CS 7280 - Project Report
Analysis and Prediction on Online News Popularity
Group: G6

1. Introduction

In recent years, popularity of social media has increased drastically. More and more people are joining social media platforms to share their views, likes, dislikes, interests and many other things with their family, friends and colleagues. Due to this, tons of news, stories, articles, images and videos are being shared on social media every day. Recent studies show that social media has become an epicenter of online news distribution and consumption. More than half of the social media site users have shared news stories, images or videos and nearly as many have discussed the news on social media [1]. As a result, there is an increased interest in identifying the news articles that will receive a significant amount of user attention.

In this project, we explore the use of linear regression models to explain the behavior of social media users and predict the number of shares of news articles based upon several statistical features extracted from the news articles. The Mashable news article dataset from UCI Machine Learning Repository is used in the analysis [2]. This dataset contains the record of news articles published by Mashable in a period of two years. In this analysis, we explore various relationship between all the statistical features and number of shares of a news article. Based upon that, various model building methods such Stepwise Regression, Regularization, Weighted Regression and Bootstrap are used and compared.

2. Data Analysis

The Online News Popularity Data Set contains 58 predictors, 2 non-predictors and a response variable, which is number of shares of an online news article. The 2 non-predictors are the unique URL and number of days between the article publication and the dataset acquisition. The predictors contain information extracted from news article, such as number of images, number of videos, title polarity, number of words in the content, rate of positive/negative words in the content etc. Among them, 3 predictors are categorical: type of article (data channel), day of week when the article was published and whether the article was published on weekend.

2.1 Data Cleaning

There are a total of 39797 instances in the dataset, which are uniquely identified with an URL. The dataset was split into training and test sets, with an 80/20 ratio. All the 58 predictors and the response variable were analyzed for the detection of missing values, outliers and collinearity between predictors. The dataset offers a description for each of the predictors [Appendix Table-1] and some properties can be inferred from that information; based on it, the values in the predictor were also checked for consistency.

Online news uses different media, like videos and images, but a lot of the predictors are related to text content. Some data points don't have any text and some of them even do not have any text, video or image. All those instances present 0 values for a high number of the predictors and they are removed from the scope of this analysis; they represent nearly 3% of the training data.

Some inconsistencies in the predictors are found during analysis:

· Rate of unique tokens higher than 1.

· Predictor (Minimum number of shares for a worst keyword) contains -1 value for more than 50% of the instances.

· Rate of nonstop words in the content become constant predictor after removing instances which don't have any text in it.

All the instances with those conditions are also removed from the training data set.

The five predictors which represent the closeness of a news article to five topics using Latent Dirichlet Allocation (LDA), have a high number of outliers on the positive side. Along with it, the predictors which represent the number of minimum, average and maximum shares of a worst, average and best keyword in a news article also have a wide spread. In both the cases, to reduce the influence of outliers, a log transformation is applied.

A closer look at the distribution of the response variable shows that it is far from normal. Number of shares for news articles goes from 1 to 843300, with a mean around 3300. This can be a problem for linear regression models, which have the assumption of a normally distributed error term. The Box-Cox transformation was applied to ensure that the distribution of the errors in the regression model follow a distribution close to normal. To evaluate the impact of the outliers in the model, we decided to perform two separate studies, with and without outliers. To remove the outliers, cook's distance is used. Cook's distance is selected because it not only detects the data points with large residuals and but also detects the data points with high leverages.

The distributions before and after applying the transformation can be observed in Figure 1.

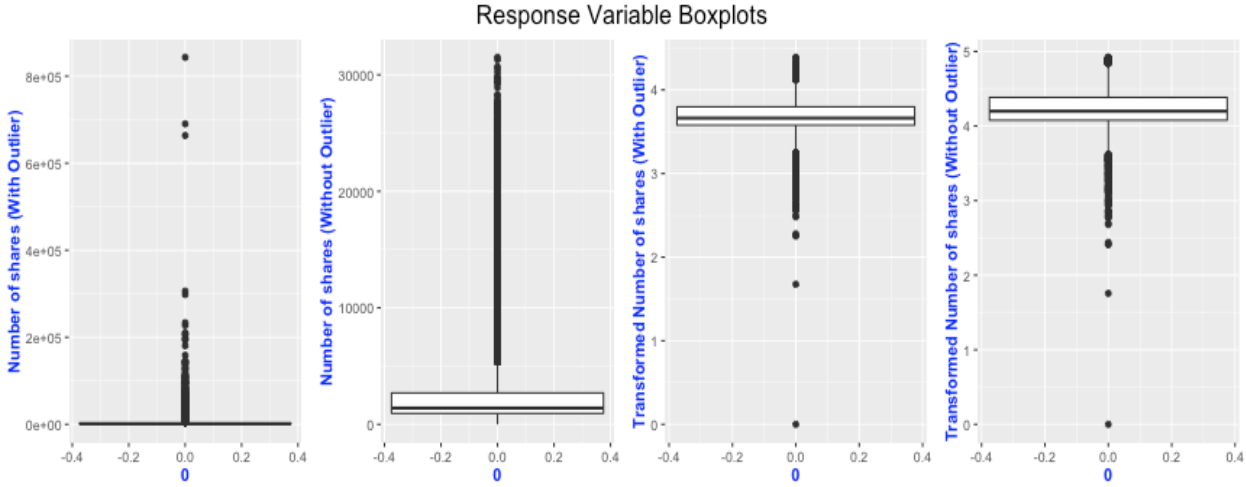

Figure 1: Response Variable Boxplots for the two train dataset with/without Box-Cox transformation

2.2 Multi-collinearity

Multi-collinearity is a serious problem as it can increase the variance in the coefficient estimation of a predictor and can cause the coefficient to change sign. To handle multi-collinearity, the pairwise Pearson correlation matrix of the continuous predictors is analyzed. [Appendix Figure 1]

The list of predictors which have high correlation, along with the approach to handle them is presented below:

1. *n_non_stop_unique_tokens* and *n_unique_tokens* have correlation of 0.887.

Those two predictors represent similar information, so n_non_stop_unique_tokens is removed from the analysis as both the predictors are semantically similar.

2. *self_reference_avg_sharess* has high correlation of 0.967 with *self_reference_min_shares* and 0.994 with *self_reference_max_shares*

Predictors *self_reference_min_shares* and *self_reference_max_shares* are removed.

3. *rate_positive_words* and *rate_negative_words* have correlation of -0.997.

Predictor variable *rate_negative_words* are removed from the analysis.

4. *kw_min_avg* and *kw_min_max* have correlation of 0.986.

Predictor variable *kw_min_max* was removed from the analysis.

5. *n_unique_tokens* and *n_tokens_content* have correlation of 0.751.
6. *title_subjectivity* and *abs_title_sentiment_polarity* have correlation of 0.71
7. *min_negative_polarity* and *avg_negative_polarity* have correlation of 0.719
8. *rate_positive_words* and *global_sentiment_polarity* have correlation of 0.779
9. *kw_max_min* and *kw_avg_min* have correlation of 0.901
10. *kw_max_avg* and *kw_avg_avg* have correlation of 0.899
11. *kw_avg_max* and *kw_max_max* have correlation of 0.913

In items from 5 to 11, interaction terms were added by taking the mean of the predictors.

3. Model Building

3.1 Stepwise Regression

Stepwise Regression is a handy method for regression problem as it automatically builds a model by successively adding or removing the predictors based upon the supplied criteria such as t-statistic, AIC or BIC. In this analysis, three stepwise regression methods are implemented; Forward, Backward and Both Directional. For each of these methods, 10-fold cross validation and AIC criteria are used for predictor selection. Using 10-fold cross validation, forward, backward and both directional stepwise regression models are trained on 10 different subsets of the dataset.

For predictor selection, the models generated in each fold are examined for all of the three stepwise regressions process. Initially, only those predictors which are selected in all the fold models for all the three stepwise regressions are kept. After that, a list of predictors that are selected in at least one of the fold model is created and the best subset selection method is applied using Mallow's cp criteria. For both the training datasets with and without outliers, initially 19 predictors are selected by all the fold models for all three stepwise regressions and 1 predictor is selected by best subset selection method from the remaining predictors. Note: Selected predictors are different for both the train datasets.

