R Project - Supervised Learning for shares forecasting

Machine Learning in Management - year 2022/23

Matteo Scarciglia

Contents

Starting point: The problem

Taking decisions in management side can be a process more or less articulated, with different grade of difficult. Independently from the specific circumstance, in business administration the decisions always turn around an efficiency problem that can be solved by two order of solutions: reduce the costs or increase the profit. As already demonstrated, making intelligence is maybe the only way to handle the problem nowadays, in this competitive scenario. In the first part of the project was shown how to cluster products which had a similar behavior in sales during a certain period of time, with the aim of explaining, for example, the effects produced by a marketing decision or even to find a common seasonality among products which was not obvious before. This kind of problem can be solved via unsupervised learning, which consists in clustering the observation with the aim of setting up an useful alarm for the managers, whom will take all the decisions.

In this part of the project is instead shown how to deal with the decision process using numbers, with supervised learning. Differently from unsupervised learning who simply clusters observation which are similar, with supervised learning given all the input variables a specific number as output is computed. This lead to have not only a precise numerical output computed as sum of feature, but it is also possible to comprehend how the input variables (known as predictor) precisely affects the output, and this computing process is known as regression problem. In this case, the management is supported by a more powerful tool that can lead to have a measure of what happens in reality and lastly a more accurate decision.

The management problem discussed here is relative to how the features of a user generated content upon a social digital platform affects the shares of the content. Hence, a supervised learning algorithm is applied

to explain how every feature influences the number of shares. Knowing more or less precisely the weight of every predictor can bring knowledge about what should be improved and even enables to a quite accurate forecasts. Below, the dataset will be explained.

Data Description - Importing Dataset

The dataset was published on UC Irvine Machine Learning Repository, which is the public archive of the University of California Irvine. Here are collected some features of 39 644 articles published by Mashable (www.mashable.com) in a period of two years, acquired on January 8, 2015. The dataset contains 61 attributes, 2 of whom two are not predictive, 58 are predictive and 1 is the goal field.

The predictive variables concern about the features of the content and are listed below. As first, the dataset will be imported:

```
dataset_gross <- read.csv("OnlineNewsPopularity.csv")
dim(dataset_gross)
```

```
## [1] 39644    61
```

```
names(dataset_gross)
```

```
##  [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"
```

As can be see, there are quite a few variables and among these, some are more important than others obviously. About this, a supervised learning algorithm can help to choose all the variables who really matter in case of creating new contents. To solve this, some model will be created and the best will be chosen among best subset selection or lasso regression.

The target variable is the shares variable and the others represent some characteristics of the articles, such as n_tokens_title (the number of words in the title), n_unique_tokens (the rate of unique words in the content) and so on. Further information about the dataset are available here: https://archive-beta.ics.uci.edu/dataset/332/online+news+popularity.

Transform, Tidy and Prepare the Dataset

The first best practice as always is checking if there are missing values:

```
sum(is.na(dataset_gross))
```

```
## [1] 0
```

There are not missing values, so a deep cleaning is not required; as mentioned, there are two non predictive attributes which are located in the first two columns of the dataset. These attributes are url, which is the url of the article and timedelta, which represent the days between the article publication and the dataset acquisition. Since these are not useful in this analysis will be deleted:

```
dataset <- dataset_gross[,-(1:2)]
names(dataset)
```

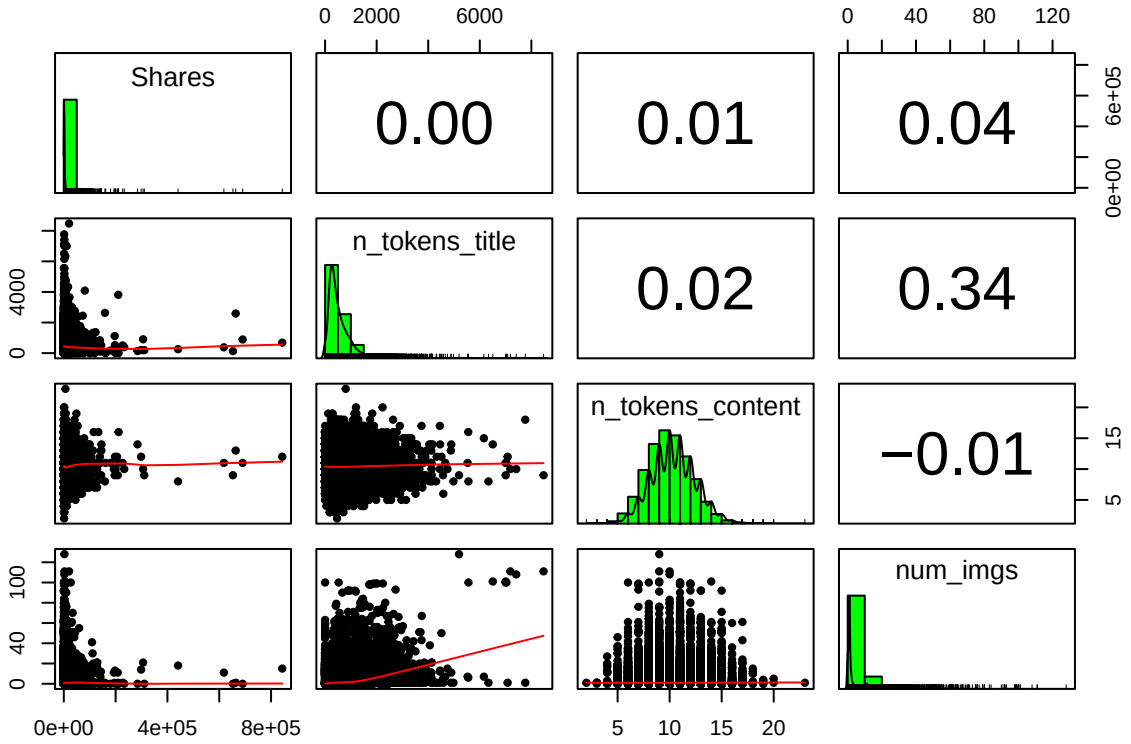
```
##  [1] "n_tokens_title"                "n_tokens_content"
##  [3] "n_unique_tokens"               "n_non_stop_words"
##  [5] "n_non_stop_unique_tokens"      "num_hrefs"
##  [7] "num_self_hrefs"                "num_imgs"
##  [9] "num_videos"                    "average_token_length"
## [11] "num_keywords"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

