ML 4375 Project 2 - Regression

Supratik Pochampally

Abstract

For my regression dataset, I chose the Online News Popularity dataset from the UCI Machine Learning Repository. The purpose of the dataset is to use attributes such as the sentiment polarity, rate of positive and negative words, and other features about articles published by Mashable in a period of two years to predict the number of shares in social networks that the article got (the popularity of the article).

Dataset

Let's start by reading in the dataset and printing the number of rows and column names:

```
# Read in the .csv file of the data set
df <- read.csv("OnlineNewsPopularity.csv", header = TRUE)
# Print number of rows
print(paste("Number of rows:", nrow(df)))
```

```
## [1] "Number of rows: 39644"
```

```
# Print attributes
names(df)
```

```
##  [1] "url"                          "timedelta"
##  [3] "n_tokens_title"               "n_tokens_content"
##  [5] "n_unique_tokens"              "n_non_stop_words"
##  [7] "n_non_stop_unique_tokens"     "num_hrefs"
##  [9] "num_self_hrefs"               "num_imgs"
## [11] "num_videos"                   "average_token_length"
## [13] "num_keywords"                 "data_channel_is_lifestyle"
## [15] "data_channel_is_entertainment" "data_channel_is_bus"
## [17] "data_channel_is_socmed"       "data_channel_is_tech"
## [19] "data_channel_is_world"        "kw_min_min"
## [21] "kw_max_min"                   "kw_avg_min"
## [23] "kw_min_max"                   "kw_max_max"
## [25] "kw_avg_max"                   "kw_min_avg"
## [27] "kw_max_avg"                   "kw_avg_avg"
## [29] "self_reference_min_shares"    "self_reference_max_shares"
## [31] "self_reference_avg_sharess"   "weekday_is_monday"
## [33] "weekday_is_tuesday"           "weekday_is_wednesday"
## [35] "weekday_is_thursday"          "weekday_is_friday"
## [37] "weekday_is_saturday"          "weekday_is_sunday"
## [39] "is_weekend"                   "LDA_00"
## [41] "LDA_01"                       "LDA_02"
## [43] "LDA_03"                       "LDA_04"
```

```
## [45] "global_subjectivity"           "global_sentiment_polarity"
## [47] "global_rate_positive_words"    "global_rate_negative_words"
## [49] "rate_positive_words"           "rate_negative_words"
## [51] "avg_positive_polarity"         "min_positive_polarity"
## [53] "max_positive_polarity"         "avg_negative_polarity"
## [55] "min_negative_polarity"         "max_negative_polarity"
## [57] "title_subjectivity"            "title_sentiment_polarity"
## [59] "abs_title_subjectivity"        "abs_title_sentiment_polarity"
## [61] "shares"
```

We see that the data set is large, with 39,644 rows and many columns, some being very ambiguously named.

Data cleaning

Link to source: https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity

Because the dataset is messy and needs cleaning, we will identify what predictors we want to use to predict our target column, being the "shares" column.

With a whopping 61 columns, it would be most efficient to go through them and identify what seem like the best contributing factors. We can quickly eliminate url, since it's just a long string that our algorithms would likely have trouble deciphering. Furthermore, there is no clear description of what the LDA_00 - LDA_04 and kw attributes are, so we should remove them to make sure we aren't using non-predictive columns.

Next, let's discuss what attributes would likely have an impact on number of shares. The actual length of the article and the number of images and videos contribute to the readability of an article, which could affect the number of shares. Furthermore, the data channel or topic of the article could possibly have an impact, since some articles of certain data channels would not be shared as often. The day the article was published may also contribute, since more people would be reading and sharing articles when they have time, such as over the weekend. Lastly, the subjectivity, polarity, and rate of positive and negative words may have an impact. Polarizing articles with a lot of positive and/or negative words may keep people wanting to discuss more about it, thus sharing it. Using this list, we can eliminate ambiguous and repetitive columns and only use what we deem valuable as features:

```
df <- df[, c(3, 4, 10, 11, 14, 15, 16, 17, 18, 19, 39, 45, 46, 47, 48, 59, 60, 61)]
names(df)
```

```
##  [1] "n_tokens_title"                "n_tokens_content"
##  [3] "num_imgs"                      "num_videos"
##  [5] "data_channel_is_lifestyle"     "data_channel_is_entertainment"
##  [7] "data_channel_is_bus"           "data_channel_is_socmed"
##  [9] "data_channel_is_tech"          "data_channel_is_world"
## [11] "is_weekend"                    "global_subjectivity"
## [13] "global_sentiment_polarity"     "global_rate_positive_words"
## [15] "global_rate_negative_words"    "abs_title_subjectivity"
## [17] "abs_title_sentiment_polarity"  "shares"
```

The reasoning for choosing these specific columns will be discussed in the feature selection section.

