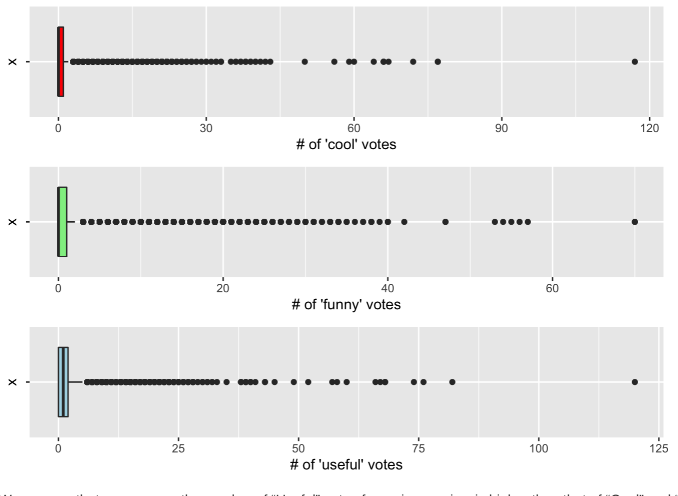
```
##  [1] "X.x"                          "business_blank"
##  [3] "business_categories.x"    "business_city.x"
##  [5] "business_full_address.x"  "business_id"
##  [7] "business_latitude.x"      "business_longitude.x"
##  [9] "business_name.x"          "business_neighborhoods.x"
## [11] "business_open.x"          "business_review_count.x"
## [13] "business_stars.x"         "business_state.x"
## [15] "business_type.x"          "cool"
## [17] "date"                     "funny"
## [19] "review_id"                "reviewer_average_stars"
## [21] "reviewer_blank"           "reviewer_cool"
## [23] "reviewer_funny"           "reviewer_name"
## [25] "reviewer_review_count"    "reviewer_type"
## [27] "reviewer_useful"          "stars"
## [29] "text"                     "type"
## [31] "useful"                   "user_id"
## [33] "X.y"
```

## Exploratory Data Analysis

### Votes (Cool, Funny, Useful)

I first wanted to look at the "Cool", "Funny", and "Useful" votes, which are options that can be chosen by other Yelp users for each review.

```
#visualizing different "votes" via boxplot
cool_plot <- ggplot(data = yelp) +
  geom_boxplot(mapping = aes(x = "", y = cool), fill = "red") +
  ylab("# of 'cool' votes") +
  coord_flip()

funny_plot <- ggplot(data = yelp) +
  geom_boxplot(mapping = aes(x = "", y = funny), fill = "light green") +
  ylab("# of 'funny' votes") +
  coord_flip()

useful_plot <- ggplot(data = yelp) +
  geom_boxplot(mapping = aes(x = "", y = useful), fill = "light blue") +
  ylab("# of 'useful' votes") +
  coord_flip()

grid.arrange(cool_plot, funny_plot, useful_plot, ncol = 1)
```




We can see that on average, the number of "Useful" votes for a given review is higher than that of "Cool" and "Funny" votes.

Let's look a bit deeper into this:

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   0.0000   0.0000   0.0000   0.8682   1.0000 117.0000
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   0.699   1.000  70.000
```

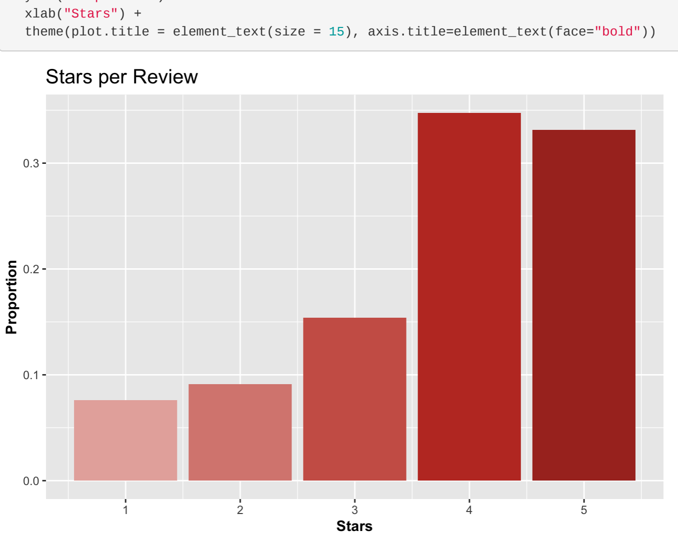
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   1.387   2.000 120.000
```

So, the number of "Useful" votes for a review is generally higher than the number of "Cool" and "Funny" votes.