3.1.1 Interaction Terms

An interaction occurs when a predictor has a different effect on the response variable based upon the different values of other predictors. As mentioned in the first section, in this dataset there are three categorical variables - type of article (data channel), day of week when the article was published and whether articles was published on weekend? Interaction of all the continuous variables with these three categorical variables are analyzed and found the following significant interaction terms based upon it. [Appendix Figure 2] Using these interaction terms new models for both the train datasets are created considering best stepwise regression model as baseline.

1. *num_hrefs* and *data_channel_is_socmed*

   Number of shares are decreasing with the increase in the number of links for the news articles which are related to social media where as its reverse case for other categories
2. *num_imgs* and *data_channel_is_socmed*

   Number of shares are decreasing with the increase in the number of images for the news articles which are related to social media where as its reverse case for other categories
3. *num_imgs* and *is_weekend*

   Number of shares are drastically increasing with the increase in the number of images for the news article which are published on weekdays.
4. *global_subjectivity* and *data_channel_is_socmed*

   Number of shares are decreasing with the increase in text subjectivity for the news articles which are related to social media where as its reverse case for other categories.
5. *avg_positive_polarity* and *data_channel_is_socmed*

   Number of shares are decreasing with the increase in average polarity of positive words for the news article which are related to social media where as its reverse case for other categories.
6. *i_n_unique_tokens_content* and *data_channel_is_bus*

   Number of shares are drastically increasing with the increase in the rate of positive words in the content and text sentiment polarity for the news articles which are related to business compare to other categories of news article
7. *min_positive_polarity* and *data_channel_is_entertainment*

   Number of shares are increasing with the increase in minimum positive polarity for the news articles which are related to entertainment compare to other categories of news article
8. *n_tokens_title* and *weekday_is_Tuesday*

   Number of shares are increasing with the increase in number of token in the title for the news articles which are published on Tuesday compare to other days
9. *num_self_hrefs* and *is_weekend*

   Number of shares are increasing with the increase in number of links to other articles for the news article which are published on weekdays
10. *max_negative_polarity* and *is_weekend*

    Number of shares are increasing with the increase in maximum negative polarity in the news articles which are published on weekends.

3.2 Regularization

Regularization can be applied to linear regression to penalize more complex models and reduce the variance in parameter estimation. There are two basic ways of doing regularization. LASSO adds a penalty proportional to the sum of absolute values of the coefficients and produces a sparse representation of the predictors. RIDGE adds a penalty proportional to the sum of squares of coefficients and shrinks the values. Elastic net combines the two forms of regularization, using an extra parameter to measure the weight of each penalty term.

The glmnet package [3] is used for building regularized models. In order to find the best values for the parameters lambda (weight of the penalty term) and alpha (elastic-net mixing parameter), a grid search strategy was performed: a set of possible values are selected for the two parameters and all the possible combination between them are tested on a 10 fold cross-validation, using the RMSE measured on the validation sets. There is no significant difference between the models and we opted to use only one pure LASSO and one pure RIDGE models, with the best value of lambda found on cross-validation. The LASSO model selected with this method is keeping most of the predictors.

4. Model Comparison

A baseline model is added to the set of models for evaluation; it simply predicts the mean number of shares in the training data. Once all the models described above are built, a 10-fold cross validation, using the same seed, is performed and the RMSE are measured for the validation set in each fold. At the end, there are 10 measures for each model and a 95% confidence interval was built using Student's distribution with 9 degrees of freedom. The models are run on both the training datasets with and without outliers and the results are shown with error bars in Figure 2.

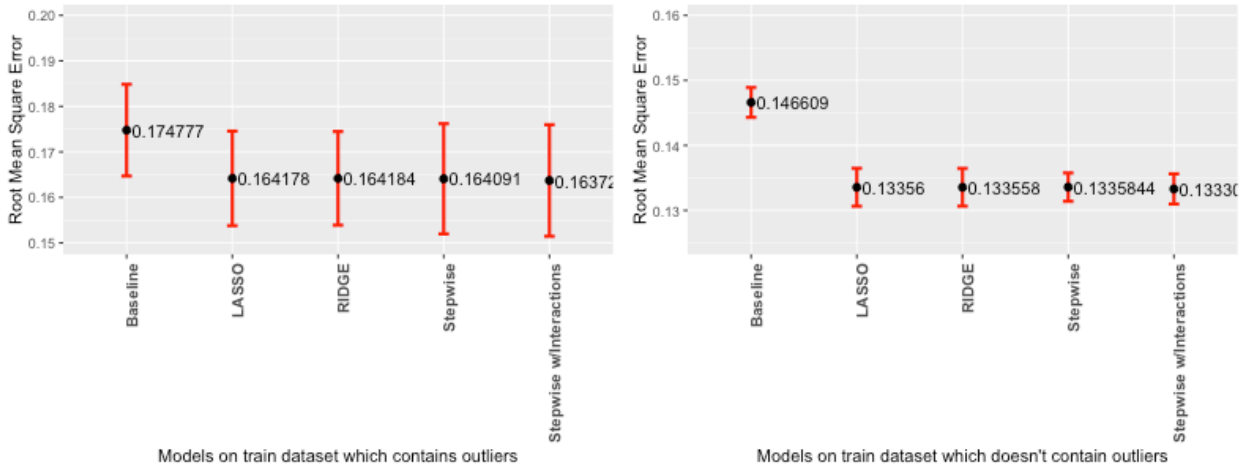

Figure 2: Various Model performance comparison on both the training data sets

Without removing outliers, the performance of the models is quite similar; there is no significant difference even when comparing to the baseline model. When outliers are removed, we see a better performance for the models, but there still isn't any significant difference between them (except for the baseline). Based on the results, we decided to use the model from stepwise feature selection, which is simpler and easier to understand as it has less predictors. Figure 3 shows the residual plots of the model trained on both the training datasets. It can be seen that the residuals don't follow the normal distribution.

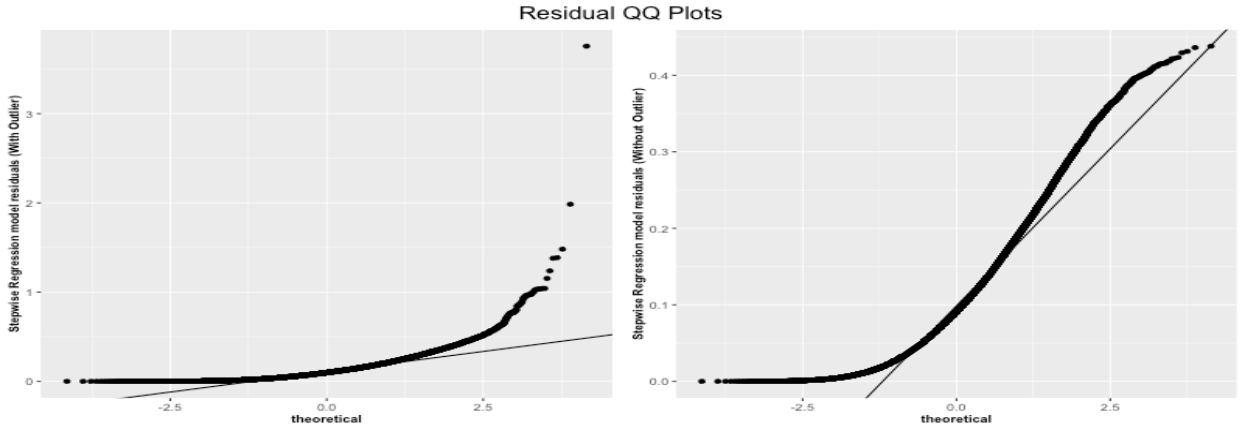

Figure 3: Residual QQ plots of stepwise regression model on both the training datasets

4.1 Weighted Regression

One of the assumptions for linear regression models is that the variance of the error term is constant over all values of the predictors. In our analysis, the variance has different value in different ranges of the predictor or response variables. Weighted regression can handle this situation as it weights the observations proportional to the reciprocal of the error variance of that observation to overcome the issue of non-constant variance. Figure 4 shows the change in the error term variance against the predicted response values using the full model.

