Animal Crossing is one of the most relaxing and friendly game series in the market today. Animal Crossing ignores many common video game aspects, such as competition and challenge, and let's you enjoy and interact with your environment and friends. The most recent version New Horizons has sold in 31.18 million copies as of Dec 2020, making it one of the best selling games on the switch.

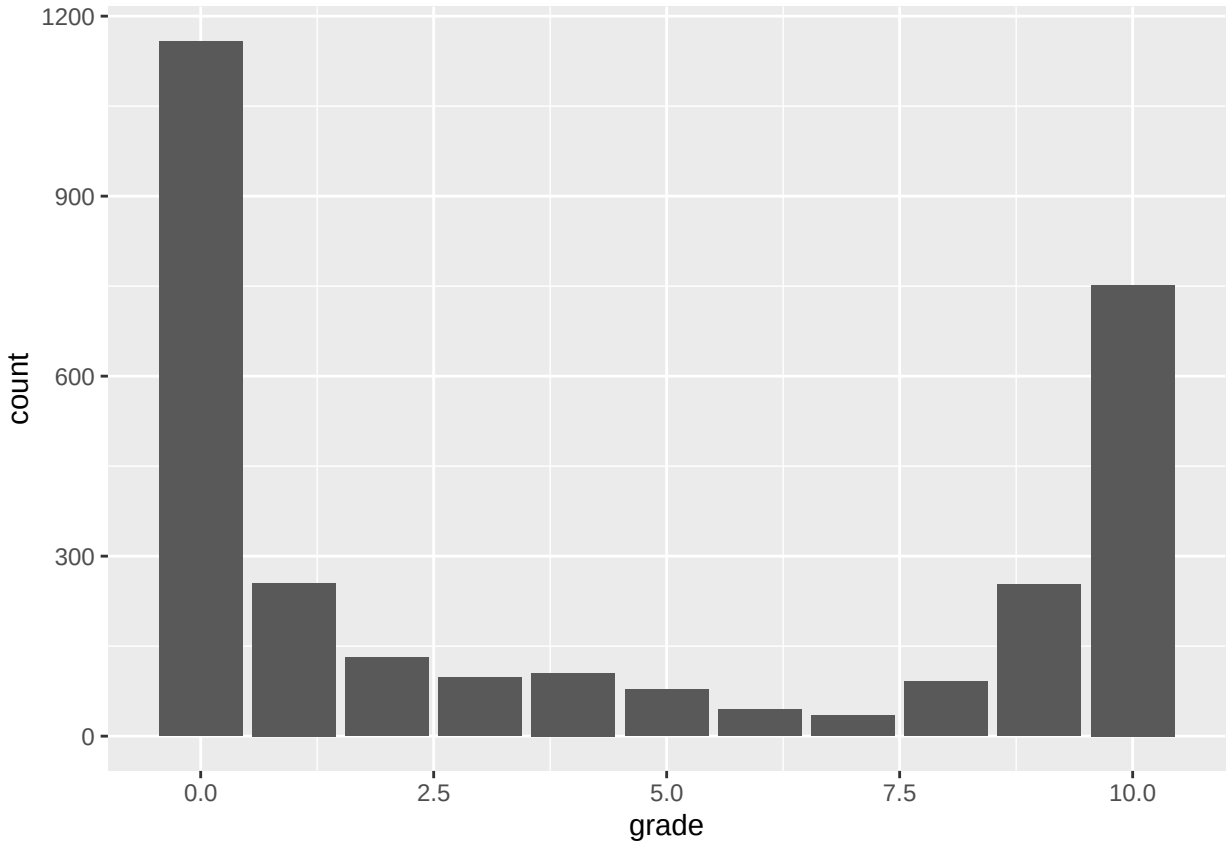
In this report I will be analyzing reviews of Animal Crossing and build a logistic regression model to predict ratings based on those reviews. The data is from tidytuesday dataset in the rfordatascience repo.

```
df <- read_tsv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-05-
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   grade = col_double(),
##   user_name = col_character(),
##   text = col_character(),
##   date = col_date(format = "")
## )
```

To begin we should look at the distribution of the ratings.