Let's also check if there are any NA or NaN values in the columns of the dataset:

```
colSums(is.na(df))
```

```
##                n_tokens_title              n_tokens_content
##                             0                             0
##                     num_imgs                    num_videos
##                             0                             0
##     data_channel_is_lifestyle data_channel_is_entertainment
##                             0                             0
##           data_channel_is_bus        data_channel_is_socmed
##                             0                             0
##          data_channel_is_tech         data_channel_is_world
##                             0                             0
##                    is_weekend           global_subjectivity
##                             0                             0
##     global_sentiment_polarity    global_rate_positive_words
##                             0                             0
##    global_rate_negative_words        abs_title_subjectivity
##                             0                             0
##  abs_title_sentiment_polarity                        shares
##                             0                             0
```

Luckily we have no NA or NaN values, so we can proceed without having to replace anything with the mean of the column.

Lastly, let's ensure all our columns are numeric or factors:

```
sapply(df, typeof)
```

```
##                n_tokens_title              n_tokens_content
##                      "double"                      "double"
##                     num_imgs                    num_videos
##                      "double"                      "double"
##     data_channel_is_lifestyle data_channel_is_entertainment
##                      "double"                      "double"
##           data_channel_is_bus        data_channel_is_socmed
##                      "double"                      "double"
##          data_channel_is_tech         data_channel_is_world
##                      "double"                      "double"
##                    is_weekend           global_subjectivity
##                      "double"                      "double"
##     global_sentiment_polarity    global_rate_positive_words
##                      "double"                      "double"
##    global_rate_negative_words        abs_title_subjectivity
##                      "double"                      "double"
##  abs_title_sentiment_polarity                        shares
##                      "double"                     "integer"
```

Since every column is numeric, we can continue to data exploration.