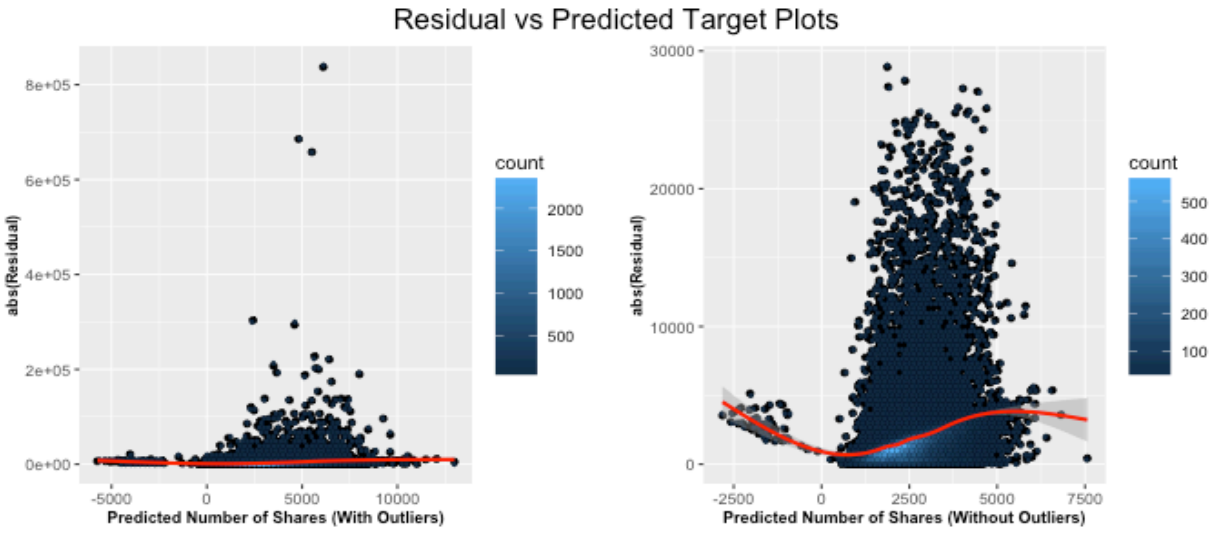

Figure 4: Residual vs Predicted response using full model on both the training datasets

To model the variance of the error terms, a linear model is fitted on all the predictors of the dataset and an untransformed response. After that absolute residuals and predicted response values of that model are used to train a new linear model to predict the standard deviation of each data point. Once the standard deviations are known, they are converted into weights by taking inverse of variance for each data point. Once the weights of each data points are known three different approaches are tried to train a model using weights.

In the first approach, instead of applying the Box-Cox transformation on the response variable, weights are used to transform the response variable and normal stepwise regression is performed using 10-fold cross validation. In the second approach, instead of applying weighted transformation on the full training dataset, it was applied on train dataset of each fold of cross validation and left the validation dataset response variable untransformed. In the third approach weights are directly used to train a weighted regression model. Performance of first two approach is equal to the baseline but third approach turns out to be very effective.

4.2 Bootstrap

The bootstrap method can give the precision of the estimated coefficient values for a fitted regression model in complex cases, such as when the error variance is not constant. From the residuals plot showed in Figure 4, we concluded that this is the case in our datasets, so bootstrap is applied on selected Stepwise Regression model. Multiple samples are taken from the observed data, with replacement, using the same number of observations. For each sample, a model is fitted and the estimated regression coefficients are calculated. Due to computational limitations, we used 300 bootstrap samples.

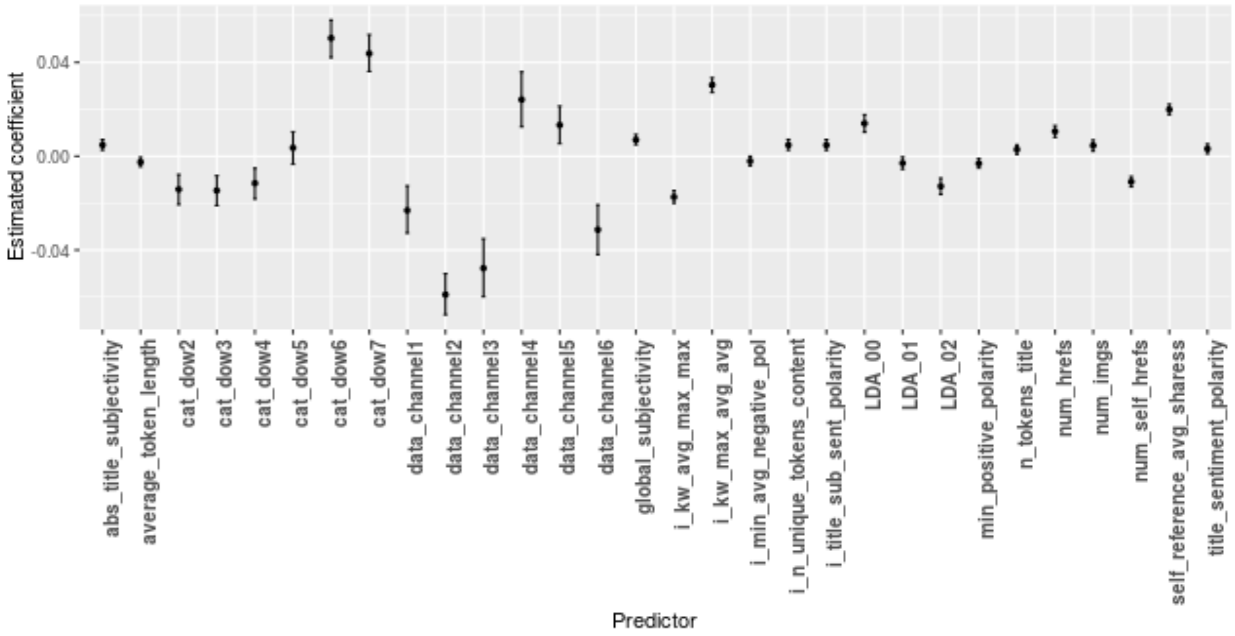

Figure 5: Predictor's coefficient estimation using Bootstrap

It can be seen in Figure 5 that only one predictor has a confidence interval that contains 0. It means that we don't have enough evidence to claim that it is significant to the model, but it can't be simply removed because it is part of the categorical variable "day of week". Overall, the categorical variables have a high influence in the model, which can be seen by the higher absolute values in the estimated coefficients.

Among all the three weighted regression models only the third approach performed better whereas the results of first and second approach were similar to the baseline model. To select the final model, Bootstrap and Weighted Regression models are compared by the means of residual plots. Figure 6 shows the residual plots of both the models and it can be seen that the residual plots are highly skewed in the Weighted Regression model compared to Bootstrap model so we selected Bootstrap model as final model.

