**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

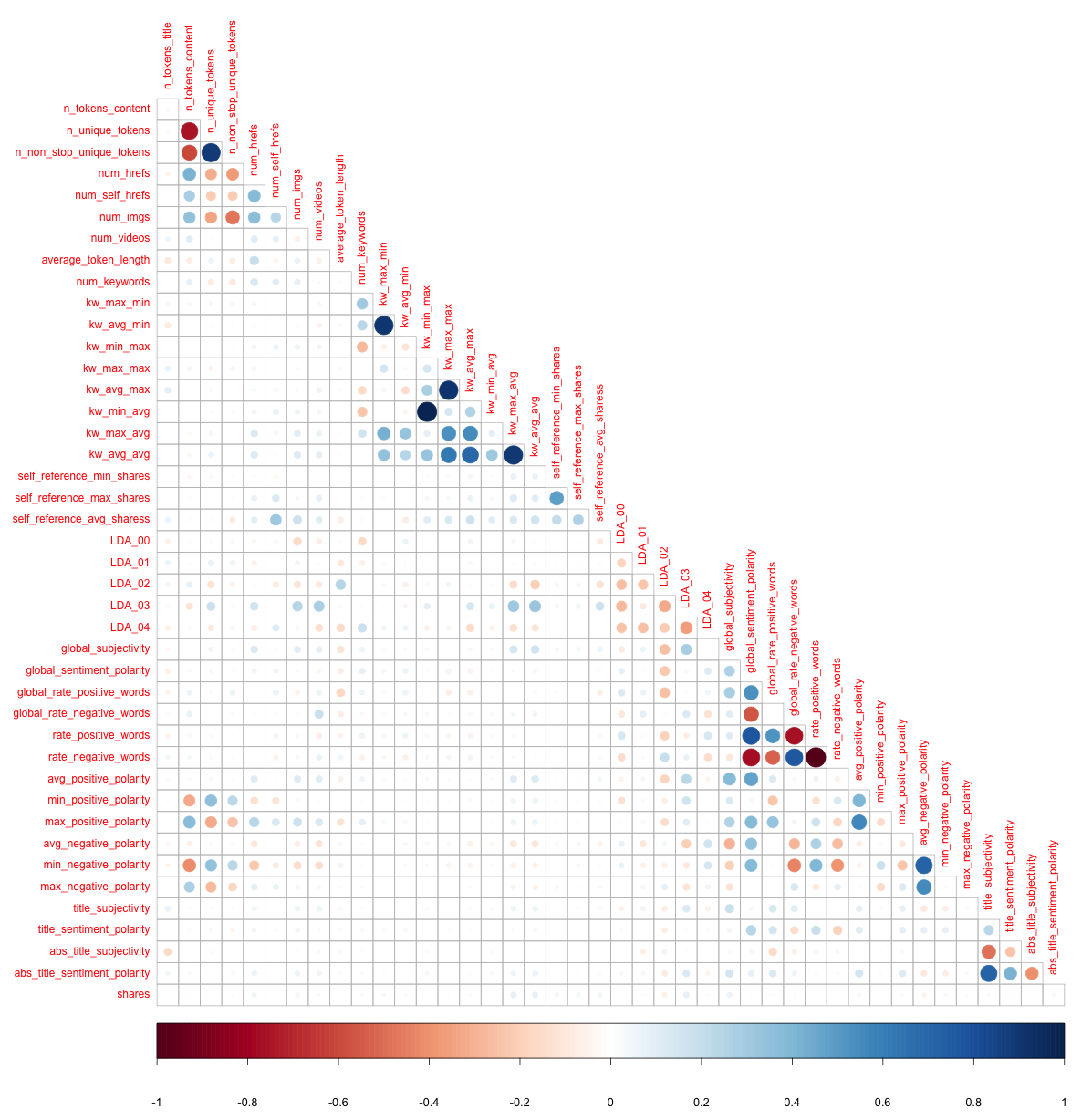
56. title\_subjectivity: Title subjectivity