```
## [53] "min_negative_polarity"         "max_negative_polarity"
## [55] "title_subjectivity"            "title_sentiment_polarity"
## [57] "abs_title_subjectivity"        "abs_title_sentiment_polarity"
## [59] "shares"
```

Knowing more about data

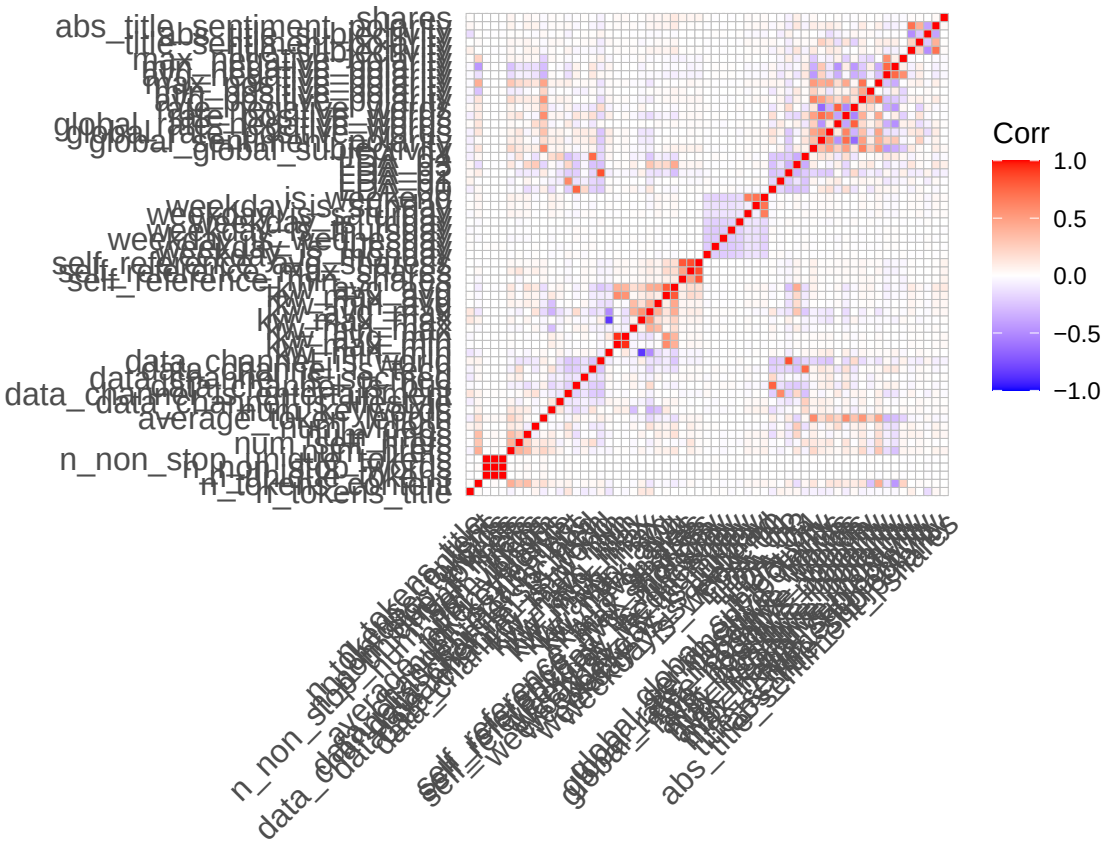
Considering that the high number of feature is not possible to represent them without losing information. Moreover, since the nature of the information is unknown building random histograms would be useless. Just for the record, when less variables are recorded could be instead useful using the R's function pairs.panels() of the psych package. This function plot histograms of the data on the diagonal, bivariate scatter plots below the diagonal and the Pearson correlation above the diagonal. Unfortunately, this is only useful for descriptive statistics of small data sets and is here built just for notice it's existence:

```
library(psych)

pp <- data.frame(dataset$shares, dataset$n_tokens_content,
                 dataset$n_tokens_title, dataset$num_imgs)
# OCCHIO CHE QUI FORSE C?é BUG

colnames(pp) <- c("Shares","n_tokens_title","n_tokens_content",
                      "num_imgs")


pairs.panels(pp, method="pearson", hist.col="green", density=TRUE,
             ellipses=FALSE)
```



Another try could be build a correlation heatmap of the variables instead, for finding correlation among variables. This could be useful for give an idea about the features and whether they are or not independent from each other:

```
library(ggcorrplot)

# plotting corr heatmap # - QUESTA è MEGLIO
heatmap_1 <- ggcorrplot::ggcorrplot(cor(dataset[,],dataset[,]))

heatmap_1
```



This can be useful but still confusing. Thus, considering the number of attributes building directly a model can be more convenient.

The Regression model

As mentioned in the beginning when the output of a problem consists in a number, this has to be considered as a regression problem who can be solved by a regression model, that compute an output considering as input certain features; More precisely compute a model means to adapt himself to the actual measured observation via an algorithm. The resultant is a mathematical function of the form:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \ \epsilon$$

Where the $X_i (1 \leq i \leq p)$ are the values of the variables (predictors), Y is the target variable while B_j is the parameter for every variable to multiply for. This latter is interpreted as the effect to the model (on average) for a unitary change in X_i holding fixed all the other predictors. Aditionally, ϵ is the erratic component, which is the difference among the model and the reality that is not statistically explainable.

The main issue here is to estimate the coefficients of the model, which differs from the actual since they will always be just an estimate of the real function related to the features (actually, B_j is unknown). Let $\hat{B}_j$ to be the estimate of the coefficient B_j, x the observation and $\hat{y}$ the predicted value of Y based on the $X = x$, the estimated model will be:

$$\hat{y} = \hat{B}_0 \ + \hat{B}_1 X_1 \ + \ \ldots \ + \hat{B}_P X_n$$

Now let $(x_1, y_1), (x_2, y_2), \ \ldots (x_n, y_n)$ represent the n observation pairs that measure X and Y. The aim is definitively to obtain coefficients $\hat{B}_j$ such that the linear model fits the data as well as it is possible:

$$y_i \approx \hat{B}_0 \ + \hat{B}_1 x_i \ + \ ... \ + \ \hat{B}_p x_n$$

There are several ways to estimate the coefficients of the model but the most commonly adopted is the OLS method, which stand for Ordinary Least Squares. The aim is to find an intercept and a slope that is as close as possible to the data points, computed via minimizing the RSS (Residual Sum of Squared), here below defined:

$$RSS = \sum_{i=1}^{n} (y_i - B_0 - B_1 x_1 \ - \ ... \ - B_n x_n)^2$$

The coefficients $\hat{B}_j$ estimated are the values that minimize the following loss function:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - B_0 - B_1 x_1 - \ ... \ B_n x_n)^2$$

Before starting with effectively building the model it is a good practice to look for collinearity among the variables, which is a condition that hinder the estimation of the model and considering the presence of dummy variables, collinearity is quite surely present. The condition of collinearity (independently of the presence of dummy variables) occurs when two or more attributes are highly related to each other and this can affect the model negatively making it not more reliable. In presence of collinearity, in fact, since two or more variables are correlated a change (even small) in one causes some changes in others. The result is that a little change in one variables summed to a change in others causes a bigger change in the whole model, **inflating the variance**. This makes the model too sensitive and for this reason too little reliable.

Without getting deeper into the topic, the presence of encoded dummy variables in the dataset leads to a trap which is called dummy variable trap, which causes problem in the estimating process. Usually the solution is to remove one of the dummy variables, for instance removing the last day of the week if there are 7 dummy variables indicating the weekday as in this case, or otherwise removing the intercept of the model.

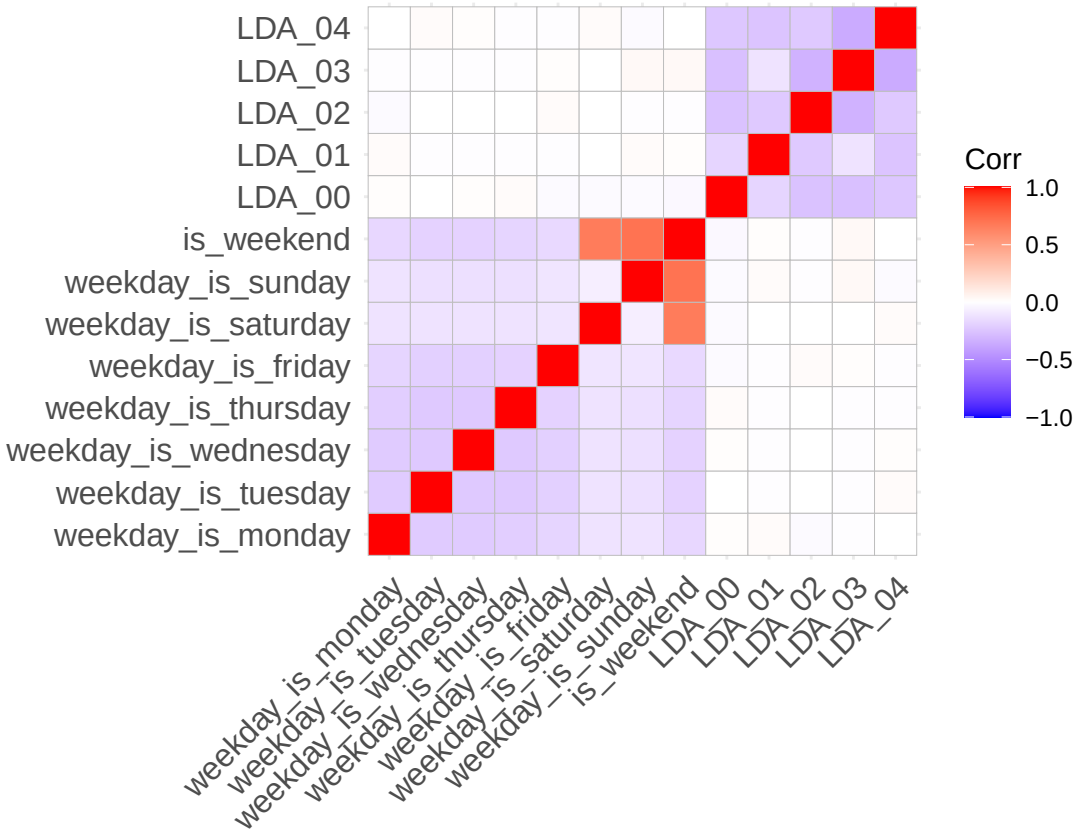
Before going any further, first go check for collinearity:

```
library(plm)

detect.lindep(dataset)
```

```
## [1] "Suspicious column number(s): 35, 36, 37"
## [1] "Suspicious column name(s):   weekday_is_saturday, weekday_is_sunday, is_weekend"
```

As seen, detect.lindep() function found 3 linear dependences among which means that collinearity is present among some attributes. The variables involves are in fact *weekday_is_saturday*, *weekday_is_sunday* and *is_weekend* which are all encoded as dummy variables. Let's call a correlation heat map of those variables for checking whether all of them are actually high correlated or just some of those:

```
library(ggcorrplot)

heatmap_2 <- ggcorrplot::ggcorrplot(
  cor(dataset[,c(30:42)],dataset[,c(30:42)])
  )
heatmap_2
```



The heatmap says that the highest correlation the attributes is among *is_weekend* attribute and *weekday_is_saturday / weekday_is_sunday* and this represent a problem; Probably this problem can be solved removing the is_weekend variables which causes the dummy variable's trap:

```
dataset <- dataset[,-c(37)]

detect.lindep(dataset)
```

```
## [1] "No linear dependent column(s) detected."
```

Without removing collinearity, the function that will be used for estimate the model would not be able to compute the coefficients, yielding NA's. This process hence is compulsory for building a reliable model. Now that finally collinearity is removed, the model can be built.

Building the Regression model

Now that all the preparation's operations are made, it is possible to proceed with building the model. The function in R that builds linear models is lm() function, who fit the model via least squares method. Usually the construction process starts from creating a model with 0 variables and adding one at time and trying different combination until the best model is found, choosing the best by the lowest RSS or the highest R^2 (this latter will be explained throughout this section). Another common iterative process is to start with a model containing all the variables and removing one of them at time, essentially in reverse way. Before see which is the best model among the possibles, let's build the model using lm() function and creating a model with all the variables:

```r
library(stats)

set.seed(1)

model_all <- lm(data=dataset, shares ~ .)