### Number of Stars

Next, I wanted to visualize the number of stars given to each review, as well as the number of stars each individual business has.
Looking at the stars given to each review:

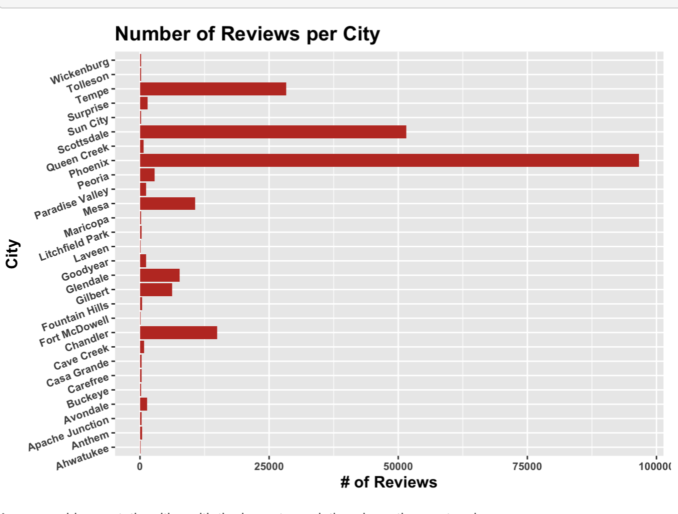
```
redvec <- c("#E6B0AA", "#D98880", "#CD6155", "#C0392B", "#A93226")

ggplot(data = yelp) +
  geom_bar(mapping = aes(x = stars, y = ..prop..), fill = redvec) +
  ggtitle("Stars per Review") +
  ylab("Proportion") +
  xlab("Stars") +
  theme(plot.title = element_text(size = 15), axis.title=element_text(face="bold"))
```




We can see that most reviews recieve a high number of stars.

Now looking at businesses- I wanted to see if a particular geographic area gets a higher number of stars than the others. First I'll show the distribution of reviews in different areas.
I filtered by city, only including cities that have more than 100 reviews, so the barplot is easier to read.

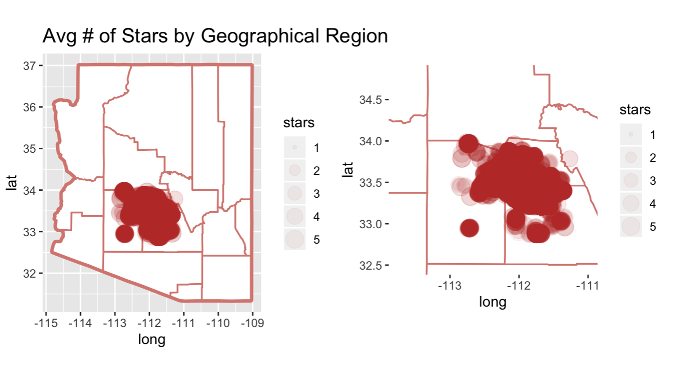
```
group_by_city <- yelp %>%
  group_by(business_city.x) %>%
  summarise(n = n()) %>%
  filter(n > 100) %>%
  arrange(desc(n))


ggplot(data = group_by_city) +
  geom_bar(mapping = aes(x = business_city.x, y = n), stat = "identity", fill = "#C0392B") +
  ggtitle("Number of Reviews per City") +
  xlab("City") +
  ylab("# of Reviews") +
  theme(text = element_text(size = 10, face = "bold"), axis.text.y = element_text(angle = 380, hjust = 1), plot.t
itle = element_text(size = 15), axis.title = element_text(size = 12)) +
  coord_flip()
```




As we would expect, the cities with the largest populations have the most reviews.