57. title\_sentiment\_polarity: Title polarity

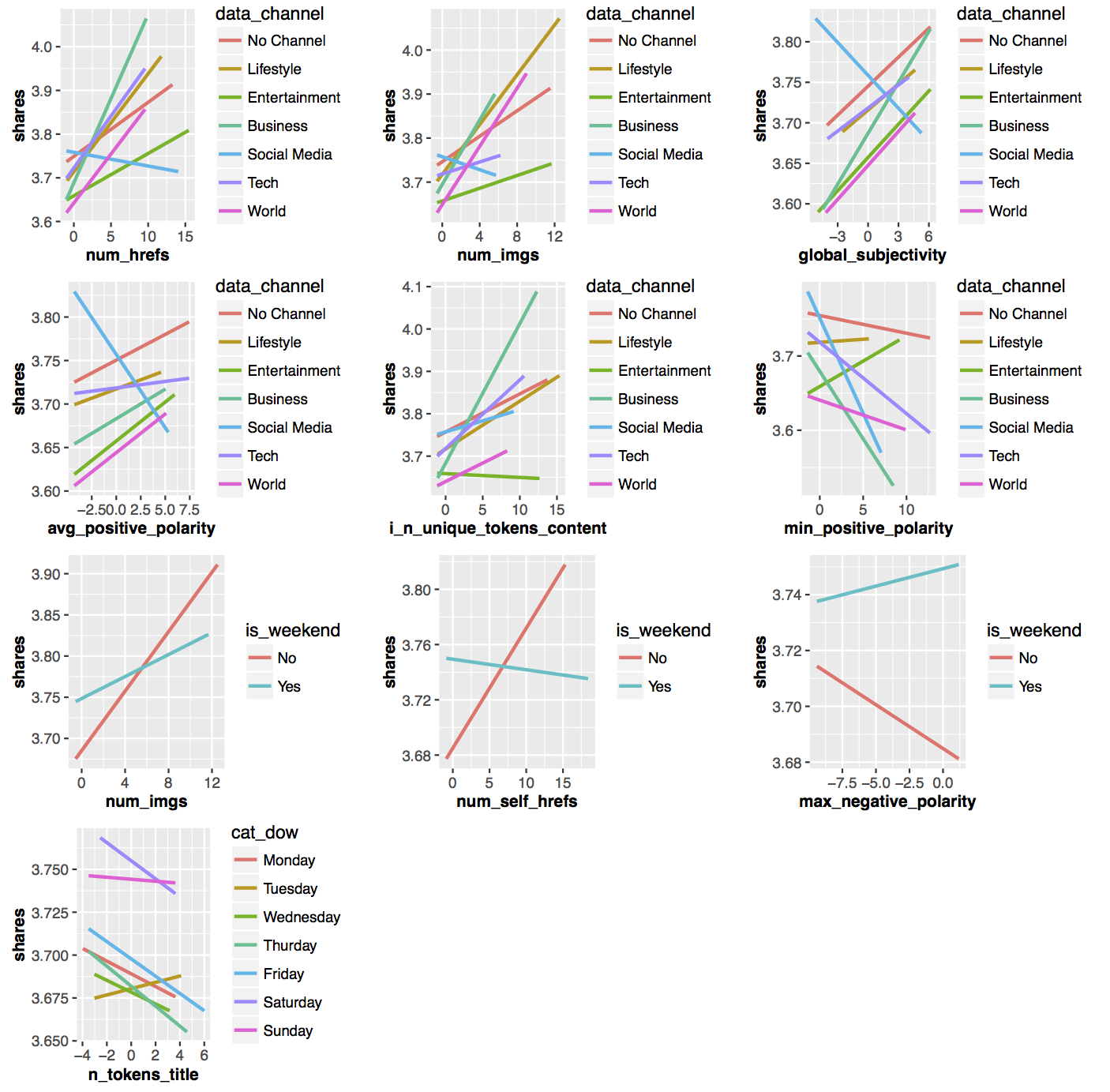
58. abs\_title\_subjectivity: Absolute subjectivity level

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$res) ~ predict(m.unweighted, data=news\_train)), data=news\_train)  
 # Second Approach, updating responce variable based upon weights and fold  
 #w <- (w - min(w))/(max(w) - min(w))  
 #news\_train$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), min=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 Shares (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 Number 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")  
  
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, intercept = 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.