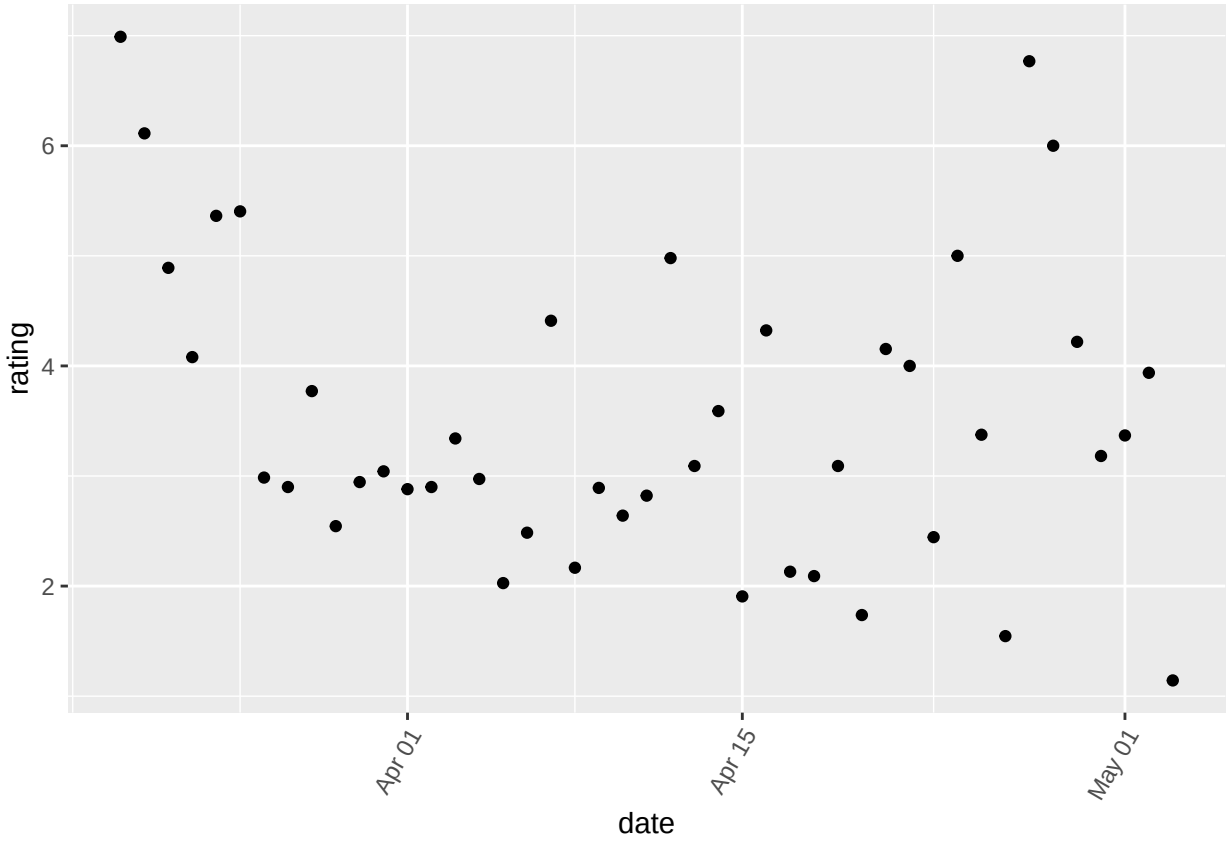
```
df %>% ggplot(aes(x=grade)) + geom_bar()
```



The plot shows how reviews can be polarizing. People who really love the game or really hate it post the most while those who are less aggressive won't give a review

Another good plot to look at is how review change over time

```
df %>% group_by(date) %>%
  summarise(rating=mean(grade)) %>%
  ggplot(aes(x=date, y=rating)) + geom_point() + theme(axis.text.x = element_text(angle = 60, hjust = 1
```



From March to June the average reviews mostly range between 2 and 6.

We can sample the reviews

```
df %>% filter(grade == 10) %>% pull(text) %>% sample(3)
```

```
## [1] "Players giving this game a zero and calling themselves Animal Crossing fans are a complete joke
## [2] "Amazing new addition to the already awesome series! I love how much work the dev and design tea
## [3] "This review contains spoilers, click expand to view.
```

Good review comment on how the game looks great and how relaxing the sound track is.

```
df %>% filter(grade == 0) %>% pull(text) %>% sample(3)
```

```
## [1] "I actually just made a metacritic account just to review this game. This would be an almost per
## [2] "Fix the game Nintendo. I bought it for me and my wife to enjoy i started the game first only to
## [3] "Zero is as low as I can go? More like -100 Really Nintendo? One island per Switch? The first us
```

Bad reviews comment on how there is only one island per console, and how there is a lot of greed from Nintendo.

Text preprocessing It seem like there was some issue with the scraping of the data as some reviews are repetitive. Those can be removed by filtering out the reviews with *Expand* at the end. Also a lot of the content in the reviews don't carry any sentiment so they will be removed as well.