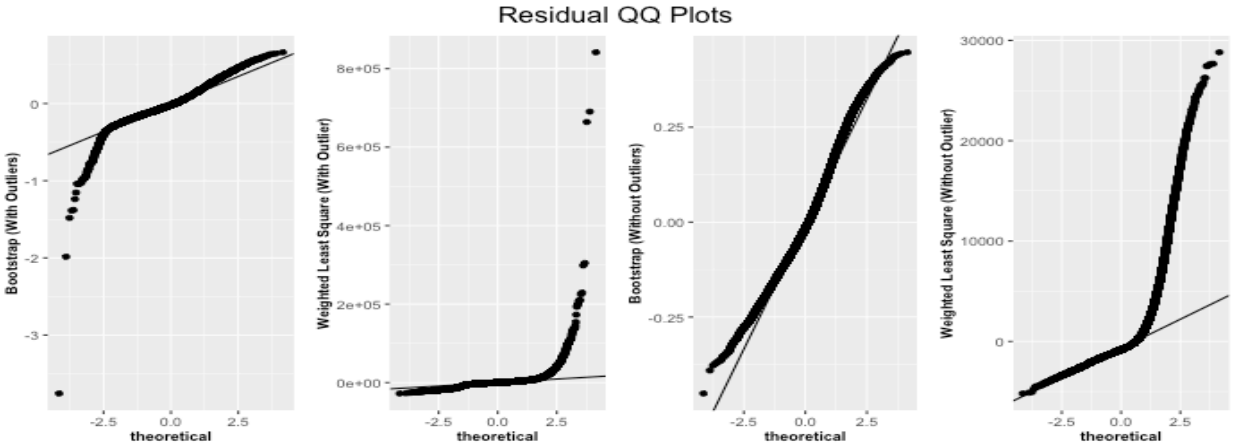

Figure 6: Residual QQ Plot for the Bootstrap and Weighted Regression model on both the training datasets

5. Evaluation

The final model used for the evaluation has the predictors selected in stepwise procedure and is bootstrapped; 300 models are trained on different bootstrap samples and the mean prediction of the models is used. The same preprocessing steps are performed on the test data and the predictions of the model are inversely transformed (from the Box-Cox transformation) to reflect the same magnitude of the original response variable. The RMSE on evaluation dataset is measured for the two separate bootstrapped models, one with outliers and one without outliers.

RMSE on evaluation set (with outliers in the training dataset): 14133.74
RMSE on evaluation set (without outliers in the training dataset): 14146.37

Removing the outliers from the training dataset doesn't improve the result on the evaluation set. So it can be concluded that the outliers are actually representative instances.

6. Discussion

Predicting the number of shares for online news using linear regression models was a challenging problem. We tried several methods for fitting a model, but there was a negligible improvement in the predictions and overall the models performed badly. The predictions were in a limited range and couldn't accommodate the data points with very high and low number of shares.

The residual plots show that a small number of data points dominate the errors. Other aspects of online news have to be investigated to see if new features can be extracted to help predicting better on those instances with very high and low number of shares. Apart from that, if we employ more complex models such as random forest or support vector machine then these wide range of number of shares could be better explained.

The assumption that the variance of the error term is constant did not hold for the simpler regression models. A weighted regression with a linear model for the variance was proposed, but there was no way to compare its results with the other models and the errors still do not show normal distribution. In the weighted regression, a more complex fit for the variance could be tried in future work.

Another approach for this dataset is to turn the problem into classification. The predictions could be whether the online news articles will be very popular or not; there could also be more than two classes represented by different intervals in the number of shares. A logistic regression model would be appropriate for that analysis.

7. References:

[1] Social Media Are Major Sources for News, Current Events (July 2015): http://kng.ht/1R5VqIE
[2] K. Fernandes, P. Vinagre and P. Cortez. A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News. Proceedings of the 17th EPIA 2015 - Portuguese Conference on Artificial Intelligence, September, Coimbra, Portugal
[3] glmnet: https://cran.r-project.org/web/packages/glmnet/vignettes/glmnet_beta.html

8. Appendix:

Table 1: Short description of the news data set predictors

0. url:	URL of the article
1. timedelta:	Days between the article publication and dataset acquisition
2. n_tokens_title:	Number of words in the title
3. n_tokens_content:	Number of words in the content
4. n_unique_tokens:	Rate of unique words in the content
5. n_non_stop_words:	Rate of non-stop words in the content
6. n_non_stop_unique_tokens:	Rate of unique non-stop words in the content
7. num_hrefs:	Number of links
8. num_self_hrefs:	Number of links to other articles published by Mashable
9. num_imgs:	Number of images
10. num_videos:	Number of videos
11. average_token_length:	Average length of the words in the content
12. num_keywords:	Number of keywords in the metadata
13. data_channel_is_lifestyle:	Is data channel 'Lifestyle'?
14. data_channel_is_entertainment:	Is data channel 'Entertainment'?
15. data_channel_is_bus:	Is data channel 'Business'?
16. data_channel_is_socmed:	Is data channel 'Social Media'?
17. data_channel_is_tech:	Is data channel 'Tech'?
18. data_channel_is_world:	Is data channel 'World'?
19. kw_min_min:	Worst keyword (min. shares)
20. kw_max_min:	Worst keyword (max. shares)
21. kw_avg_min:	Worst keyword (avg. shares)
22. kw_min_max:	Best keyword (min. shares)
23. kw_max_max:	Best keyword (max. shares)
24. kw_avg_max:	Best keyword (avg. shares)
25. kw_min_avg:	Avg. keyword (min. shares)
26. kw_max_avg:	Avg. keyword (max. shares)
27. kw_avg_avg:	Avg. keyword (avg. shares)
28. self_reference_min_shares:	Min. shares of referenced articles in Mashable
29. self_reference_max_shares:	Max. shares of referenced articles in Mashable
30. self_reference_avg_sharess:	Avg. shares of referenced articles in Mashable
31. weekday_is_monday:	Was the article published on a Monday?

32. weekday_is_tuesday:	Was the article published on a Tuesday?
33. weekday_is_wednesday:	Was the article published on a Wednesday?
34. weekday_is_thursday:	Was the article published on a Thursday?
35. weekday_is_friday:	Was the article published on a Friday?
36. weekday_is_saturday:	Was the article published on a Saturday?
37. weekday_is_sunday:	Was the article published on a Sunday?
38. is_weekend:	Was the article published on the weekend?
39. LDA_00:	Closeness to LDA topic 0
40. LDA_01:	Closeness to LDA topic 1
41. LDA_02:	Closeness to LDA topic 2
42. LDA_03:	Closeness to LDA topic 3
43. LDA_04:	Closeness to LDA topic 4
44. global_subjectivity:	Text subjectivity
45. global_sentiment_polarity:	Text sentiment polarity
46. global_rate_positive_words:	Rate of positive words in the content
47. global_rate_negative_words:	Rate of negative words in the content
48. rate_positive_words:	Rate of positive words among non-neutral tokens
49. rate_negative_words:	Rate of negative words among non-neutral tokens
50. avg_positive_polarity:	Avg. polarity of positive words
51. min_positive_polarity:	Min. polarity of positive words
52. max_positive_polarity:	Max. polarity of positive words
53. avg_negative_polarity:	Avg. polarity of negative words
54. min_negative_polarity:	Min. polarity of negative words
55. max_negative_polarity:	Max. polarity of negative words
56. title_subjectivity:	Title subjectivity
57. title_sentiment_polarity:	Title polarity
58. abs_title_subjectivity:	Absolute subjectivity level
59. abs_title_sentiment_polarity:	Absolute polarity level
60. shares:	Number of shares (target)