Data exploration

R functions

Let's use some R functions for data exploration.

```r
# Print the first 6 rows
head(df)
```

```
##   n_tokens_title n_tokens_content num_imgs num_videos data_channel_is_lifestyle
## 1             12              219        1          0                          0
## 2              9              255        1          0                          0
## 3              9              211        1          0                          0
## 4              9              531        1          0                          0
## 5             13             1072       20          0                          0
## 6             10              370        0          0                          0
##   data_channel_is_entertainment data_channel_is_bus data_channel_is_socmed
## 1                             1                  0                      0
## 2                             0                  1                      0
## 3                             0                  1                      0
## 4                             1                  0                      0
## 5                             0                  0                      0
## 6                             0                  0                      0
##   data_channel_is_tech data_channel_is_world is_weekend global_subjectivity
## 1                    0                     0          0           0.5216171
## 2                    0                     0          0           0.3412458
## 3                    0                     0          0           0.7022222
## 4                    0                     0          0           0.4298497
## 5                    1                     0          0           0.5135021
## 6                    1                     0          0           0.4374086
##   global_sentiment_polarity global_rate_positive_words
## 1                0.09256198                 0.04566210
## 2                0.14894781                 0.04313725
## 3                0.32333333                 0.05687204
## 4                0.10070467                 0.04143126
## 5                0.28100348                 0.07462687
## 6                0.07118419                 0.02972973
##   global_rate_negative_words abs_title_subjectivity
## 1                0.013698630             0.00000000
## 2                0.015686275             0.50000000
## 3                0.009478673             0.50000000
## 4                0.020715631             0.50000000
## 5                0.012126866             0.04545455
## 6                0.027027027             0.14285714
##   abs_title_sentiment_polarity shares
## 1                    0.1875000    593
## 2                    0.0000000    711
## 3                    0.0000000   1500
## 4                    0.0000000   1200
## 5                    0.1363636    505
## 6                    0.2142857    855
```

```r
# Display the internal structure of the data frame
str(df)
```

```
## 'data.frame':    39644 obs. of  18 variables:
##  $ n_tokens_title              : num  12 9 9 9 13 10 8 12 11 10 ...
##  $ n_tokens_content            : num  219 255 211 531 1072 ...
##  $ num_imgs                    : num  1 1 1 1 20 0 20 20 0 1 ...
```

```
##  $ num_videos                   : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ data_channel_is_lifestyle    : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ data_channel_is_entertainment: num  1 0 0 1 0 0 0 0 0 0 ...
##  $ data_channel_is_bus          : num  0 1 1 0 0 0 0 0 0 0 ...
##  $ data_channel_is_socmed       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ data_channel_is_tech         : num  0 0 0 0 1 1 0 1 1 0 ...
##  $ data_channel_is_world        : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ is_weekend                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ global_subjectivity          : num  0.522 0.341 0.702 0.43 0.514 ...
##  $ global_sentiment_polarity    : num  0.0926 0.1489 0.3233 0.1007 0.281 ...
##  $ global_rate_positive_words   : num  0.0457 0.0431 0.0569 0.0414 0.0746 ...
##  $ global_rate_negative_words   : num  0.0137 0.01569 0.00948 0.02072 0.01213 ...
##  $ abs_title_subjectivity       : num  0 0.5 0.5 0.5 0.0455 ...
##  $ abs_title_sentiment_polarity : num  0.188 0 0 0 0.136 ...
##  $ shares                       : int  593 711 1500 1200 505 855 556 891 3600 710 ...
```

```
# Print summary of dataset
summary(df)
```

```
##  n_tokens_title n_tokens_content    num_imgs         num_videos
##  Min.   : 2.0   Min.   :   0.0   Min.   :  0.000   Min.   : 0.00
##  1st Qu.: 9.0   1st Qu.: 246.0   1st Qu.:  1.000   1st Qu.: 0.00
##  Median :10.0   Median : 409.0   Median :  1.000   Median : 0.00
##  Mean   :10.4   Mean   : 546.5   Mean   :  4.544   Mean   : 1.25
##  3rd Qu.:12.0   3rd Qu.: 716.0   3rd Qu.:  4.000   3rd Qu.: 1.00
##  Max.   :23.0   Max.   :8474.0   Max.   :128.000   Max.   :91.00
##  data_channel_is_lifestyle data_channel_is_entertainment data_channel_is_bus
##  Min.   :0.00000           Min.   :0.000                 Min.   :0.0000
##  1st Qu.:0.00000           1st Qu.:0.000                 1st Qu.:0.0000
##  Median :0.00000           Median :0.000                 Median :0.0000
##  Mean   :0.05295           Mean   :0.178                 Mean   :0.1579
##  3rd Qu.:0.00000           3rd Qu.:0.000                 3rd Qu.:0.0000
##  Max.   :1.00000           Max.   :1.000                 Max.   :1.0000
##  data_channel_is_socmed data_channel_is_tech data_channel_is_world
##  Min.   :0.0000         Min.   :0.0000       Min.   :0.0000
##  1st Qu.:0.0000         1st Qu.:0.0000       1st Qu.:0.0000
##  Median :0.0000         Median :0.0000       Median :0.0000
##  Mean   :0.0586         Mean   :0.1853       Mean   :0.2126
##  3rd Qu.:0.0000         3rd Qu.:0.0000       3rd Qu.:0.0000
##  Max.   :1.0000         Max.   :1.0000       Max.   :1.0000
##    is_weekend     global_subjectivity global_sentiment_polarity
##  Min.   :0.0000   Min.   :0.0000      Min.   :-0.39375
##  1st Qu.:0.0000   1st Qu.:0.3962      1st Qu.: 0.05776
##  Median :0.0000   Median :0.4535      Median : 0.11912
##  Mean   :0.1309   Mean   :0.4434      Mean   : 0.11931
##  3rd Qu.:0.0000   3rd Qu.:0.5083      3rd Qu.: 0.17783
##  Max.   :1.0000   Max.   :1.0000      Max.   : 0.72784
##  global_rate_positive_words global_rate_negative_words abs_title_subjectivity
##  Min.   :0.00000            Min.   :0.000000           Min.   :0.0000
##  1st Qu.:0.02838            1st Qu.:0.009615           1st Qu.:0.1667
##  Median :0.03902            Median :0.015337           Median :0.5000
##  Mean   :0.03962            Mean   :0.016612           Mean   :0.3418
##  3rd Qu.:0.05028            3rd Qu.:0.021739           3rd Qu.:0.5000
##  Max.   :0.15549            Max.   :0.184932           Max.   :0.5000
```

```
##  abs_title_sentiment_polarity     shares
##  Min.   :0.0000               Min.   :     1
##  1st Qu.:0.0000               1st Qu.:   946
##  Median :0.0000               Median :  1400
##  Mean   :0.1561               Mean   :  3395
##  3rd Qu.:0.2500               3rd Qu.:  2800
##  Max.   :1.0000               Max.   :843300
```

```
# Print the average number of shares
print(paste("Average shares:", mean(df$shares)))
```

```
## [1] "Average shares: 3395.38018363435"
```