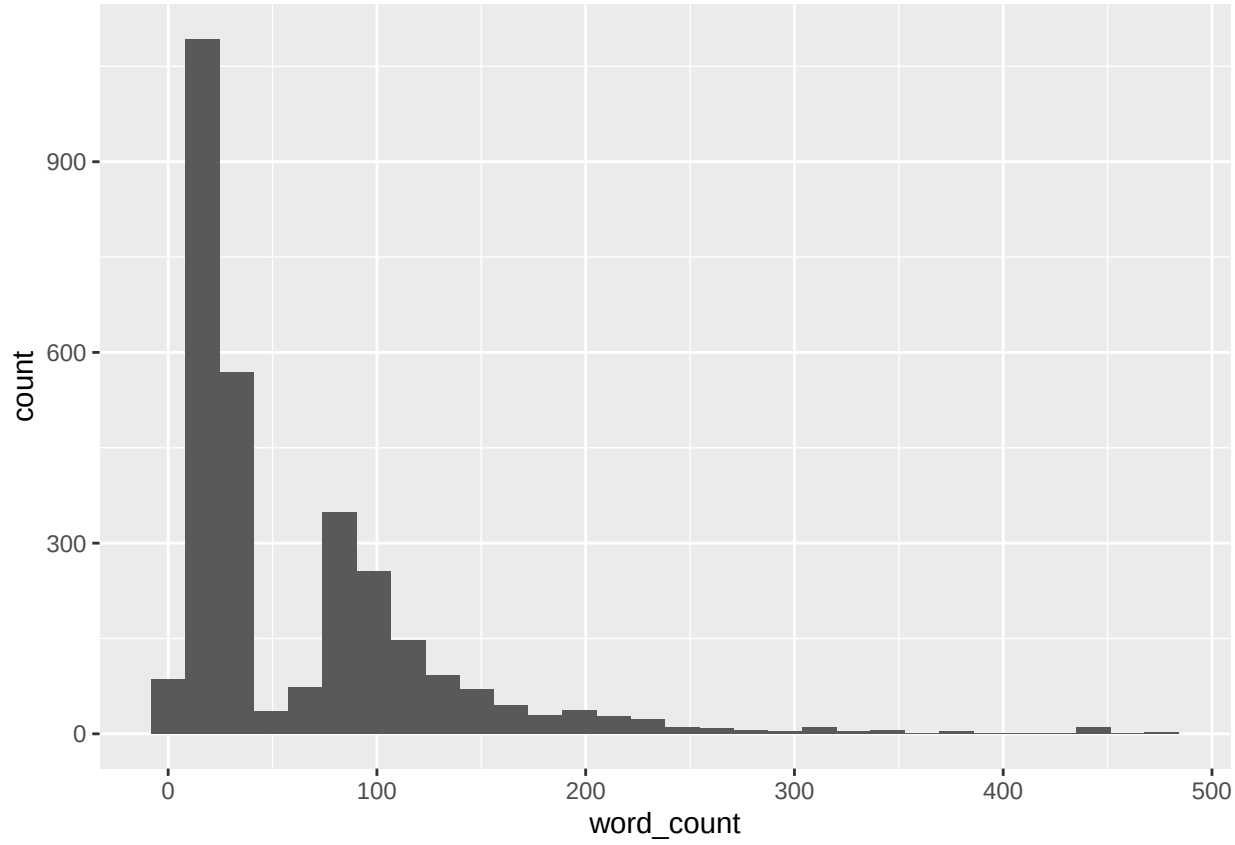
- Remove punctuation
- Remove stop words
- Remove words that are one letter long
- Remove white spaces

```
stop_word_list <- c(stopwords(), 'user', 'far', 'gets', 'ac', 'want', 'day', 'things', 'que', 'always',
df <- df %>% mutate(org_text = text,
                    text = str_remove(text, 'Expand$'),
                    text = str_to_lower(text) %>%
                           gsub(pattern = '\\W', replace=' ') %>%
                           removeWords(stop_word_list) %>%
                           gsub(pattern='\\b[a-z]\\b{1}', replace=' ') %>%
                           stripWhitespace(),
                    len = sapply(strsplit(text, ' '), length))
```

Some of those reviews were long while others are short, lets look at a histogram of the distribution of reviews

```
df %>% unnest_tokens(word, text) %>%
  count(user_name, name='word_count') %>%
  ggplot(aes(word_count)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We can also look at the difference in sentiment

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
words = df %>% unnest_tokens(word, text) %>% count(word, name='word_count')

words %>% inner_join(get_sentiments('afinn')) %>%
          inner_join(get_sentiments('nrc')) %>%
          inner_join(get_sentiments('bing')) %>%
          acast(word ~ sentiment, value.var='word_count', fill=0) %>%
  comparison.cloud(colors=c('red', 'blue'), max.words = 50)
```