Lets elaborate more on this idea, now looking at each individual business and its geographical area. To do this, I plotted each business based on its longitude and latitude on a map of Arizona.

```
avg_review_by_business <- yelp %>% group_by(business_id) %>%
  mutate(avg = mean(stars))

state <- map_data("state")
az <- state[state$region == "arizona", ]

county <- map_data("county")
county <- county[county$region == "arizona", ]

#arizona map
az_map <- ggplot() +
  geom_polygon(data = az, mapping = aes(x = long, y = lat, group = group), color = "#D98880", fill = "white", lwd
= 1.8) +
  geom_polygon(data = county, mapping = aes(x = long, y = lat, group = group), color = "#D98880", fill = "white",
lwd = 0.6) +
  coord_quickmap() +
  geom_point(data = avg_review_by_business, mapping = aes(x = business_longitude.x, y = business_latitude.x, size
= stars), color = "#C0392B", alpha = 0.05) +
  ggtitle("Avg # of Stars by Geographical Region") +
  theme(plot.title = element_text(size = 15))


#zoomed map
zoomed_map <- ggplot() +
  geom_polygon(data = az, mapping = aes(x = long, y = lat, group = group), color = "#D98880", fill = "white", lwd
= 1.8) +
  geom_polygon(data = county, mapping = aes(x = long, y = lat, group = group), color = "#D98880", fill = "white",
lwd = 0.6) +
  coord_quickmap(xlim = c(-113.75, -111), ylim = c(32.5, 34.8)) +
  geom_point(data = avg_review_by_business, mapping = aes(x = business_longitude.x, y = business_latitude.x, size
= stars), color = "#C0392B", alpha = 0.05)

grid.arrange(az_map, zoomed_map, ncol = 2)
```



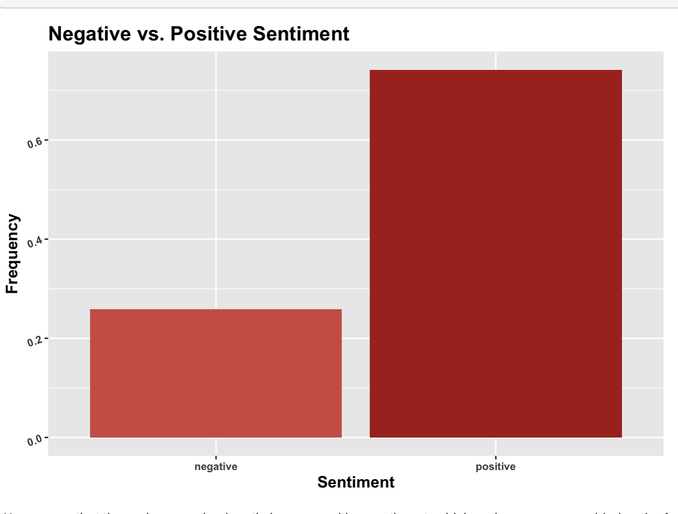

### Sentiments

Next, I wanted to explore the sentiment of each review.

```
yelp_tibble <- tibble(text = yelp$text, review_id = yelp$review_id) %>% mutate(text = as.character(text), review_
index = row_number())

#tokenize reviews
token <- yelp_tibble %>% unnest_tokens(word, text, to_lower = TRUE)
#remove stopwords
no_stopwords <- token %>% anti_join(get_stopwords())

#sentiments
bing <- get_sentiments("bing")
sentiments <- no_stopwords %>% inner_join(bing)
```

```
#join sentiment dataset w/ main dataset
all <- inner_join(yelp, sentiments, by = "review_id")
```