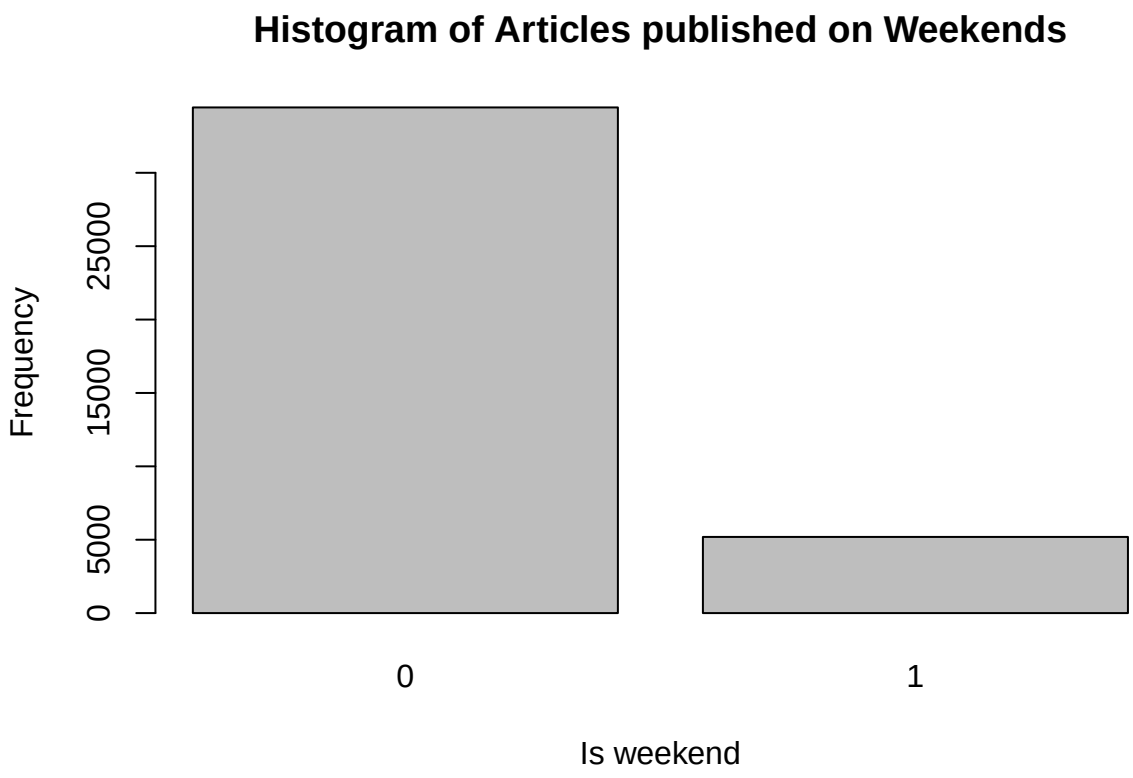
```
# Print the average global_subjectivity
print(paste("Average global subjectivity:", mean(df$global_subjectivity)))
```

```
## [1] "Average global subjectivity: 0.443370199550738"
```