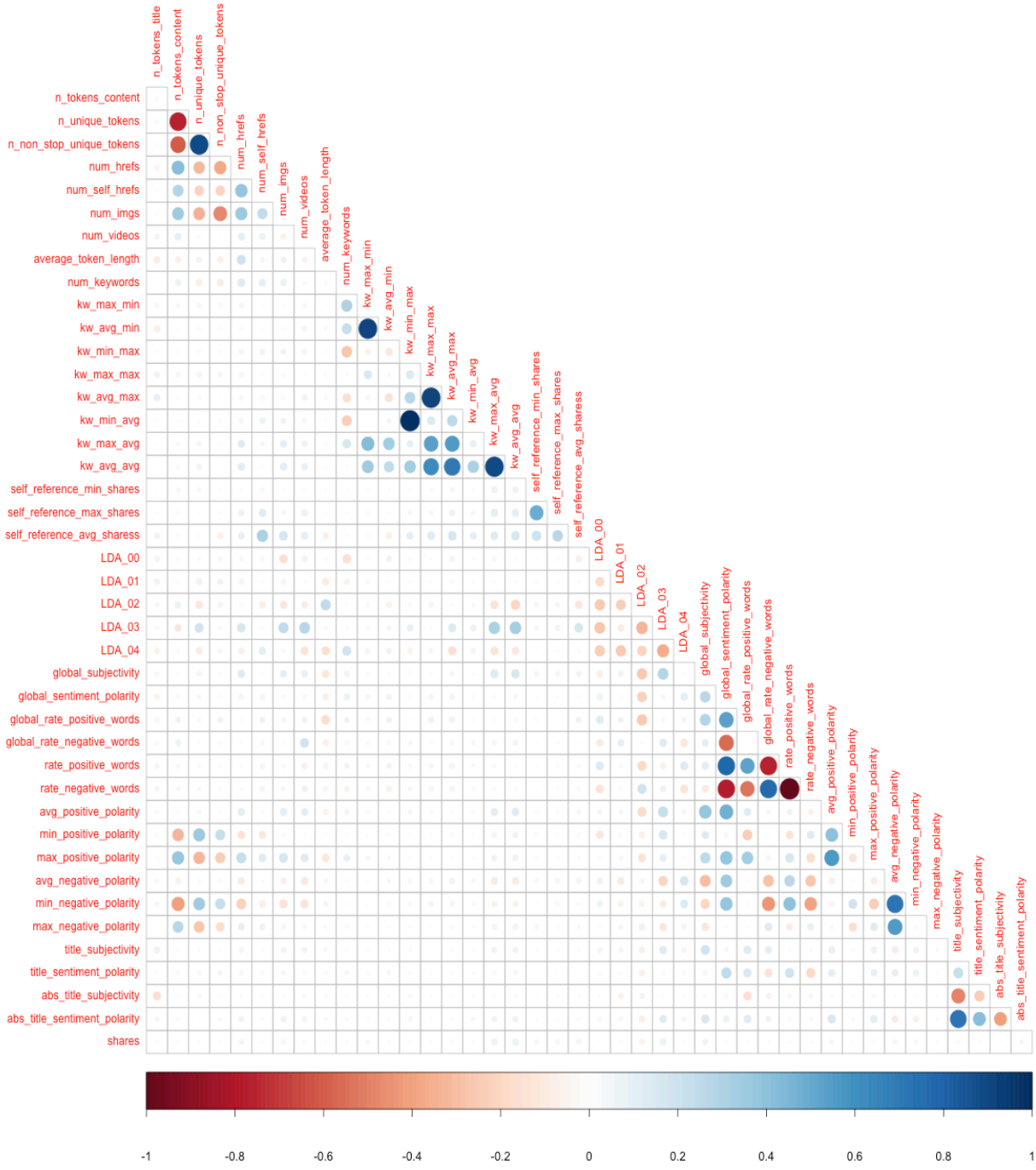

Figure 1: Graphical representation of Continuous predictors' correlation matrix

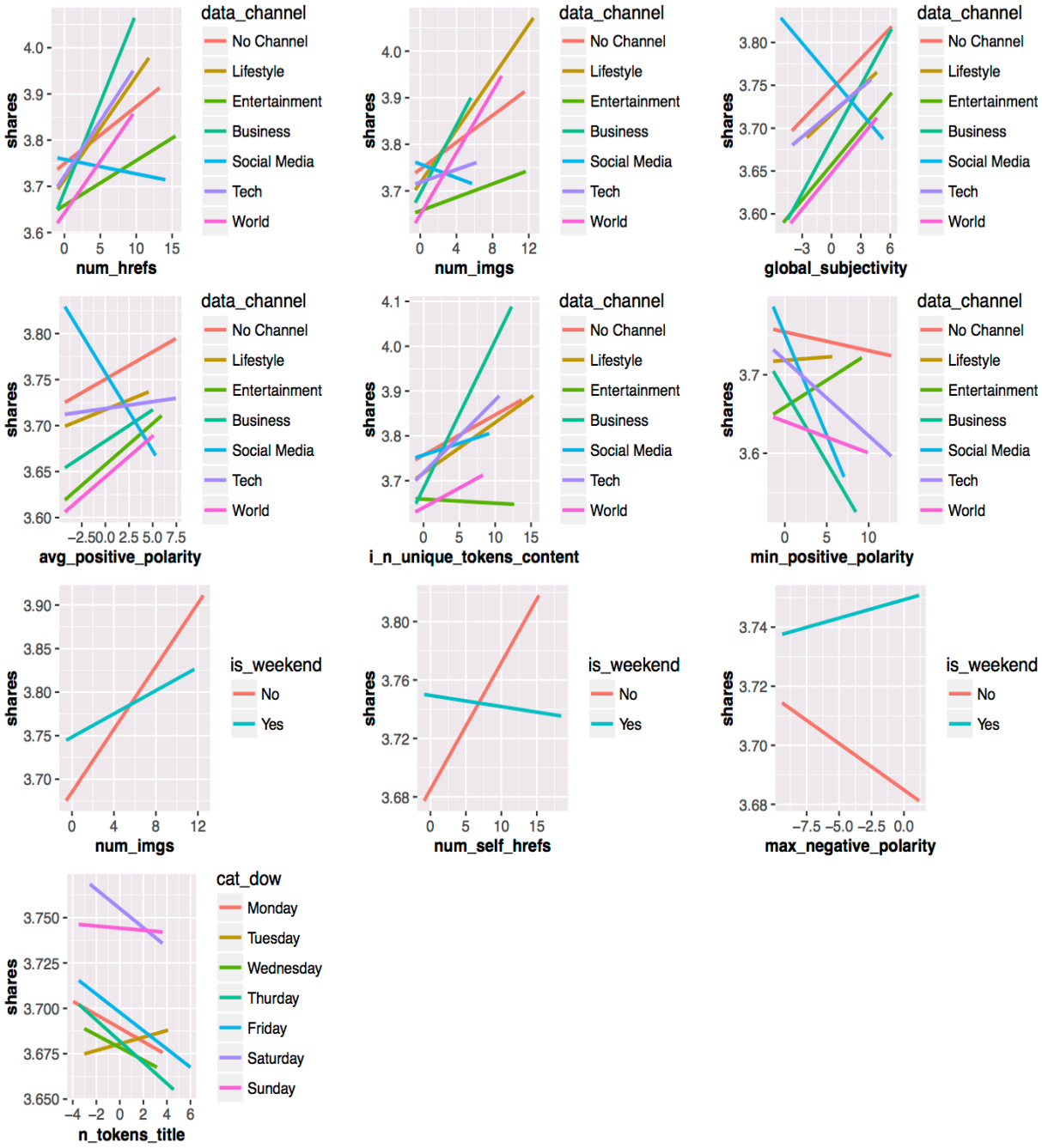

Figure 2: Graphical representation of Interaction between continuous variables and categorical variables considering response variable

R Code Sections

Section 1: Loading dataset
Section 2: Data Cleaning
Section 3: Stepwise Regression Model
Section 4: LASSO and RIDGE (Regularization)
Section 5: Weighted Regression
Section 6: Bootstrap
Section 7: R Plot Scripts

Section 1: Loading dataset

```
library(dplyr)
library(car)
library(caret)
library(glmnet)
library(ggplot2)
library(grid)
library(gridExtra)

setwd("/Users/Darshan/Documents/Online_News_Popularity")

# Train - Test data split
news <- read.csv("OnlineNewsPopularity.csv", header = TRUE)

set.seed(10)

# shuffle the data set
news <- news[sample(1:nrow(news)),]

# 20 % of data will be separated for testing
n_test_samples <- round(nrow(news) * 0.20)

news_test <- news[(1:n_test_samples),]
news_train <- news[((1+n_test_samples):nrow(news)),]

write.csv("Train.csv", row.names = FALSE, x = news_train)
write.csv("Test.csv", row.names = FALSE, x = news_test)
```