summary(model_all)
```

```
##
## Call:
## lm(formula = shares ~ ., data = dataset)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -29737  -2255  -1220   -107 837629
##
## Coefficients: (1 not defined because of singularities)
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -1.763e+05  6.130e+06  -0.029  0.97706
## n_tokens_title                 8.986e+01  2.867e+01   3.134  0.00173 **
## n_tokens_content               5.936e-01  2.235e-01   2.656  0.00791 **
## n_unique_tokens                3.985e+03  1.919e+03   2.077  0.03779 *
## n_non_stop_words              -1.484e+03  5.911e+03  -0.251  0.80182
## n_non_stop_unique_tokens      -1.641e+03  1.629e+03  -1.007  0.31405
## num_hrefs                      2.654e+01  6.706e+00   3.958 7.58e-05 ***
## num_self_hrefs                -5.764e+01  1.782e+01  -3.235  0.00122 **
## num_imgs                       1.190e+01  8.942e+00   1.330  0.18336
## num_videos                     5.645e+00  1.575e+01   0.358  0.72006
## average_token_length          -5.867e+02  2.429e+02  -2.416  0.01570 *
## num_keywords                   4.949e+01  3.714e+01   1.333  0.18262
## data_channel_is_lifestyle     -1.050e+03  3.947e+02  -2.661  0.00780 **
## data_channel_is_entertainment -1.180e+03  2.552e+02  -4.626 3.74e-06 ***
## data_channel_is_bus           -8.023e+02  3.827e+02  -2.096  0.03605 *
## data_channel_is_socmed        -6.029e+02  3.724e+02  -1.619  0.10542
## data_channel_is_tech          -5.509e+02  3.714e+02  -1.483  0.13800
## data_channel_is_world         -4.831e+02  3.764e+02  -1.284  0.19931
## kw_min_min                     2.209e+00  1.623e+00   1.361  0.17364
## kw_max_min                     8.718e-02  5.012e-02   1.739  0.08201 .
## kw_avg_min                    -3.468e-01  3.078e-01  -1.127  0.25990
## kw_min_max                    -2.067e-03  1.174e-03  -1.761  0.07819 .
## kw_max_max                    -5.153e-04  5.786e-04  -0.890  0.37321
## kw_avg_max                    -7.186e-04  8.293e-04  -0.867  0.38621
## kw_min_avg                    -3.659e-01  7.567e-02  -4.836 1.33e-06 ***
## kw_max_avg                    -2.026e-01  2.530e-02  -8.010 1.18e-15 ***
## kw_avg_avg                     1.662e+00  1.438e-01  11.558  < 2e-16 ***
## self_reference_min_shares      2.615e-02  7.525e-03   3.476  0.00051 ***
## self_reference_max_shares      5.762e-03  4.083e-03   1.411  0.15824
## self_reference_avg_sharess    -5.779e-03  1.044e-02  -0.554  0.57989
## weekday_is_monday              2.603e+02  2.631e+02   0.989  0.32248
## weekday_is_tuesday            -2.785e+02  2.592e+02  -1.074  0.28265
## weekday_is_wednesday          -1.125e+02  2.592e+02  -0.434  0.66433
## weekday_is_thursday           -2.889e+02  2.597e+02  -1.112  0.26599
## weekday_is_friday             -2.508e+02  2.690e+02  -0.932  0.35113
```

```
## weekday_is_saturday            3.846e+02  3.206e+02   1.200  0.23026
## weekday_is_sunday                     NA         NA      NA       NA
## LDA_00                         1.762e+05  6.130e+06   0.029  0.97707
## LDA_01                         1.753e+05  6.130e+06   0.029  0.97719
## LDA_02                         1.749e+05  6.130e+06   0.029  0.97724
## LDA_03                         1.757e+05  6.130e+06   0.029  0.97713
## LDA_04                         1.757e+05  6.130e+06   0.029  0.97713
## global_subjectivity            2.470e+03  8.506e+02   2.905  0.00368 **
## global_sentiment_polarity      6.789e+02  1.668e+03   0.407  0.68391
## global_rate_positive_words    -1.343e+04  7.165e+03  -1.874  0.06090 .
## global_rate_negative_words     2.097e+03  1.368e+04   0.153  0.87813
## rate_positive_words            2.117e+03  5.776e+03   0.367  0.71396
## rate_negative_words            2.003e+03  5.822e+03   0.344  0.73084
## avg_positive_polarity         -1.614e+03  1.367e+03  -1.181  0.23760
## min_positive_polarity         -1.965e+03  1.144e+03  -1.717  0.08600 .
## max_positive_polarity          3.492e+02  4.311e+02   0.810  0.41796
## avg_negative_polarity         -1.723e+03  1.259e+03  -1.369  0.17092
## min_negative_polarity          1.295e+02  4.589e+02   0.282  0.77781
## max_negative_polarity         -1.828e+02  1.047e+03  -0.175  0.86137
## title_subjectivity            -1.002e+02  2.742e+02  -0.365  0.71479
## title_sentiment_polarity       2.128e+02  2.504e+02   0.850  0.39548
## abs_title_subjectivity         6.445e+02  3.641e+02   1.770  0.07669 .
## abs_title_sentiment_polarity   6.108e+02  3.958e+02   1.544  0.12271
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11500 on 39587 degrees of freedom
## Multiple R-squared:  0.0231, Adjusted R-squared:  0.02172
## F-statistic: 16.71 on 56 and 39587 DF,  p-value: < 2.2e-16
```

Unfortunately, the model yielded 1 not defined coefficients (weekday_is_sunday attribute is again involved) because of singularities. Probably R is still sensitive respect the dummy variable's trap that clearly has not been removed; let's try to solve this removing the intercept of the model adding 0 in formula argument of the function:

```
model_all <- lm(data=dataset, shares ~  0 + .)