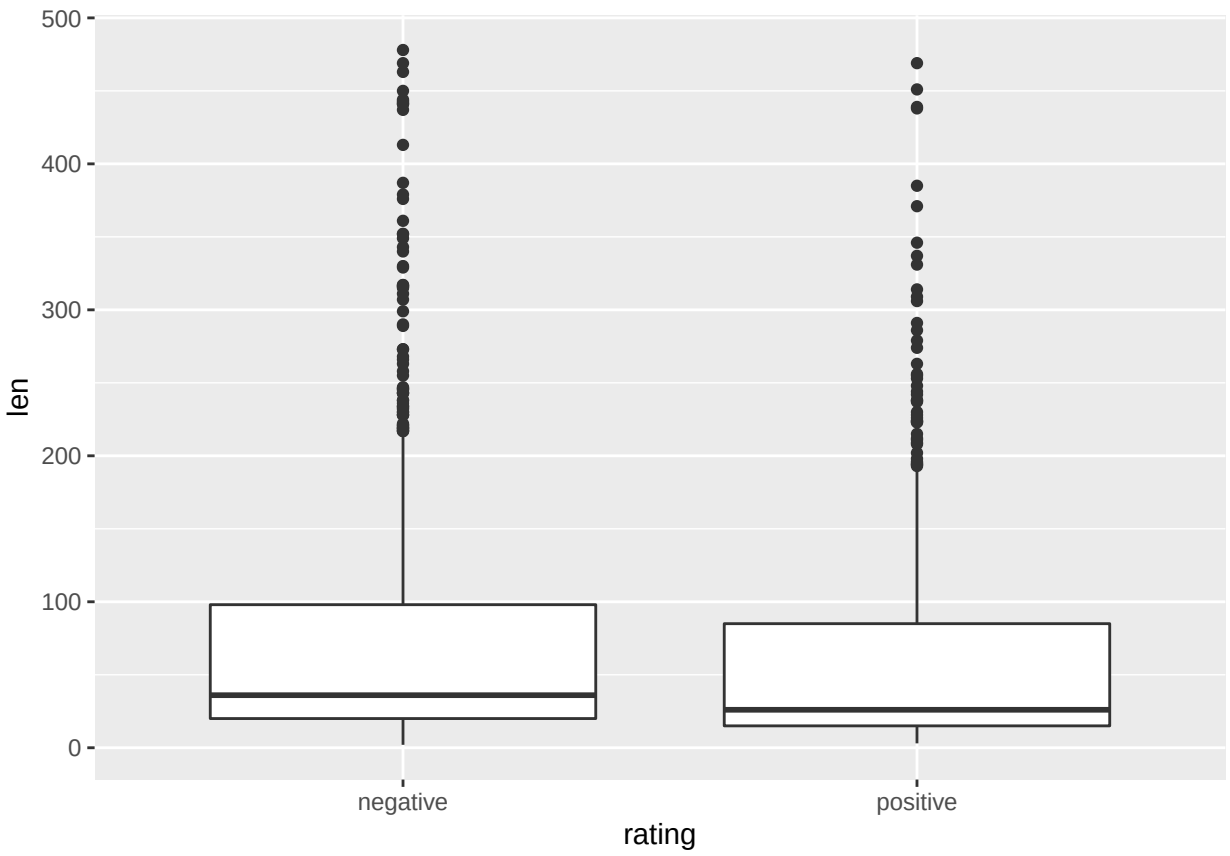
```
## Joining, by = "word"
```

```
## Joining, by = "word"
```

```
## Joining, by = c("word", "sentiment")
```

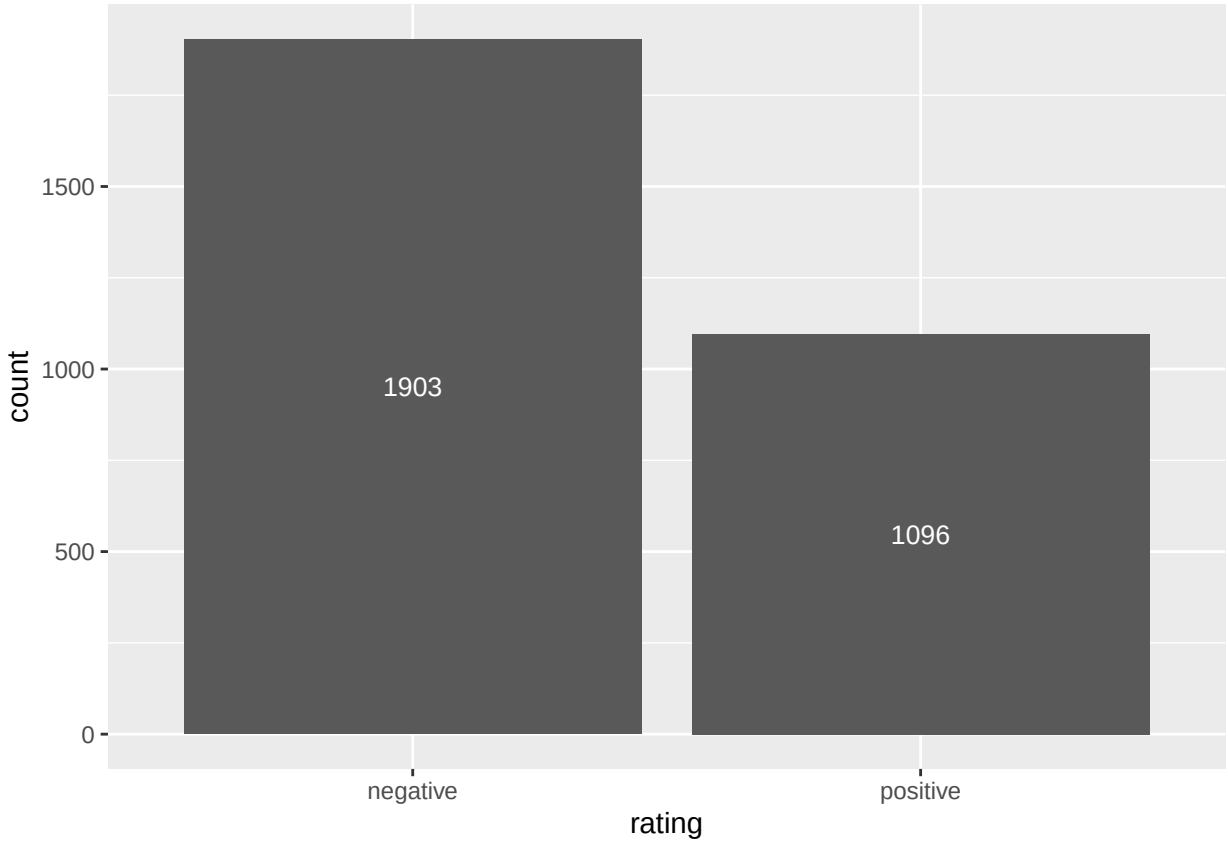
Creating the labels. Grades that are greater than 7 will be considered a positive review and those less than 7 negative.

```
df <- df %>% mutate(rating = case_when(grade > 7 ~ 'positive', TRUE ~ 'negative'))
df %>% select(rating, len) %>% ggplot(aes(x=rating, y=len)) + geom_boxplot()
```




```
df %>% ggplot(aes(rating)) + geom_bar(sstat='count') + stat_count(geom = "text", colour = "white", size
```