Section 2: Data Cleaning

```
# This function cleans the train dataset
data_cleaning <- function(news){

  # non-predictor
  news$timedelta <- NULL

  # Removing instances which don't have any text content in it.
  news <- filter(news, n_tokens_content != 0)
```

```
  news$n_non_stop_words <- NULL # Constanct predictor
  news$kw_min_min <- NULL # More than 50% instances contain -1 value
  news <- filter(news, n_unique_tokens <= 1) # Outlier value greater than 1

  news <- filter(news, kw_min_avg >= 0, kw_avg_min>=0)
  # Outlier value less than 1

  # Log transformation becuase there are high number of outliers
  news$LDA_00 <- log(news$LDA_00 + 1)
  news$LDA_01 <- log(news$LDA_01 + 1)
  news$LDA_02 <- log(news$LDA_02 + 1)
  news$LDA_03 <- log(news$LDA_03 + 1)
  news$LDA_04 <- log(news$LDA_04 + 1)

  news$self_reference_avg_sharess <- log(news$self_reference_avg_sharess + 1)

  news$kw_max_min <- log(news$kw_max_min + 1)
  news$kw_avg_min <- log(news$kw_avg_min + 1)

  news$kw_min_avg <- log(news$kw_min_avg + 1)
  news$kw_min_max <- log(news$kw_min_max + 1)
  news$kw_max_avg <- log(news$kw_max_avg + 1)
  news$kw_avg_avg <- log(news$kw_avg_avg + 1)

  news$kw_avg_max <- log(news$kw_avg_max + 1)
  news$kw_max_max <- log(news$kw_max_max + 1)

  return(news)

}

# This function handle the multi-collinearity in the train dataset.
correlation_cleaning <- function(news){

  news$rate_negative_words <- NULL
  # n_non_stop_unique_tokens and n_unique_tokens have correlation of 0.887
  #  n_non_stop_unique_tokens is removed from the analysis as both the predictors
  # are semantically similar
  news$n_non_stop_unique_tokens <- NULL

  # self_reference_min_shares and self_reference_max_shares has high corelation with
  # self_reference_avg_sharess
  news$self_reference_min_shares <- NULL
  news$self_reference_max_shares <- NULL

  news$i_n_unique_tokens_content <- news$n_unique_tokens + news$n_tokens_content
  # 0.751 colinearity between n_unique_tokens and n_tokens_content
  news$n_unique_tokens <- NULL
  news$n_tokens_content <- NULL
```

```
  news$i_title_sub_sent_polarity <- (news$title_subjectivity +
                                      news$abs_title_sentiment_polarity) / 2.0
  # 0.71 colinearity between title_subjectivity and abs_title_sentiment_polarity
  news$title_subjectivity <- NULL
  news$abs_title_sentiment_polarity <- NULL

  # 0.719 colinearity between min_negative_polarity and avg_negative_polarity
  news$i_min_avg_negative_pol <- (news$min_negative_polarity +
                                    news$avg_negative_polarity) / 2.0
  news$min_negative_polarity <- NULL
  news$avg_negative_polarity <- NULL

  # 0.779 colinearity between rate_positive_words and global_sentiment_polarity
  news$i_rate_pos_gsent_polarity <- (news$rate_positive_words *
                                       news$global_sentiment_polarity)
  news$rate_positive_words <- NULL
  news$global_sentiment_polarity <- NULL

  #kw_max_min and kw_avg_min have correlation of 0.901
  news$i_kw_max_avg_min <- (news$kw_max_min + news$kw_avg_min) / 2.0
  #kw_max_avg and kw_avg_avg have correlation of 0.899
  news$i_kw_max_avg_avg <- (news$kw_max_avg + news$kw_avg_avg) / 2.0

  # High collinearity after applying log transformation on kw_min_avg and kw_min_max
  # Log transformation has improved the r-squared value
  news$kw_min_max<- NULL

  # High collinearity after applying log transformation on kw_avg_max and kw_max_max
  # Log transformation has improved the r-squared value
  news$i_kw_avg_max_max <- (news$kw_avg_max + news$kw_max_max) / 2.0
  news$kw_avg_max <- NULL
  news$kw_max_max <- NULL

  news$kw_max_min <- NULL
  news$kw_avg_min <- NULL
  news$kw_max_avg <- NULL
  news$kw_avg_avg <- NULL

  # After trying different interactions between the predictors,
  # correlation did not decrease significantly, so
  # self_reference_min_shares and self_reference_max_shares
  # predictors are both removed.
  news$self_reference_min_shares <- NULL
  news$self_reference_max_shares <- NULL

  return(news)

}

# This function applies the Box-Cox transformation on responce variable
target_transformation <- function(news) {
```

```
  p <- powerTransform(news$shares)
  shares_transformed <- bcPower(news$shares, p$lambda)
  news$shares <- shares_transformed

  return(list("news"=news, "lambda"=p$lambda))
}

# This function returns the actual value of the responce variable
# from the Box-Cox transformation
target_inverse <- function(shares, lambda) {
  if (lambda == 0) {
    shares <- exp(shares)
  }
  else {
    shares <- (lambda*shares + 1)^(1/lambda)
  }

  return(shares)
}

# This funciton normalize continuous variables of the train dataset
normalization <- function(news_train){

  # All Column names
  column_names <- names(news_train)

  # Column names which needs to be ignored due to categorical and target feature
  ignored_column_names <- c("url", "timedelta", "data_channel_is_lifestyle",
                            "data_channel_is_entertainment", "data_channel_is_bus",
                            "data_channel_is_world", "data_channel_is_socmed",
                            "data_channel_is_tech", "weekday_is_monday",
                            "weekday_is_tuesday",
                            "weekday_is_wednesday", "weekday_is_thursday",
                            "weekday_is_friday",
                            "weekday_is_saturday", "weekday_is_sunday", "is_weekend",
                            "shares")

  needed_columns <- setdiff(column_names,ignored_column_names)

  # Normalized Train Data
  #news_train_norm <- news_train %>% mutate_each_(funs(scale),vars=needed_columns)

  # Saving standard deviation of the columns which are normalized
  sd_values <- Map(sd, news_train[,needed_columns])

  # Saving mean of the columns which are normalized
  mean_values <- Map(mean, news_train[,needed_columns])

  news_train[,needed_columns] <- (news_train[,needed_columns] - mean_values) / sd_values
```

```
  return(list("sd_values"=sd_values, "mean_values"=mean_values, "news_train"=news_train))

}

# This funciton normalize continuous variables of the test datset
apply_normalization <- function(news, means, sds) {
  # All Column names
  column_names <- names(news_train)

  # Column names which needs to be ignored due to categorical and target feature

  ignored_column_names <- c("url", "timedelta", "data_channel_is_lifestyle",
                            "data_channel_is_entertainment", "data_channel_is_bus",
                            "data_channel_is_world", "data_channel_is_socmed",
                            "data_channel_is_tech", "weekday_is_monday",
                            "weekday_is_tuesday",
                            "weekday_is_wednesday", "weekday_is_thursday",
                            "weekday_is_friday",
                            "weekday_is_saturday", "weekday_is_sunday", "is_weekend",
                            "shares")

  needed_columns <- setdiff(column_names,ignored_column_names)

  news[,needed_columns] <- (news[,needed_columns] - means) / sds

  return(news)
}

# This function creates the factor/single categorical variable by combining
# multiple/one hot encoded variables
cat_encoding <- function(news){

  dow_cols = c("weekday_is_monday", "weekday_is_tuesday", "weekday_is_wednesday",
               "weekday_is_thursday", "weekday_is_friday", "weekday_is_saturday",
               "weekday_is_sunday")

  news$cat_dow <- 0

  for (dow in dow_cols) {
    dow_idx = which(news[,dow] == 1)
    #print(dow_idx)
    news[dow_idx,"cat_dow"] <- which(dow_cols==dow)
  }

  news$cat_dow <- as.factor(news$cat_dow)

  data_channel_cols = c("data_channel_is_lifestyle", "data_channel_is_entertainment",
                        "data_channel_is_bus", "data_channel_is_socmed",
                        "data_channel_is_tech",
                        "data_channel_is_world")
```

```
  news$data_channel <- 0

  for (channel in data_channel_cols) {
    channel_idx <- which(news[,channel] == 1)
    news[channel_idx,"data_channel"] <- which(data_channel_cols==channel)
  }

  news$data_channel <- as.factor(news$data_channel)

  news$is_weekend <- as.factor(news$is_weekend)

  return(news)

}

OUTLIERS_HIGH_CUTOFF = 0.1
OUTLIERS_LOW_CUTOFF = 0.05
outliers_removal <- function(news) {
  # sort by shares
  sorted_news <- news[order(news$shares),]

  num_rows <- nrow(news)
  # remove lower tail
  cut_low_point <- as.integer(OUTLIERS_LOW_CUTOFF*num_rows)
  cut_high_point <- as.integer((1-OUTLIERS_HIGH_CUTOFF)*num_rows)
  sorted_news <- sorted_news[cut_low_point:cut_high_point, ]
  news <- sorted_news[sample(nrow(sorted_news)),]
  return(sorted_news)
}

# This function removes the outlier from the dataset based upon the
# cook's distance
cook_outliers_removal <- function(news){

  cutoff <- 4/nrow(news)
  model <- lm(shares ~ ., data=news)
  infl <- lm.influence(model, do.coef = FALSE)

  cooks.distance <- cooks.distance(model, infl = infl,
                                   res = weighted.residuals(model),
                                   sd = sqrt(deviance(model)/df.residual(model)),
                                   hat = infl$hat)

  index <- cooks.distance <= cutoff
  news <- news[index,]

  return(news)

}

# This function loads the train data set and applies the
```

```
# data cleaning operation to it.
load_processed_train_data <- function(outliers.removed=FALSE,
                                      one.hot.encoding.remove=TRUE){

  news <- read.csv("Train.csv", header = TRUE)

  news <- data_cleaning(news)
  news <- correlation_cleaning(news)

  obj <- normalization(news)
  news <- obj$news

  news <- cat_encoding(news)

  url <- news$url
  news$url <- NULL

  if(one.hot.encoding){

    categorical_var <- c("data_channel_is_lifestyle",
                         "data_channel_is_entertainment", "data_channel_is_bus",
                         "data_channel_is_world", "data_channel_is_socmed",
                         "data_channel_is_tech", "weekday_is_monday", "weekday_is_tuesday",
                         "weekday_is_wednesday", "weekday_is_thursday", "weekday_is_friday",
                         "weekday_is_saturday", "weekday_is_sunday")

    news_with_cat <- subset(news, select = categorical_var)
    news <- subset(news, select = setdiff(names(news),categorical_var))
  }

  if(outliers.removed){
    news <- cook_outliers_removal(news)
  }

  return(news)
}
```