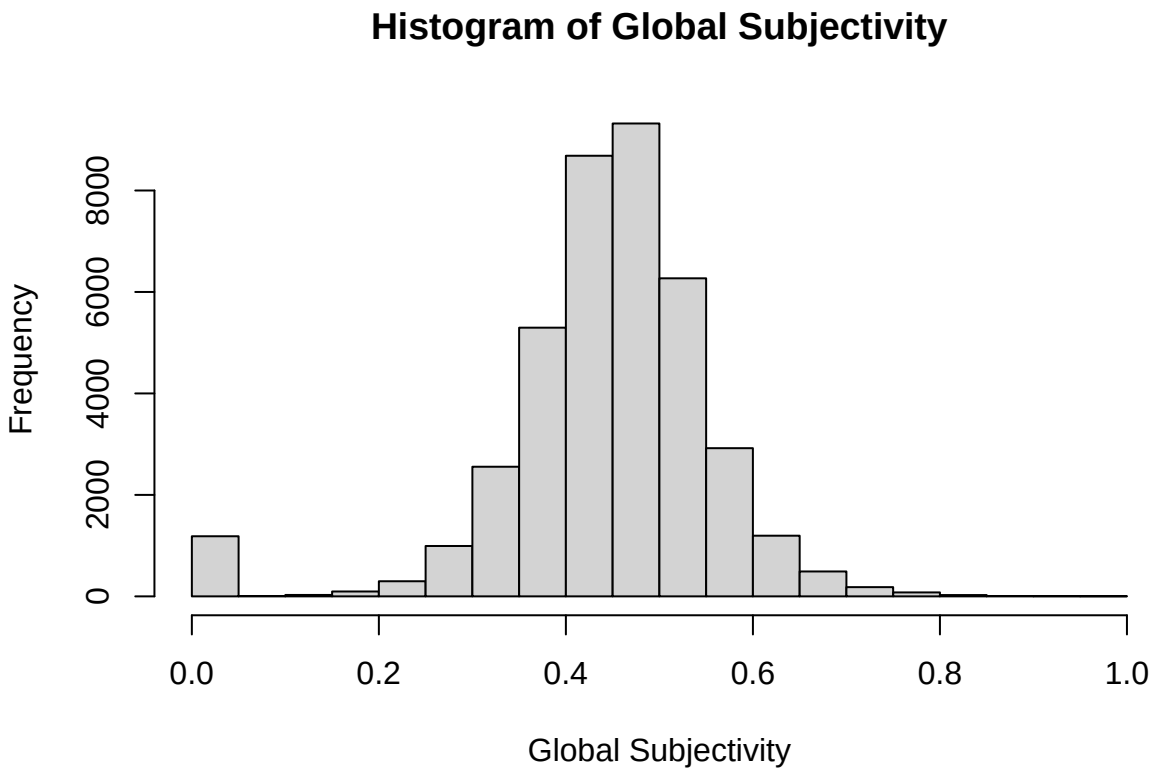
R graphs

Now let's create some informative R graphs for data exploration.

```
# Histogram of published on weekend or not
barplot(table(df$is_weekend), main = "Histogram of Articles published on Weekends", xlab = "Is weekend"
```



```r
# Histogram of global_subjectivity
hist(df$global_subjectivity, main = "Histogram of Global Subjectivity", xlab = "Global Subjectivity", yl
```



Based on the graphs, we can see a much higher frequency of articles published on weekdays than weekends. This imbalance may prove to be an issue when building the models. Furthermore, we can see that global_subjectivity has a normal distribution in terms of number of articles.

ML algorithms

We will attempt to run 3 regression algorithms over this dataset - Linear Regression, kNN regression, and SVM regression. Before running each algorithm, let's discuss feature selection.

Feature selection

The features in the dataset that are used to determine the number of shares in social networks are as follows and their justification is as follows:

- n_tokens_title - The length of the title may affect the readability of an article, and low readability may lead to less shares
- n_tokens_content - Similarly, a lengthy article may affect readability
- num_imgs - Images could keep an article engaging, potentially leading to more shares
- num_videos - Similarly, engaging videos could lead to more shares

- data_channel_is_lifestyle - Articles that are about lifestyle may have more shareable content, leading to more shares
- data_channel_is_entertainment - Articles that are about entertainment may have more shareable content, leading to more shares
- data_channel_is_bus - Articles that are about business may have more shareable content, leading to more shares
- data_channel_is_socmed - Articles that are about social media may have more shareable content, leading to more shares
- data_channel_is_tech - Articles that are about technology may have more shareable content, leading to more shares
- data_channel_is_world - Articles that are about the world may have more shareable content, leading to more shares
- is_weekend - Articles published over the weekend may coincide with people's schedules better, leading to more shares
- global_subjectivity - Subjective articles may be shared more to promote discussion
- global_sentiment_polarity - Polarizing articles may be shared more to promote discussion
- global_rate_positive_words - Articles with a high rate of positive words may lead people to share more (or not)
- global_rate_negative_words - Articles with a high rate of negative words may lead people to share more (or not)
- abs_title_subjectivity - Articles with subjective titles may be shared more to promote discussion
- abs_title_sentiment_polarity - Articles with polarizing titles may be shared more to promote discussion

We can now start implementing our regression algorithms. Let's begin by splitting our data into train and test sets of 75% and 25% respectively.

```
# Set seed to ensure the same split of training and testing sets
set.seed(1234)
# Split the data
i <- sample(1:nrow(df), nrow(df) * 0.75, replace = FALSE)
train <- df[i, ]
test <- df[-i, ]
```