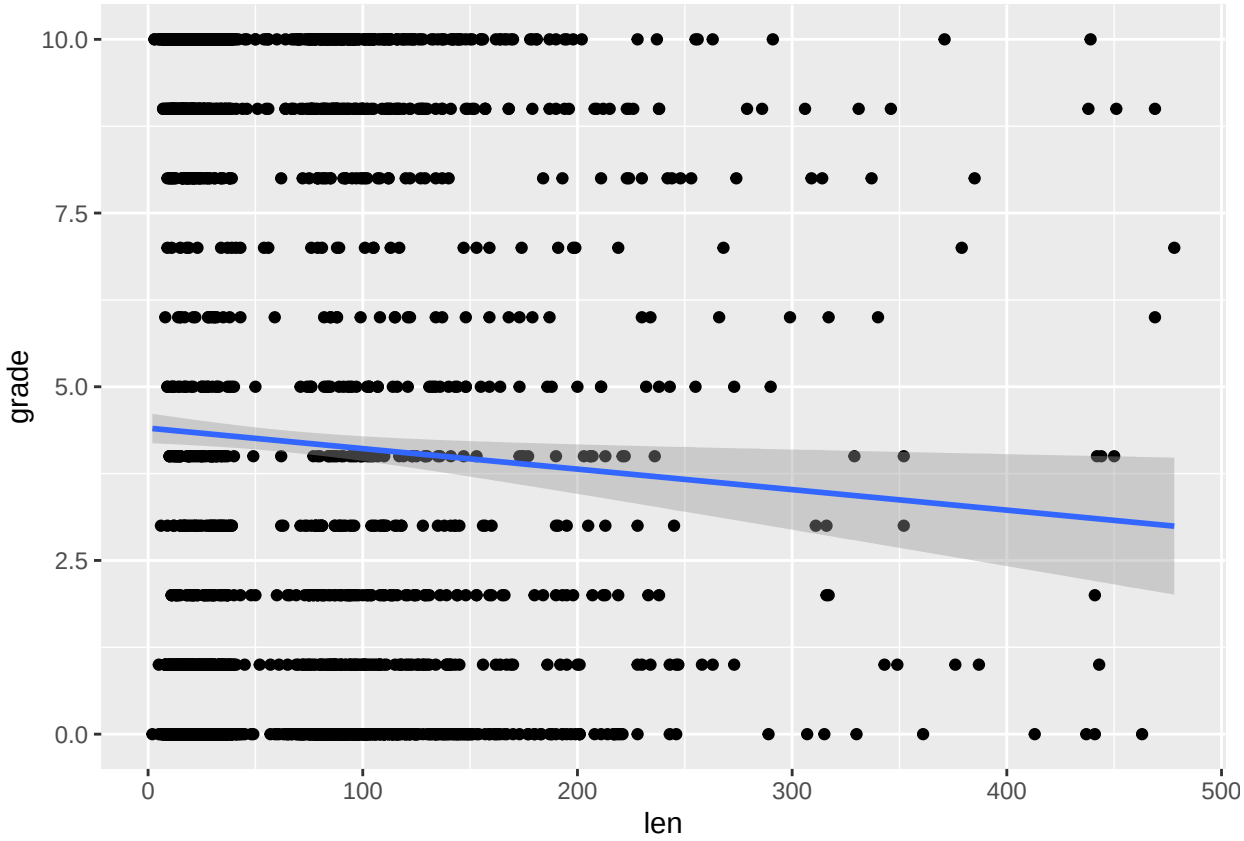
```
## Warning: Ignoring unknown parameters: sstat
```



As we saw from the first bar plot there are more negative reviews that positive ones; however the different isn't too larger for unbalance to be an issue.

```
df %>% ggplot(aes(x=len, y=grade)) + geom_point() + geom_smooth(method='lm')
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
lmodel <- lm(grade ~ len, data=df)
summary(lmodel)
```

```
##
## Call:
## lm(formula = grade ~ len, data = df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.399 -4.180 -2.097  5.614  6.891
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.404642   0.109860  40.093   <2e-16 ***
## len         -0.002951   0.001197  -2.465   0.0138 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.346 on 2997 degrees of freedom
## Multiple R-squared:  0.002023,   Adjusted R-squared:  0.00169
## F-statistic: 6.074 on 1 and 2997 DF,  p-value: 0.01378
```

An R^2 of 0.0012 makes it very clear that there is virtually no relationship with review length and the grade one gives it. The scatter plot verified that conclusion. These results also imply linear regression might not be good for this problem.