Section 3: Stepwise Regression Model

```
set.seed(464)

news <- load_processed_train_data()

K <- 10
# 10 - fold cross validation
folds <- createFolds(news$shares, k = K, list=TRUE, returnTrain=TRUE)

models <- list()
```

```
rmses <- c()
R2s <- c()

for (i in 1:K) {

  news_train <- news[folds[[i]],]
  news_val <- news[-folds[[i]],]

  null=lm(shares~1, data=news_train)
  full=lm(shares~., data=news_train)

  model <- step(null, scope=list(lower=null, upper=full), direction="both", trace=0)
  #model <- step(full, direction="backward", trace=0)

  pred <- predict(model, news_val)

  #pred <- target_inverse(pred, lamda)
  #shares_val <- target_inverse(news_val$shares, lamda)
  #mse <- sum((pred - shares_val)**2) / nrow(news_val)

  mse <- sum((pred - news_val$shares)**2) / nrow(news_val)
  rmses <- append(rmses, sqrt(mse))

  R2s <- append(R2s, summary(model)$adj.r.squared)

  models[[i]] <- model

}

# Displaying which variables are selected in the each fold
unique_coef <- c()

for(i in 1:length(models)){
  model_coef <- names(models[[i]]$coefficients)
  unique_coef <- unique(c(model_coef, unique_coef))
}

model_variables <- data.frame(matrix(NA,nrow=length(unique_coef),ncol=length(models)+1))
model_variables$X1 <- unique_coef

for(i in 1:length(models)){

  model_coef <- names(models[[i]]$coefficients)
  tf_coef <- unique_coef %in% model_coef
  var <- paste("X", toString(i+1), sep = "")
  model_variables[var] <- tf_coef

}
```

Section 4: LASSO and RIDGE (Regularization)

```
set.seed(464)

# run grid search with cross validation to select best values for lambda and alpha in elastic net
select_model <- function(news, t_lambda) {
  K = 10
  # alpha = 0 -> Ridge; alpha = 1 -> Lasso
  alphas = c(0,1)
  lambdas = c(1e-05, 1e-04, 1e-03, 1e-02, 0.1, 1.,10.)

  folds <- createFolds(news$shares, k = K, list=TRUE, returnTrain=TRUE)
  # for each combination of parameters
  for (alpha in alphas) {
    for (lambda in lambdas) {
      rmses <- c()
      R2s <- c()

      # for each fold
      for (i in 1:K) {
        news_train <- news[folds[[i]],]
        news_val <- news[-folds[[i]],]

        X_train <- data.matrix(subset(news_train,select=-shares))
        y_train <- data.matrix(news_train$shares)
        X_val <- data.matrix(subset(news_val,select=-shares))
        y_val <- data.matrix(news_val$shares)

        model <- glmnet(X_train, y_train, family="gaussian", alpha=alpha, standardize=TRUE,
                        lambda=lambda, nlambda=1)

        pred_train <- predict(model, newx=X_train, s=lambda)
        shares_train <- y_train

        # calculate R^2 in the fitted data
        ssto <- sum((shares_train - mean(shares_train))**2)
        sse <- sum((pred_train - shares_train)**2)
        R2 <- 1 - (sse/ssto)

        R2s <- append(R2s, R2)

        pred <- predict(model, newx=X_val, s=lambda)
        shares_val <- y_val

        sse <- sum((pred - shares_val)**2)
        rmse <- sqrt(sse / nrow(news_val))

        rmses <- append(rmses,rmse)
      }
```

```
    mrmse= mean(rmses)
    srmse= sd(rmses)
    mR2 = mean(R2s)
    cat(sprintf("alpha = %f, lambda = %f, avg rmse = %f, sd rmse = %f, avg R-2 = %f\n",
            alpha, lambda, mrmse,srmse,mR2))
  }
 }

}

news <- load_processed_train_data()
select_model(news, t_lambda)
```

Section 5: Weighted Regression

```
ncvTest(lm(shares ~ .,data=news))
m.unweighted <- lm(shares ~ ., data=news)

# Learing weights of each data point
w <- predict(lm(abs(m.unweighted$res) ~ predict(m.unweighted, data=news)), data=news)
# First Approach, updating responce variable based upon weights
#w <- (w - min(w))/(max(w) - min(w))
#news$shares <- news$shares * w

K <- 10
# Third Approach; Learning from the weights
model <- lm(formula = shares ~ ., data = news, weights = 1/(w^2))

folds <- createFolds(news$shares, k = K, list=TRUE, returnTrain=TRUE)

models <- list()
rmses <- c()
R2s <- c()
for (i in 1:K) {

  news_train <- news[folds[[i]],]
  news_val <- news[-folds[[i]],]
  #w <- w[folds[[i]]]

  m.unweighted <- lm(shares ~ ., data=news_train)
  w <- predict(lm(abs(m.unweighted$res) ~ predict(m.unweighted, data=news_train)), data=ne
ws_train)
  # Second Approach, updating responce variable based upon weights and fold
  #w <- (w - min(w))/(max(w) - min(w))
  #news_train$shares <- news_train$shares * w

  null=lm(shares~1, data=news_train)
  full=lm(shares~., data=news_train)
```

```
  model <- lm(formula = shares ~ ., data = news_train, weights = 1/(w^2))
  #model <- step(null, scope=list(lower=null, upper=full), direction="forward", trace=0)

  pred <- predict(model, news_val)

  #pred <- target_inverse(pred, lamda)
  #shares_val <- target_inverse(news_val$shares, lamda)
  #mse <- sum((pred - shares_val)**2) / nrow(news_val)

  mse <- sum((pred - news_val$shares)**2) / nrow(news_val)
  rmses <- append(rmses, sqrt(mse))

  R2s <- append(R2s, summary(model)$adj.r.squared)

  models[[i]] <- model

}
```