summary(model_all)
```

```
##
## Call:
## lm(formula = shares ~ 0 + ., data = dataset)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -29737  -2255  -1220   -107 837629
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## n_tokens_title                 8.986e+01  2.867e+01   3.134  0.00173 **
## n_tokens_content               5.936e-01  2.235e-01   2.656  0.00791 **
## n_unique_tokens                3.985e+03  1.919e+03   2.077  0.03779 *
## n_non_stop_words              -1.484e+03  5.911e+03  -0.251  0.80182
```

```
## n_non_stop_unique_tokens      -1.641e+03  1.629e+03  -1.007  0.31405
## num_hrefs                      2.654e+01  6.706e+00   3.958 7.58e-05 ***
## num_self_hrefs                -5.764e+01  1.782e+01  -3.235  0.00122 **
## num_imgs                       1.190e+01  8.942e+00   1.330  0.18336
## num_videos                     5.645e+00  1.575e+01   0.358  0.72006
## average_token_length          -5.867e+02  2.429e+02  -2.416  0.01570 *
## num_keywords                   4.949e+01  3.714e+01   1.333  0.18262
## data_channel_is_lifestyle     -1.050e+03  3.947e+02  -2.661  0.00780 **
## data_channel_is_entertainment -1.180e+03  2.552e+02  -4.626 3.74e-06 ***
## data_channel_is_bus           -8.023e+02  3.827e+02  -2.096  0.03605 *
## data_channel_is_socmed        -6.029e+02  3.724e+02  -1.619  0.10542
## data_channel_is_tech          -5.509e+02  3.714e+02  -1.483  0.13800
## data_channel_is_world         -4.831e+02  3.764e+02  -1.284  0.19931
## kw_min_min                     2.209e+00  1.623e+00   1.361  0.17364
## kw_max_min                     8.718e-02  5.012e-02   1.739  0.08201 .
## kw_avg_min                    -3.468e-01  3.078e-01  -1.127  0.25990
## kw_min_max                    -2.067e-03  1.174e-03  -1.761  0.07819 .
## kw_max_max                    -5.153e-04  5.786e-04  -0.890  0.37321
## kw_avg_max                    -7.186e-04  8.293e-04  -0.867  0.38621
## kw_min_avg                    -3.659e-01  7.567e-02  -4.836 1.33e-06 ***
## kw_max_avg                    -2.026e-01  2.530e-02  -8.010 1.18e-15 ***
## kw_avg_avg                     1.662e+00  1.438e-01  11.558  < 2e-16 ***
## self_reference_min_shares      2.615e-02  7.525e-03   3.476  0.00051 ***
## self_reference_max_shares      5.762e-03  4.083e-03   1.411  0.15824
## self_reference_avg_sharess    -5.779e-03  1.044e-02  -0.554  0.57989
## weekday_is_monday             -1.760e+05  6.130e+06  -0.029  0.97710
## weekday_is_tuesday            -1.765e+05  6.130e+06  -0.029  0.97703
## weekday_is_wednesday          -1.764e+05  6.130e+06  -0.029  0.97705
## weekday_is_thursday           -1.765e+05  6.130e+06  -0.029  0.97702
## weekday_is_friday             -1.765e+05  6.130e+06  -0.029  0.97703
## weekday_is_saturday           -1.759e+05  6.130e+06  -0.029  0.97711
## weekday_is_sunday             -1.763e+05  6.130e+06  -0.029  0.97706
## LDA_00                         1.762e+05  6.130e+06   0.029  0.97707
## LDA_01                         1.753e+05  6.130e+06   0.029  0.97719
## LDA_02                         1.749e+05  6.130e+06   0.029  0.97724
## LDA_03                         1.757e+05  6.130e+06   0.029  0.97713
## LDA_04                         1.757e+05  6.130e+06   0.029  0.97713
## global_subjectivity            2.470e+03  8.506e+02   2.905  0.00368 **
## global_sentiment_polarity      6.789e+02  1.668e+03   0.407  0.68391
## global_rate_positive_words    -1.343e+04  7.165e+03  -1.874  0.06090 .
## global_rate_negative_words     2.097e+03  1.368e+04   0.153  0.87813
## rate_positive_words            2.117e+03  5.776e+03   0.367  0.71396
## rate_negative_words            2.003e+03  5.822e+03   0.344  0.73084
## avg_positive_polarity         -1.614e+03  1.367e+03  -1.181  0.23760
## min_positive_polarity         -1.965e+03  1.144e+03  -1.717  0.08600 .
## max_positive_polarity          3.492e+02  4.311e+02   0.810  0.41796
## avg_negative_polarity         -1.723e+03  1.259e+03  -1.369  0.17092
## min_negative_polarity          1.295e+02  4.589e+02   0.282  0.77781
## max_negative_polarity         -1.828e+02  1.047e+03  -0.175  0.86137
## title_subjectivity            -1.002e+02  2.742e+02  -0.365  0.71479
## title_sentiment_polarity       2.128e+02  2.504e+02   0.850  0.39548
## abs_title_subjectivity         6.445e+02  3.641e+02   1.770  0.07669 .
## abs_title_sentiment_polarity   6.108e+02  3.958e+02   1.544  0.12271
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11500 on 39587 degrees of freedom
## Multiple R-squared:  0.09986,    Adjusted R-squared:  0.09857
## F-statistic: 77.05 on 57 and 39587 DF,  p-value: < 2.2e-16
```

Now everything works fine. The coefficients are interpreted as follows: for a unitary increase of *n_tokens_title* the number of *shares* **increases of 8.986** (on average), kept the other attributes fixed; seems like (imprecisely) writing longer title involves a less number of shares.

Actually, the evaluation of a model begin interpreting two significant metrics, which are *F-statistic* and *p-value*. Without getting deeper, the F-statistic explain whether the null hypothesis which says that there are not any variables associated with the output Y, neither one. The alternative hypothesis say instead that there is at least 1 variable related to the output Y. Thus the F-statistic is a general test applied to the whole model, which indicates that the model has no sense when it is close to 1 because neither one predictor shows significant evidence in explaining the dependent variable Y. Otherwise, if the alternative hypothesis is true the expected value is greater than one, and the model makes sense, as in this case.

This global test is important because when every single predictor is tested the p-value related to this one can be misleading, and is not possible to decide on the global significance of the linear regression model based on the p-values of the feature's coefficients. In fact, each coefficient's p-value comes from a separate statistical test that has a 5% chance of being a false positive result (assuming a significance level of 0.05). The p-value, simply, quantify the evidence of a partial effect of adding X_j to the model. The lowest it is, the higher the evidence of a partial effect is.

Anyways, according to F-statistic and the p-value related to, there is at least one variable that statistically significant affect the predicted value Y hence it is possible to go ahead in assess the accuracy of the model.

Assessing the accuracy of the model

To assess the quality of a regression model is broadly used the R^2 statistic, which is a numerical measure of the quantity of variance explained by the model, against the total variance. The R^2 is computed as following:

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

where $TSS = \sum_{i=1}^{n}(y_i - \overline{y})^2$ is the total sum of squares, or rather the total variance in the response Y while

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y})^2$$

is the residual sum of squares which indicates the amount of variability that is left unexplained.

When R^2 is close to 1 means that a big portion of the variability in the response is explained by the regression while when it is close to 0 indicates that the regression does not explain much of the variability in the response. Let's recall the R^2 statistic for the previous model containing all the variables:

```
summary(model_all)$r.squared # The model without the intercept
```

```
## [1] 0.09986398
```

The R^2 of the model is quite low, which means that the model is able to explain a very low portion of variance. In practice, this means that there are quite surely a best path to dive in to build a model with a better performance. Use all the variables not necessarily means to build the perfect model; as a matter of fact the real problem is to find the variables and more precisely the combination of these that allows to an efficient model, which is the topic of the section below.

Variables Selection: Best Subset Selection

As mentioned, actually, not all the variables of the model are significant in determining the output Y. Having a model which is unnecessary difficult may bring problems in business decisions, such as not easy interpretation, useless and not accurate results, and so on. As said earlier, one of the approach in finding the best model is to build a model with no predictors as first and adding one variable at time until every combination out of all is tried. Thus, having p predictors lead to having, potentially, 2^p models.

The Best Subset Selection is an iterative approach to remove not-significant predictors from a model, solving the problem. The algorithm works just as previously anticipated:

1. Create a model M_0 which contains no predictors;
2. For all the p predictors fit a model that contains as first $p = 1$ then $p = 2$ and so on until a model with $p = k$ predictors is reached;
3. Pick the best among these model according to the smallest RSS or the largest R^2;
4. Select the best out of all using the test error.

The Best Subset Selection algorithm is applied using regsubsets() function in leaps library and the best model is chosen using RSS. It's configuration keep the same structure as the lm() function:

```
library(leaps)

set.seed(1)

model_best <- regsubsets(data=dataset, shares ~ 0 + ., really.big=TRUE,
                         nvmax=56, method="forward")
```

```
## Reordering variables and trying again:
```

The summary() function applied to the model shows the best subset of variables (which of those that are included) for each model of a certain size. The algorithm built 53 models containing the best variables for each p-size model, building the best model for each size. Every model contains the intercept and for example at the rows number 2 of the summary() result, is shown that the best subset selection for a model which contains 2 predictor is made by *kw_avg_avg* and *self_reference_min_shares*.