Model building The model will be trained on a 70% of the data and tested on the remaining 30%.

```
df_split <- initial_split(df, strata=rating)
df_train <- training(df_split)
df_test <- testing(df_split)
```

Before feeding the data into the model we need to quantify it for matrix operations. The process is:

- Tokenization for text
- Text filtering for text that are too frequent
- Term frequency-inverse document frequency
- Centering and scaling

```
text_rec <- recipe(rating ~ text, data = df_train) %>%
  step_tokenize(text) %>%
  step_tokenfilter(text, max_tokens = 600) %>%
  step_tfidf(text) %>%
  step_normalize(all_predictors())
```

Build classification model with logistic regression mixture = 1 to train a lasso model

Since our labels is categorical, logistic regression is be a good fit. The glmnet R package fits a generalized linear model via penalized maximum likelihood. This method of estimating the logistic regression slope parameters uses a penalty on the process so that less relevant predictors are driven towards a value of zero.

$$\Pr(\text{rating} = \textit{positive} \mid \text{x1, x2, x3 ...}) = \frac{\exp(\beta_0 + \beta_1 \text{amazing} + \beta_2 \text{greed} + \cdots + \beta_{12} \text{excellent})}{1 + \exp(\beta_0 + \beta_1 \text{amazing} + \beta_2 \text{greed} + \cdots + \beta_{12} \text{excellent})}$$

```
model <- logistic_reg(penalty = tune(), mixture = 1) %>% set_engine('glmnet')
model_wf <- workflow() %>%
  add_recipe(text_rec) %>%
  add_model(model)
```

Model tuning Before we fit this model, we need to set up a grid of penalty values to tune. Since we have only one hyperparameter to tune here, we can set the grid up using a one-column tibble with 30 values. Similarly we will bootstrap from the training set to tune the model.

```
grid <- grid_regular(penalty(), levels=20)
folds <- bootstraps(df_train, strata = rating)
```

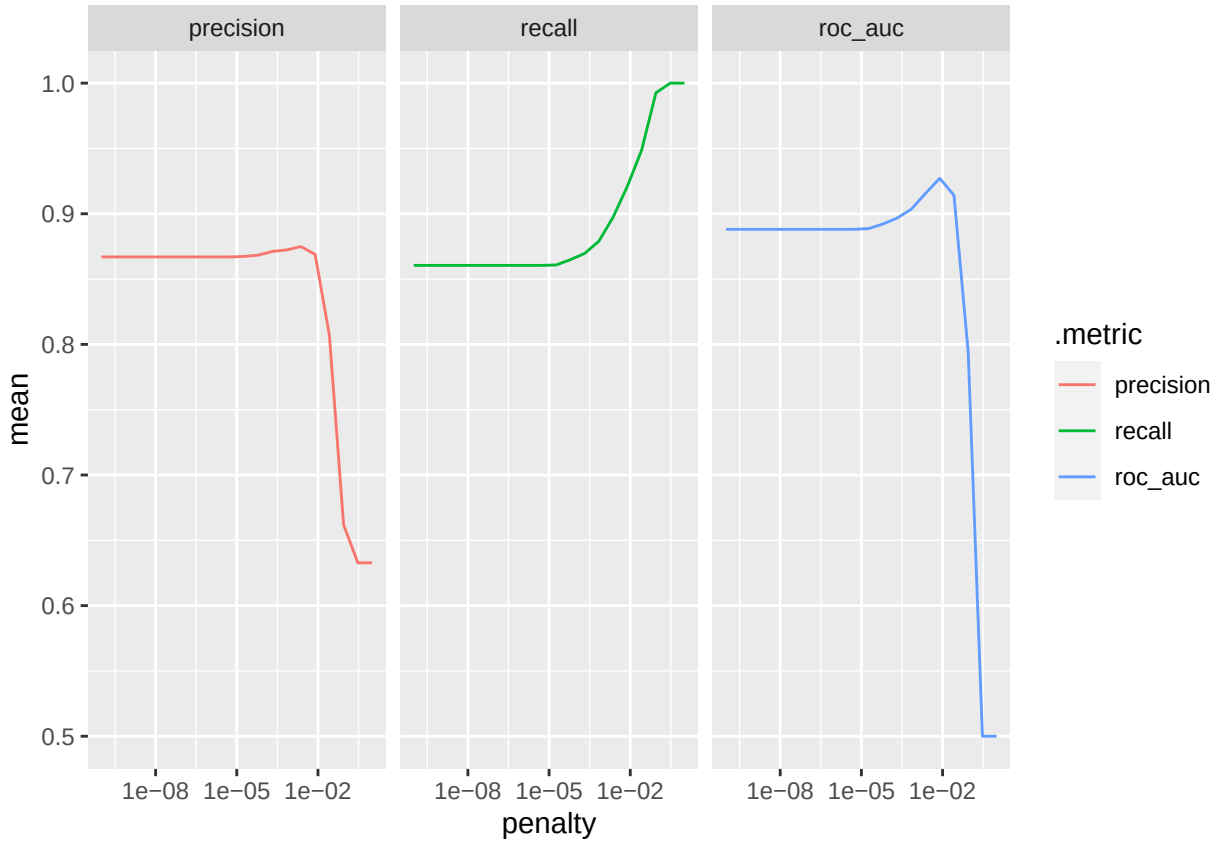
As one would expect model tuning is a long process so we will employ parallelism to speed up the process

```
doParallel::registerDoParallel()

model_grid <- tune_grid(model_wf, resamples = folds, grid=grid, metrics=metric_set(roc_auc, precision, r
```