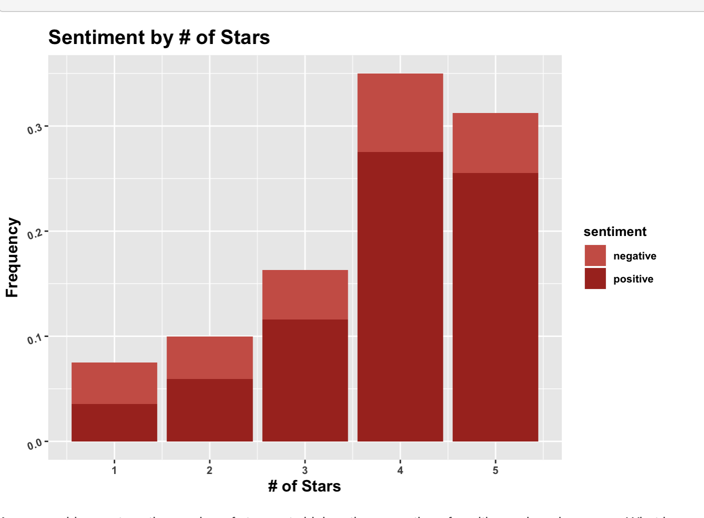
```
grouped_by_sentiment <- sentiments %>% group_by(sentiment) %>% mutate(count = n())

#frequency of sentiments
ggplot(data = grouped_by_sentiment) +
  geom_bar(mapping = aes(x = sentiment, y = (..count..)/sum(..count..)), fill = c("#CD6155", "#A93226")) +
  ggtitle("Negative vs. Positive Sentiment") +
 xlab("Sentiment") +
  ylab("Frequency") +
  theme(text = element_text(size = 10, face = "bold"), axis.text.y = element_text(angle = 380, hjust = 1), plot.t
itle = element_text(size = 15), axis.title = element_text(size = 12))
```




We can see that the reviews predominantly have a positive sentiment, which makes sense, considering the frequency of "stars" given, (mostly 4's and 5's), as we explored previously.

Next, I wanted to look at the overall sentiment of each review, compared to the number of stars it was given, in order to see if there's any discrepancies.

```
by_id <- all %>% group_by(review_id) %>% mutate(votes = cool + funny + useful)

ggplot(data = by_id) +
  geom_bar(mapping = aes(x = stars, y = (..count..)/sum(..count..), fill = sentiment)) +
ggtitle("Sentiment by # of Stars") +
 xlab("# of Stars") +
  ylab("Frequency") +
  theme(text = element_text(size = 10, face = "bold"), axis.text.y = element_text(angle = 380, hjust = 1), plot.t
itle = element_text(size = 15), axis.title = element_text(size = 12)) +
scale_fill_manual(values=c("#CD6155", "#A93226"))
```




As we would expect, as the number of stars gets higher, the proportion of positive reviews increases. What is surprising, however, is that for very low star ratings, such as *1 star*, the proportion of positive reviews is still ~50%.

```
pos_sentiment <- by_id %>% filter(sentiment == "positive")
neg_sentiment <- by_id %>% filter(sentiment == "negative")

#5 number summary for stars
summary(pos_sentiment$stars)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   4.000   3.883   5.000   5.000
```

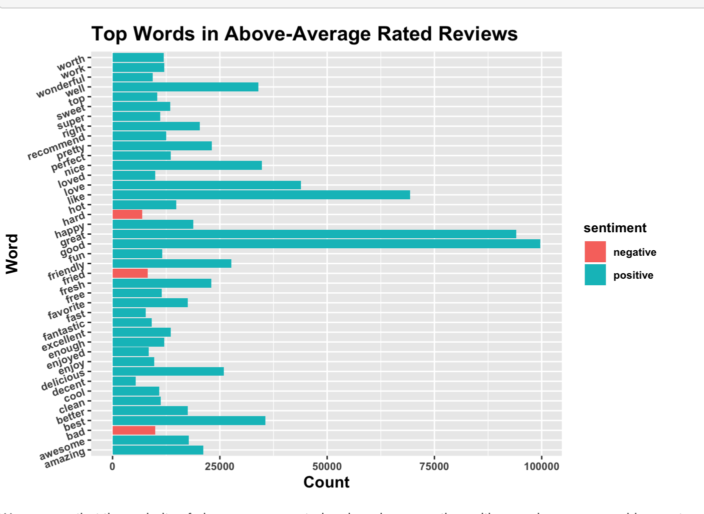
```
summary(neg_sentiment$stars)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   4.000   3.272   4.000   5.000
```

So, the mean star rating is quite a bit higher for *positive* reviews than it is for *negative* reviews.