Code to run linear regression

Now we can run linear regression over our predictors.

```
# Run linear regression
lm1 <- lm(shares~., data = train)
# Print summary of model
summary(lm1)
```

```
##
## Call:
## lm(formula = shares ~ ., data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -7443  -2182  -1426   -357 836732
##
## Coefficients:
```

```
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    3.598e+03  5.413e+02   6.647 3.05e-11 ***
## n_tokens_title                 9.737e+01  3.379e+01   2.882 0.003961 **
## n_tokens_content               1.940e-01  1.688e-01   1.149 0.250669
## num_imgs                       2.285e+01  9.494e+00   2.407 0.016095 *
## num_videos                     3.351e+01  1.778e+01   1.885 0.059391 .
## data_channel_is_lifestyle     -2.490e+03  3.635e+02  -6.851 7.49e-12 ***
## data_channel_is_entertainment -3.227e+03  2.485e+02 -12.984  < 2e-16 ***
## data_channel_is_bus           -2.680e+03  2.674e+02 -10.022  < 2e-16 ***
## data_channel_is_socmed        -2.456e+03  3.483e+02  -7.052 1.80e-12 ***
## data_channel_is_tech          -2.984e+03  2.540e+02 -11.745  < 2e-16 ***
## data_channel_is_world         -3.689e+03  2.543e+02 -14.505  < 2e-16 ***
## is_weekend                     4.719e+02  2.061e+02   2.289 0.022077 *
## global_subjectivity            2.559e+03  7.414e+02   3.452 0.000557 ***
## global_sentiment_polarity     -1.683e+03  1.233e+03  -1.364 0.172517
## global_rate_positive_words    -2.183e+03  5.945e+03  -0.367 0.713524
## global_rate_negative_words    -1.699e+04  9.429e+03  -1.802 0.071611 .
## abs_title_subjectivity         1.058e+03  4.100e+02   2.582 0.009836 **
## abs_title_sentiment_polarity   1.066e+03  3.394e+02   3.140 0.001691 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11940 on 29715 degrees of freedom
## Multiple R-squared:  0.01271,    Adjusted R-squared:  0.01214
## F-statistic:  22.5 on 17 and 29715 DF,  p-value: < 2.2e-16
```

Looking at the summary of the model, we can conclude that the data channel that the article is published in and global subjectivity are the most contributing factors to the number of shares. Sentiment polarity, title subjectivity & polarity, global rate of positive & negative words, number of images & videos, and title and content length have little to no impact on the number of shares.

Now let's test the model over the testing set.

```
# Make predictions based on the model
pred1 <- predict(lm1, newdata = test)
```

Linear regression metrics

These predictions compute to be the following metrics:

```
# Calculate and print the metrics of the predictions
cor1 <- cor(pred1, test$shares)
print(paste("Correlation:", cor1))
```

```
## [1] "Correlation: 0.0962160027408114"
```

```
mse1 <- mean((pred1 - test$shares)^2)
print(paste("MSE:", mse1))
```

```
## [1] "MSE: 107099708.216446"
```

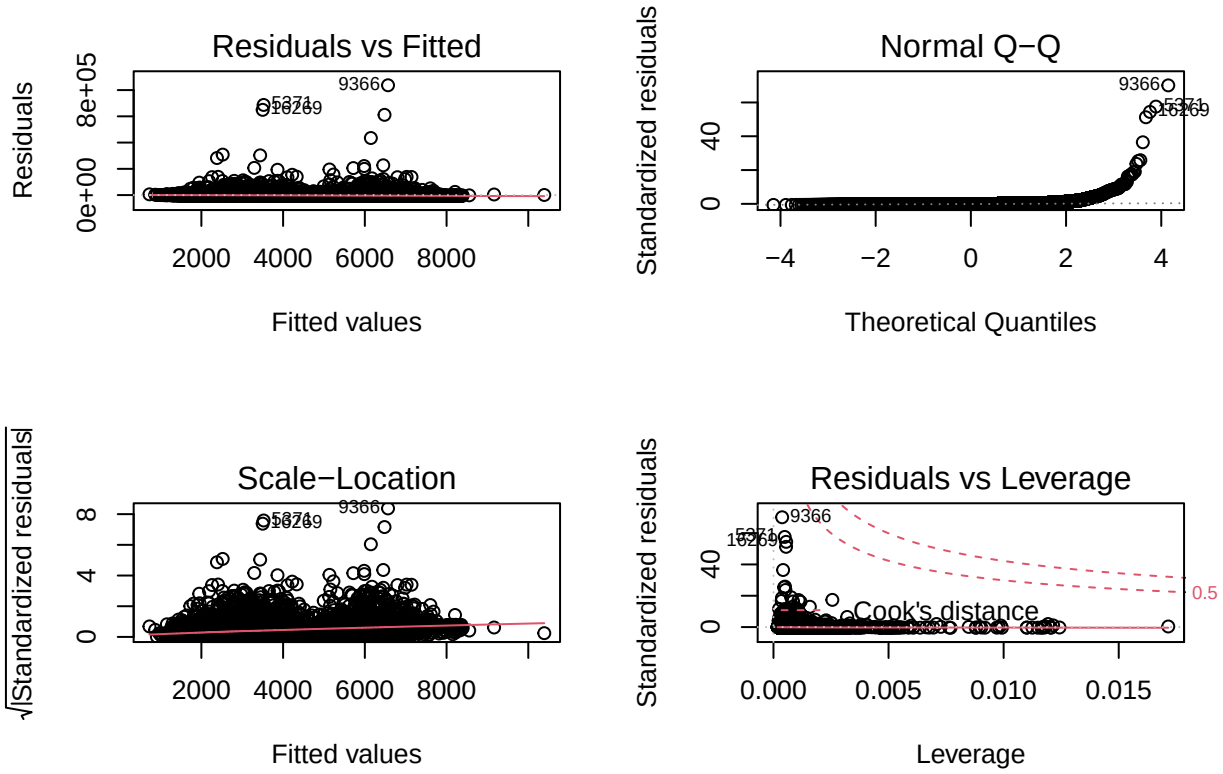
```
rmse1 <- sqrt(mse1)
print(paste("RMSE:", rmse1))
```

```
## [1] "RMSE: 10348.8988890822"
```