With the tuning process completed we can visualize the metrics

```
model_grid %>% collect_metrics() %>% ggplot(aes(x=penalty, y=mean, color=.metric)) + geom_line() + facet
```



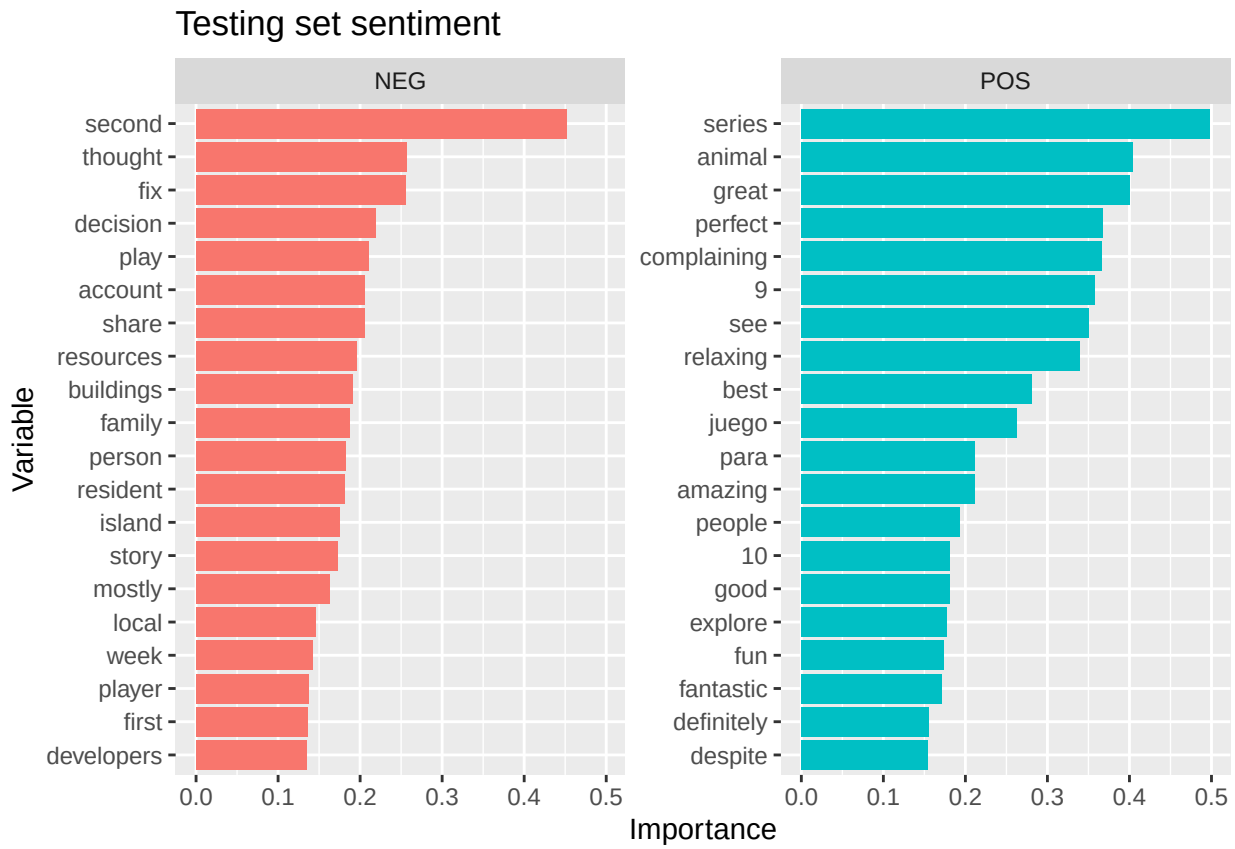
Like all model training there is a general give and take with precision and recall. The best ROC_AUC will also give approximately the best precision at the cost of some recall; however, that is the best we can do right now.

```
best_auc <- model_grid %>% select_best('roc_auc')
final_model <- finalize_workflow(model_wf, best_auc)
```