Section 6: Bootstrap

```
set.seed(464)
news <- load_processed_train_data()
B = 300
bootstrap <- function(formula, data) {
  n_rows <- nrow(data)

  models <- vector(mode="list", length=B)
  for (i in 1:B) {
    # sample the same number of points with replacement
    boot_idx <- sample(n_rows, n_rows, replace = TRUE)
    boot_data <- data[boot_idx, ]

    m <- lm(formula, data=boot_data)

    models[[i]] <- m
  }

  return(models)
}


# stepwise selection (with outliers)
predictors <- c("data_channel", "cat_dow", "i_kw_max_avg_avg",
                "self_reference_avg_sharess", "i_kw_avg_max_max",
                "num_hrefs", "global_subjectivity", "LDA_00",
                "LDA_01", "LDA_02", "num_self_hrefs",
                "i_n_unique_tokens_content", "i_title_sub_sent_polarity",
```

```
            "abs_title_subjectivity", "n_tokens_title", "min_positive_polarity",
            "num_imgs", "average_token_length", "title_sentiment_polarity",
            "i_min_avg_negative_pol")

# # # stepwise selection (without outliers)
# predictors <- c("num_hrefs", "num_self_hrefs", "num_imgs",
#   "self_reference_avg_sharess", "LDA_00", "LDA_02", "global_subjectivity",
#   "global_rate_positive_words", "global_rate_negative_words", "min_positive_polarity",
#   "max_negative_polarity", "title_sentiment_polarity", "abs_title_subjectivity",
#   "i_n_unique_tokens_content", "i_rate_pos_gsent_polarity", "i_kw_max_avg_avg",
#   "i_kw_avg_max_max", "cat_dow", "data_channel", "i_title_sub_sent_polarity")


formula <- as.formula(paste("shares~", paste(predictors,collapse="+")))

# number of coefficients in the model
N_COEF <- 31

# get the coefficients values from each model
coef <- matrix(nrow = B, ncol=N_COEF)
models <- bootstrap(formula, news)
for (i in 1:length(models)) {
  for (j in 2:N_COEF) {
    coef[i,j] <- coef(models[[i]])[[j]]
  }
}

# train a model on the full dataset
full_model <- lm(formula, data=news)
full_coef <- vector(mode="list", length=N_COEF)
predictor_names <- names(full_model$coefficients)[2:N_COEF]

# get the coefficients of the full model
for (i in 2:N_COEF) {
  full_coef[[i]] <- coef(full_model)[[i]]
}

# calculate coefficients confidence intervals
coef_max <- vector(mode="list", length=N_COEF)
coef_min <- vector(mode="list", length=N_COEF)
for (i in 2:N_COEF) {
  b_star_upper <- qnorm(0.975, mean=mean(coef[,i]), sd=sd(coef[,i]))
  b_star_lower <- qnorm(0.025, mean=mean(coef[,i]), sd=sd(coef[,i]))

  d1 <- full_coef[[i]] - b_star_upper
  d2 <- b_star_lower - full_coef[[i]]

  coef_max[[i]] <-  full_coef[[i]] - d2
  coef_min[[i]] <- full_coef[[i]] + d1

  cat(sprintf("predictor: %s, lower_value = %f, upper_value = %f\n",
```

```
            predictor_names[i], coef_min[[i]], coef_max[[i]]))
}

# plot the coefficient and their confidence interval
results = data.frame(name=predictor_names, coef=unlist(full_coef), max=unlist(coef_max), mi
n=unlist(coef_min))

ggplot(results, aes(x = name, y = coef)) +
  geom_point(size = 1) +
  labs(x = "Predictor", y = "Estimated coefficient") +
  geom_errorbar(aes(ymax = max, ymin = min),width=0.1) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size=10, face="bold"))

# prediction for all the models
pred <- matrix(nrow = nrow(news), ncol=B)
for (i in 1:length(models)) {
  m <- models[[i]]
  pred[,i] <- predict(m, subset(news,select=-shares))
}

sse <- sum((rowMeans(pred) - news$shares)**2)
rmse <- sqrt(sse / nrow(news))
```

Section 7: R Plot Scripts

```
news <- load_processed_train_data()

news_wo_outlier <- cook_outliers_removal(news)

model <- lm(shares ~ ., data=news)

news$res <- abs(model$residuals)
news$pre <- predict(model, data=news)

model <- lm(shares ~ ., data=news_wo_outlier)

news_wo_outlier$res <- abs(model$residuals)
news_wo_outlier$pre <- predict(model, data=news_wo_outlier)

p1 <- ggplot(aes(x=pre,y=res) ,data=news)  +  geom_point() + xlab("Predicted Number of Sha
res (With Outliers)") + ylab("abs(Residual)") + stat_binhex(bins = 75)  + geom_smooth(color =
"red") + theme(axis.title=element_text(size=9,face="bold"))

p2 <- ggplot(aes(x=pre,y=res) ,data=news_wo_outlier)  +  geom_point() + xlab("Predicted Nu
mber of Shares (Without Outliers)") + ylab("abs(Residual)") + stat_binhex(bins = 75)  + geom_
smooth(color = "red") + theme(axis.title=element_text(size=9,face="bold"))

grid.arrange(p1, p2, ncol = 2, top = "Residual vs Predicted value of Shares")
```

```r
news <- load_processed_train_data()

model <- lm(shares ~ data_channel + cat_dow + i_kw_max_avg_avg +
            self_reference_avg_sharess + i_kw_avg_max_max +
            num_hrefs + global_subjectivity + LDA_00 +  LDA_01 +
            LDA_02 + num_self_hrefs + i_n_unique_tokens_content +
             i_title_sub_sent_polarity + abs_title_subjectivity +
            n_tokens_title +  min_positive_polarity +
            num_imgs + average_token_length + title_sentiment_polarity +
            i_min_avg_negative_pol, data=news)

news$res <- abs(model$residuals)
news$pre <- predict(model, data=news)
y <- quantile(news$res, c(0.25, 0.75))
x <- qnorm(c(0.25, 0.75))
slope <- diff(y)/diff(x)
int <- y[1L] - slope * x[1L]

p1 <- ggplot(news, aes(sample=res)) + stat_qq() + geom_abline(slope = slope, intercept = int)
+ ylab("Stepwise Regression model residuals (With Outlier)") + theme(axis.title=element_text(
size=9,face="bold"))

news_wo_outlier <- cook_outliers_removal(news)

model <- lm(shares ~ num_hrefs +  num_self_hrefs + num_imgs +
             self_reference_avg_sharess + LDA_00 + LDA_02 +
            global_subjectivity + global_rate_positive_words + global_rate_negative_words +
            min_positive_polarity + max_negative_polarity + title_sentiment_polarity +
            abs_title_subjectivity + i_n_unique_tokens_content +
            i_rate_pos_gsent_polarity + i_kw_max_avg_avg + i_kw_avg_max_max +
            cat_dow + data_channel +
            i_title_sub_sent_polarity, data=news_wo_outlier)

news_wo_outlier$res <- abs(model$residuals)
news_wo_outlier$pre <- predict(model, data=news)

y <- quantile(news_wo_outlier$res, c(0.25, 0.75))
x <- qnorm(c(0.25, 0.75))
slope <- diff(y)/diff(x)
int <- y[1L] - slope * x[1L]

p2 <- ggplot(news_wo_outlier, aes(sample=res)) + stat_qq() +geom_abline(slope = slope, int
ercept = int) + ylab("Stepwise Regression model residuals (Without Outlier)") + theme(axis.title
=element_text(size=9,face="bold"))

grid.arrange(p1, p2, ncol = 2, top = "Residual QQ Plots")
```

Statement of Contributions

Darshan Patel mainly worked on building the Stepwise and Weighted least square regression models. He programmed various variable selection methods in R and compared their results. He also wrote R scripts to visualize Multi-collinearity, Interaction Terms and Residual plots. Along with that he assisted Gabriel in building LASSO, Adaptive LASSO, RIDGE and Bootstrap and Data analyses/cleaning process.

Gabriel Bakiewicz mainly worked on building the Regularization and Bootstrap models. He programmed LASSO, Adaptive LASSO, RIDGE and Bootstrap in R and build scripts to visualize the confidence interval of the predictor estimation and compare the different models. Along with that he assisted Darshan in building Stepwise and Weighted least square regression models and Data analyses/cleaning process.