Looking at specific words, I decided to only look at reviews with higher ratings. So, I only looked at reviews that had above 3.272 stars, the average.

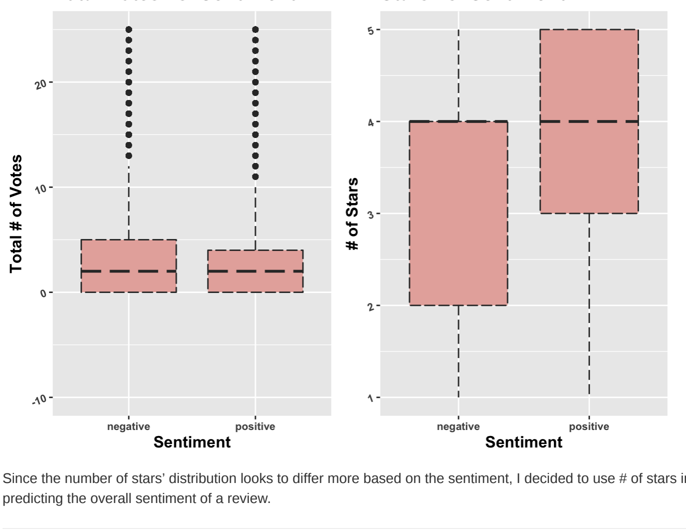
```
mean_stars <- mean(all$stars)

above_avg_stars <- all %>% group_by(word) %>% mutate(word_count = n())  %>% filter(stars > mean_stars, word_count
> 10000)

ggplot(data = above_avg_stars) +
  geom_bar(mapping = aes(x = word, ..count.., fill = sentiment)) +
  coord_flip() +
  xlab("Word") +
  ylab("Count") +
  ggtitle("Top Words in Above-Average Rated Reviews") +
  theme(text = element_text(size = 10, face = "bold"), axis.text.y = element_text(angle = 380, hjust = 1), plot.t
itle = element_text(size = 15), axis.title = element_text(size = 12))
```




We can see that the majority of above-average rated reviews have mostly positive words, as one would expect.

## Classification

```
#boxplot of sentiment and total votes
sentiment_box_1 <- ggplot(data = by_id) +
  geom_boxplot(mapping = aes(x = sentiment, y = votes), linetype = 5, fill = "#E6B0AA") +
  ylab("Total # of Votes") +
  xlab("Sentiment") +
  ylim(c(-10, 25)) +
  ggtitle("Total Votes vs. Sentiment") +
  theme(text = element_text(size = 10, face = "bold"), axis.text.y = element_text(angle = 380, hjust = 1), plot.t
itle = element_text(size = 15), axis.title = element_text(size = 12))

#boxplot of sentiment and stars
sentiment_box_2 <- ggplot(data = by_id) +
  geom_boxplot(mapping = aes(x = sentiment, y = stars), linetype = 5, fill = "#E6B0AA") +
  ylab("# of Stars") +
  xlab("Sentiment") +
  ggtitle("Stars vs. Sentiment") +
  theme(text = element_text(size = 10, face = "bold"), axis.text.y = element_text(angle = 380, hjust = 1), plot.t
itle = element_text(size = 15), axis.title = element_text(size = 12))

grid.arrange(sentiment_box_1, sentiment_box_2, ncol = 2)
```




Since the number of stars' distribution looks to differ more based on the sentiment, I decided to use # of stars in my logistic regression model for predicting the overall sentiment of a review.