Now that we have tuned the model we can fit it to the training data and visualize works that correspond more to positive and negative reviews Variable importance

```
y_train_pred <- final_model %>% fit(df_train) %>% pull_workflow_fit() %>% vi(lambda = best_auc$penalty)
y_train_pred %>%
  mutate(Importance = abs(Importance)) %>%
  group_by(Sign) %>%
  top_n(20, wt = Importance) %>%
  ungroup() %>%
  mutate(Variable = fct_reorder(str_remove(Variable, 'tfidf_text_'), Importance)) %>%
  ggplot(aes(x=Variable, y=Importance, fill=Sign)) + geom_col(show.legend=F) + coord_flip() + facet_wra
```


```
y_test_pred <- final_model %>% fit(df_test) %>% pull_workflow_fit() %>% vi(lambda = best_auc$penalty)
y_test_pred %>%
  mutate(Importance = abs(Importance)) %>%
  group_by(Sign) %>%
  top_n(20, wt = Importance) %>%
  ungroup() %>%
  mutate(Variable = fct_reorder(str_remove(Variable, 'tfidf_text_'), Importance)) %>%
  ggplot(aes(x=Variable, y=Importance, fill=Sign)) + geom_col(show.legend=F) + coord_flip() + facet_wra
```

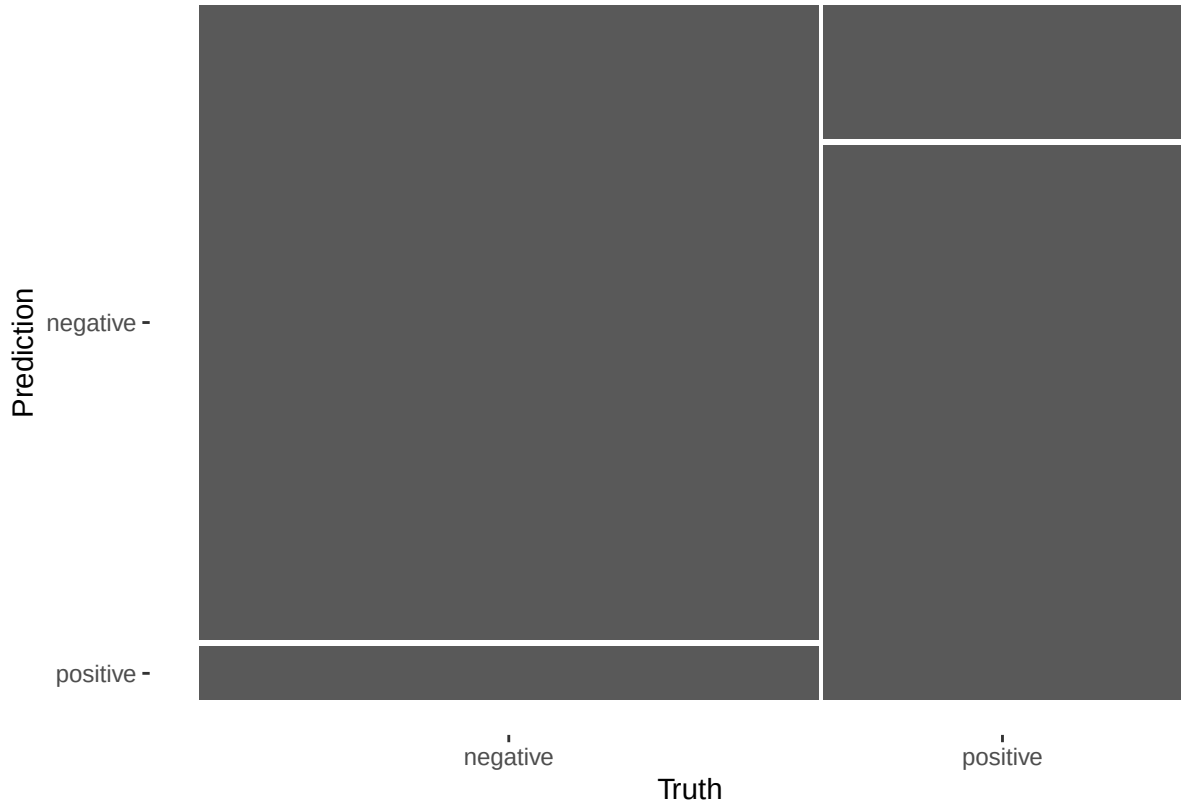


From both the training and testing set we can users gave negative reviews based on the *$60* price tag, the fact that there was *money* and *greed* involved and a lack of *sharing* due to there being only one *island* per game. The positive reviews are about the gaming being the *best*, *amazing*, *perfect* and fun. Some of the positive reviews claim that the negative reviews are bias towards the game and are review *bombing*.

Finally we can evaluate the model performance by evaluating on the testing data

```
review_final <- last_fit(final_model, df_split)

review_final %>% collect_predictions() %>% conf_mat(rating, .pred_class) %>% autoplot(type='mosaic')
```




```
review_final %>% collect_predictions() %>% conf_mat(rating, .pred_class)
```

```
##           Truth
## Prediction negative positive
##   negative      438       53
##   positive       37      221
```

From the mosaic plot we can see our model does well at predicting negative values, however it struggles a little more on predicting positing examples. The confusion matrix gives the granular details.

Conclusion The analysis done here is a good start but more work can be done. The text preprocessing would be a good start to remove other words that aren't very helpful. Also increasing the number of words used for training will give the model for data to work with; however it will take longer to tune and train. Over all the current model performed well from such a simplistic approach. The steps taken for this analysis can be use on any dataset in need for sentiment analysis.