We see fairly poor performance when looking at all the metrics. A correlation of 0.1 is very low, and the MSE and RMSE of 107099708 and 10349 show that the linear regression model is easily prone to error.

Let's try plotting the residuals of the linear regression model:

```
par(mfrow=c(2, 2))
plot(lm1)
```



1) Residuals vs. Fitted - We see a fairly horizontal trend line, which means we captured most of the variation in the data
2) Normal Q-Q - The diagonal line is not diagonally straight, meaning that the residuals are not normally distributed
3) Scale-Location - The line is fairly horizontal, but the points are not distributed equally around it, meaning that the data may not be homoscedastic
4) Residuals vs. Leverage: We see both leverage points and outliers

Code for kNN regression

Next we will try kNN regression using the same training and testing set from before so we get the most accurate comparison. Let's use 200 as our k-value, since that's the approximate square root of the number of rows in our data set.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
fit <- knnreg(train[, 1:17], train[, 18], k = 200)
```

Now let's test the model over the testing set.

```
# Make predictions based on the model
pred2 <- predict(fit, test[, 1:17])
cor2 <- cor(pred2, test$shares)
print(paste("Correlation:", cor2))
```

```
## [1] "Correlation: 0.0852066146953678"
```

Let's see if scaling our data improves performance:

```
train_scaled <- train[, 1:17]
means1 <- sapply(train_scaled, mean)
stdvs1 <- sapply(train_scaled, sd)
train_scaled <- scale(train_scaled, center=means1, scale=stdvs1)
test_scaled <- scale(test[, 1:17], center=means1, scale=stdvs1)

fit2 <- knnreg(train_scaled, train$shares, k = 200)
pred3 <- predict(fit2, test_scaled)
cor3 <- cor(pred3, test$shares)
print(paste("Correlation:", cor3))
```

```
## [1] "Correlation: 0.0913221461394832"
```

Since scaling improves performance, let's used the scaled model.

kNN regression metrics

These predictions compute to be the following metrics:

```
# Calculate and print the metrics of the predictions
print(paste("Correlation:", cor3))
```

```
## [1] "Correlation: 0.0913221461394832"
```

```
mse2 <- mean((pred3 - test$shares)^2)
print(paste("MSE:", mse2))
```

```
## [1] "MSE: 107610882.238891"
```

```
rmse2 <- sqrt(mse2)
print(paste("RMSE:", rmse2))
```

```
## [1] "RMSE: 10373.5665148921"
```

Once again, we see fairly poor performance when looking at all the metrics. A correlation of 0.09 is very low, even lower than our linear regression model. The MSE and RMSE of 107610882 and 10374 show that the kNN regression model is easily prone to error, even more than the linear regression model.

Code for SVM regression

Lastly, we will try radial SVM regression once again using the same training and testing set from before so we get the most accurate comparison.

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.4
```

```
svm1 <- svm(shares~., data=train, kernel="radial", cost=1, gamma=1, scale=FALSE)
summary(svm1)
```

```
##
## Call:
## svm(formula = shares ~ ., data = train, kernel = "radial", cost = 1,
##     gamma = 1, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  29716
```