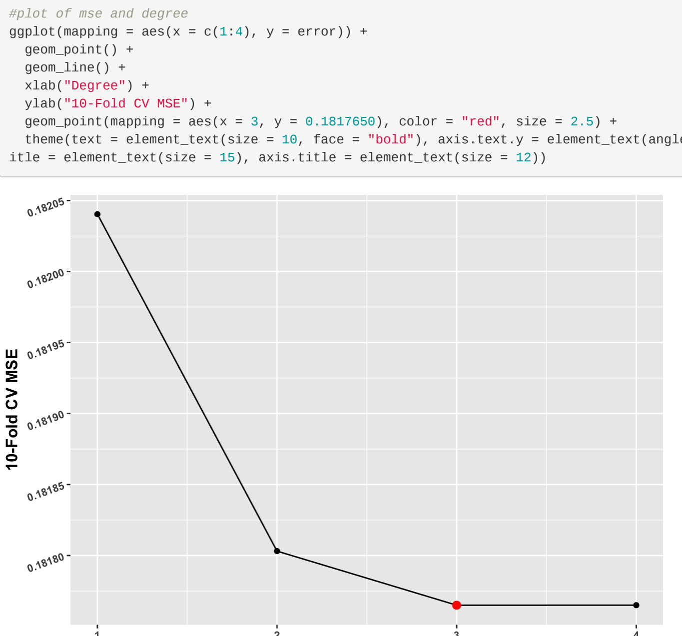
### 10-Fold Cross Validation for Logistic Regression

In order to choose the most accurate model, I'm using a 10-fold cross validation Because the polynomial degree of the "stars" variable needs to be less than the number of unique values, and the star ratings are 1-5, the degree must be less than or equal to 4 for this model.

```
set.seed(100)

#transform sentiments to be 0 (negative) or 1 (positive)
all <- all %>% mutate(binary_sentiment = ifelse(sentiment == "positive", 1, 0))

error <- rep(1:4)

#try polynomials 1-4
for (i in 1:4) {

logisticreg <- glm(binary_sentiment ~ poly(stars, i), family = binomial(), data = all)
#mse vector
error[i] <- cv.glm(all, logisticreg, K = 10)$delta[1]

}

error
```

```
## [1] 0.1820404 0.1818031 0.1817650 0.1817650
```

```
#plot of mse and degree
ggplot(mapping = aes(x = c(1:4), y = error)) +
  geom_point() +
  geom_line() +
  xlab("Degree") +
  ylab("10-Fold CV MSE") +
  geom_point(mapping = aes(x = 3, y = 0.1817650), color = "red", size = 2.5) +
  theme(text = element_text(size = 10, face = "bold"), axis.text.y = element_text(angle = 380, hjust = 1), plot.t
itle = element_text(size = 15), axis.title = element_text(size = 12))
```




### Final Logistic Regression Model

Therefore, the final model is the one with the lowest Mean Squared Error: $sentiment = \alpha + stars + stars^2 + stars^3$

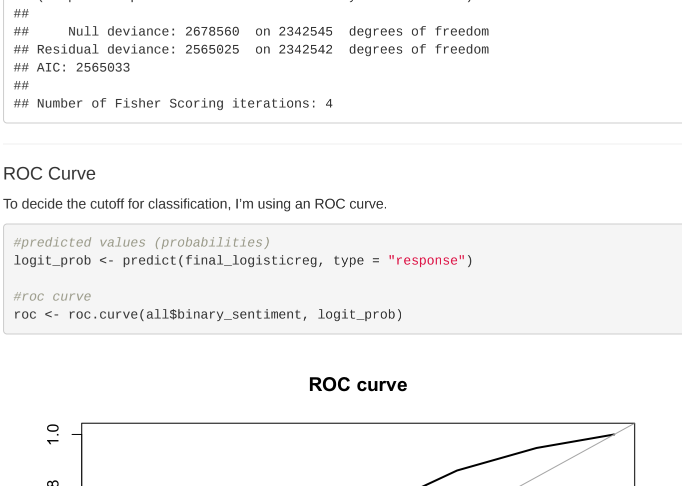
```
final_logisticreg <- glm(binary_sentiment ~ poly(stars, 3), family = binomial(), data = all)
summary(final_logisticreg)
```

```
##
## Call:
## glm(formula = binary_sentiment ~ poly(stars, 3), family = binomial(),
##     data = all)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8436  -1.1373   0.6354   0.6936   1.2180
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.101e+00  1.557e-03  707.18   <2e-16 ***
## poly(stars, 3)1  7.164e+02  2.247e+00  318.86   <2e-16 ***
## poly(stars, 3)2 -1.345e+02  2.290e+00  -58.74   <2e-16 ***
## poly(stars, 3)3 -4.637e+01  2.218e+00  -20.91   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2678560  on 2342545  degrees of freedom
## Residual deviance: 2565025  on 2342542  degrees of freedom
## AIC: 2565033
##
## Number of Fisher Scoring iterations: 4
```