We see that there were 29716 support vectors generated overall. Now let's test the model over the testing set.

```
pred4 <- predict(svm1, newdata=test)
```

kNN regression metrics

These predictions compute to be the following metrics:

```r
cor_svm1 <- cor(pred4, test$shares)
print(paste("Correlation:", cor_svm1))
```

```
## [1] "Correlation: 0.0183464375535451"
```

```r
mse_svm1 <- mean((pred4 - test$shares)^2)
print(paste("MSE:", mse_svm1))
```

```
## [1] "MSE: 111766833.796079"
```

```r
rmse_svm1 <- sqrt(mse_svm1)
print(paste("RMSE:", rmse_svm1))
```

```
## [1] "RMSE: 10571.9834371833"
```

Similar to the first two models, we once again see fairly poor performance when looking at all the metrics. A correlation of 0.02 is very low, the worst of the three models we tested. The MSE and RMSE of 111766834 and 10572 show that the SVM regression model is easily prone to error, even more than the kNN and linear regression models.

Results analysis

When analyzing the results of the algorithms, it is important to take into account all the metrics that we calculated earlier. These being correlation, MSE, and RMSE.

Performance and rankings

Linear regression showed to have the highest correlation of the three models of 0.1, followed by kNN regression at 0.09 and SVM regression at 0.02. Linear regression also had the best MSE and RMSE of 107099708 and 10349, followed by kNN regression's values of 107610882 and 10373, and SVM regression's values of 111766833 and 10572. Based on all these metrics, I believe the following is the correct rankings for the three algorithms:

1) Linear Regression
2) kNN regression
3) SVM regression

Linear regression is clearly the best performing algorithm because it has the best correlation value and MSE and RMSE values, showing that it has the best predictive value and is the least prone to errors. I believe that kNN regression performs better than SVM regression because it has a better correlation and MSE and RMSE values.

Why Linear Regression was the best

Let's compare linear regression with the kNN regression and SVM regression algorithms to see why it may have performed better in our data set. Let's also compare kNN and SVM regression, since there was a large disparity between the two.

Linear Regression vs. kNN regression

The biggest benefit of linear regression when comparing it to kNN regression is that linear regression is better at handling noise. In a dataset with so many features and observations, we probably have to handle a lot of noise as opposed to signals in our data. Because of this, the linear regression model may have shown to have a better correlation and better MSE and RMSE values compared to kNN regression. Furthermore, linear regression is a parametric model while kNN regression is a non-parametric model. Because of this, linear regression is much faster at recognizing and computing the coefficients for each feature, while kNN is much slower and has to keep track of all the training data to find the neighbor nodes. This once again will not be as good at handling noise that may interfere with the neighbor field of each node being trained and tested through kNN.

Linear Regression vs. SVM regression

Just looking at the number of support vectors our SVM regression algorithm created, which was 29716, we immediately knew that there was little to no regularity in our data. This meant that using the support vectors to try and find correlations between our attributes and target variable would be very difficult, as almost each attribute had it's own support vector. Likely because of this, we saw the lowest correlation for our SVM regression of 0.02. Our SVM regression algorithm seemed to have overfit our data set by a large margin, rendering it almost useless compared to our linear regression model. Although SVM takes care of outliers better than linear regression, we saw in our residual plots that there were not many outliers in our data set, meaning that we didn't have to worry about this.

kNN regression vs. SVM regression

I wanted to compare these two algorithms to specifically point out that SVM regression is much worse at handling a large training set compared to kNN regression. kNN regression is the most effective when the number of observations is drastically greater than the number of features. Although SVM regression takes care of outliers better than kNN regression, we saw in our residual plots that there were not many outliers in our data set, meaning that we didn't have to worry about this.

Big picture

Although our regression models by themselves would not be very effective at predicting the number of shares on social media for an article based on it's features, we still learned a lot from attempting to run regression models over this data set. We learned that the data channel or topic of an article has a significant impact on the number of shares it gets on social media. Articles about entertainment, business, technology, and the world seem to be the most impactful. Furthermore, the subjectivity of the actual content of the article has much more to do with the number of shares than the subjectivity of the title. Instead, the length of the title has a lot more to do with the number of shares than the actual title. With this information, writers and publishers would know to make sure the title is short and catchy and instead focus on making the actual content of the articles subjective in order to increase the number of shares on social media.