#### ROC Curve

To decide the cutoff for classification, I'm using an ROC curve.

```
#predicted values (probabilities)
logit_prob <- predict(final_logisticreg, type = "response")

#roc curve
roc <- roc.curve(all$binary_sentiment, logit_prob)
```




```
roc
```

```
## Area under the curve (AUC): 0.626
```

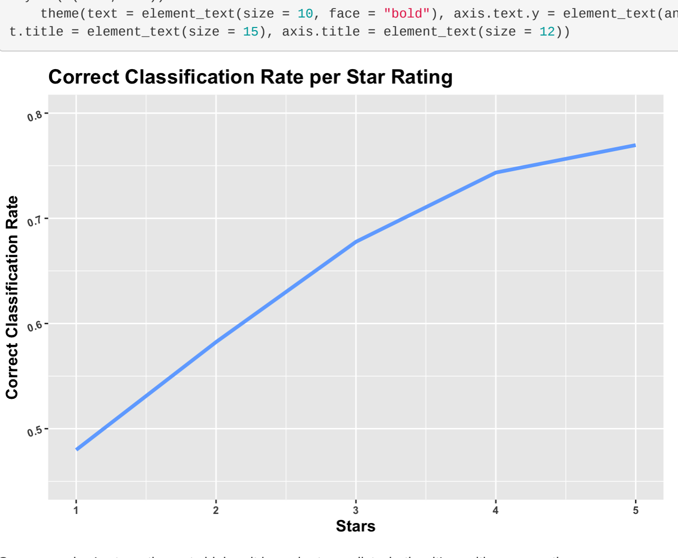
```
#find optimal cutoff
cutoff <- optimalCutoff(all$binary_sentiment, logit_prob)
cutoff
```

```
## [1] 0.4772052
```

#### Misclassification Rate

```
#classification
classification <- ifelse(logit_prob > cutoff, "1", "0") %>% as.factor()
all %>% mutate(classification = classification)

#misclassification rate
mean(all$binary_sentiment != classification)
```

```
incorrect_class <- all %>% group_by(stars) %>% summarize(prop_incorrect = (sum(binary_sentiment != classificatio
n))/ sum((binary_sentiment == classification) | (binary_sentiment != classification)))
correct_class <- all %>% group_by(stars) %>% summarize(prop_correct = (sum(binary_sentiment == classification))/
sum((binary_sentiment == classification) | (binary_sentiment != classification)))


ggplot(data = correct_class) +
  geom_line(mapping = aes(x = stars, y = prop_correct), color = "#70ACFF", lwd = 1.3) +
  xlab("Stars") +
  ylab("Correct Classification Rate") +
  ggtitle("Correct Classification Rate per Star Rating") +
  ylim(c(0.45, 0.8)) +
    theme(text = element_text(size = 10, face = "bold"), axis.text.y = element_text(angle = 380, hjust = 1), plo
t.title = element_text(size = 15), axis.title = element_text(size = 12))
```




So, as a review's star rating gets higher, it is easier to predict whether it's positive or negative.