```
coef(model_best, 2)
```

```
##                 (Intercept)                kw_avg_avg self_reference_min_shares
##               359.85005773                0.93392070                0.02673002
```

Let's extract the model which has the lowest RSS / the highest R^2 and see which are the most significant features:

```
which.min(summary(model_best)$rss) #KEEP
```

```
## [1] 55
```

```r
which.max(summary(model_best)$rsq) # KEEP
```

```
## [1] 55
```

```r
# R2 of the model found with Best Subset Selection:

summary(model_best)$rsq[[55]]
```

```
## [1] 0.02285641
```

The best model according to the algorithm contains all the variables (55th place) + the incercept which has the R^2 equals to 0.023 which is still quite low. For find the contained variables and the coefficients:

```r
coef(model_best, 55)
```

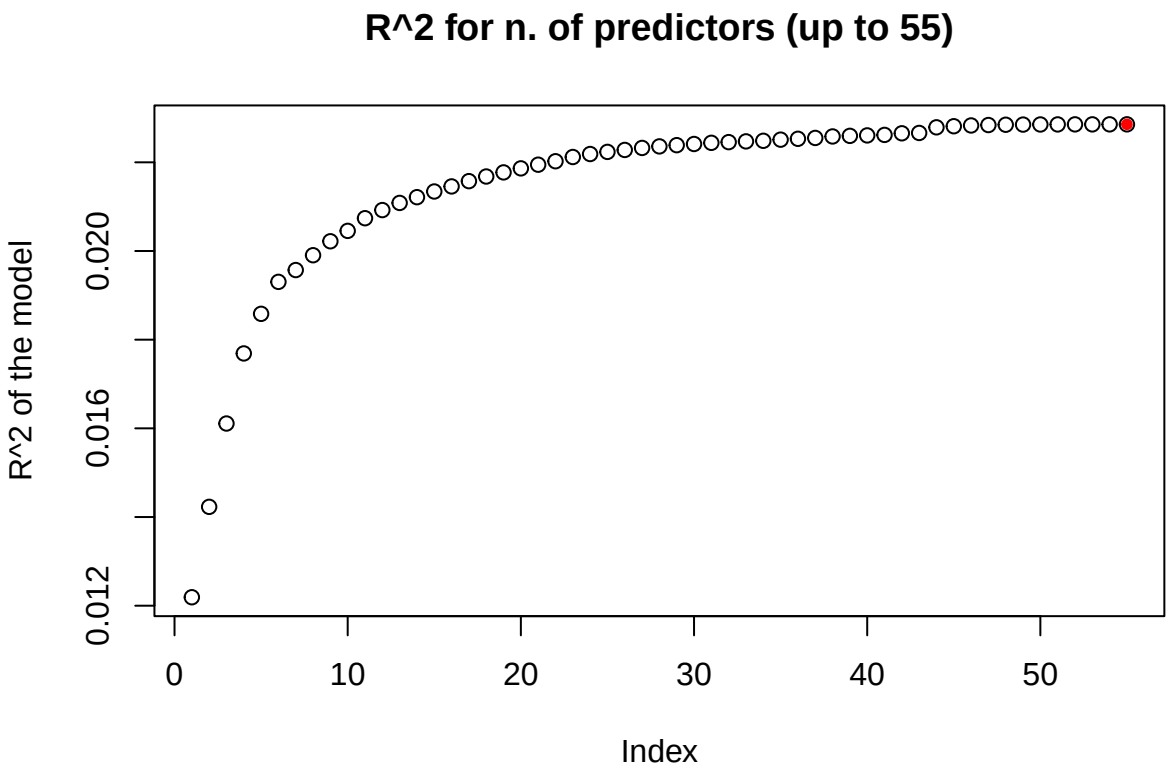
```
##                   (Intercept)              n_tokens_content
##                 -2.430949e+05                  5.948381e-01
##               n_unique_tokens              n_non_stop_words
##                  3.999673e+03                 -1.391364e+03
##      n_non_stop_unique_tokens                     num_hrefs
##                 -1.699983e+03                  2.585022e+01
##                num_self_hrefs                      num_imgs
##                 -5.705801e+01                  1.155350e+01
##                    num_videos          average_token_length
##                  6.597750e+00                 -6.568382e+02
##                  num_keywords     data_channel_is_lifestyle
##                  5.425200e+01                 -1.069499e+03
## data_channel_is_entertainment           data_channel_is_bus
##                 -1.118347e+03                 -7.871372e+02
##        data_channel_is_socmed          data_channel_is_tech
##                 -6.297719e+02                 -5.543828e+02
##         data_channel_is_world                    kw_min_min
##                 -4.526431e+02                  2.193249e+00
##                    kw_max_min                    kw_avg_min
##                  8.819668e-02                 -3.522246e-01
##                    kw_min_max                    kw_max_max
##                 -2.134593e-03                 -4.916449e-04
##                    kw_avg_max                    kw_min_avg
##                 -5.016808e-04                 -3.665304e-01
##                    kw_max_avg                    kw_avg_avg
##                 -2.006596e-01                  1.648174e+00
##     self_reference_min_shares     self_reference_max_shares
##                  2.595513e-02                  5.619652e-03
##    self_reference_avg_sharess             weekday_is_monday
##                 -5.434806e-03                  5.535904e+02
##            weekday_is_tuesday          weekday_is_wednesday
##                  1.946394e+01                  1.853760e+02
##             weekday_is_friday           weekday_is_saturday
##                  4.313201e+01                  6.723368e+02
##                         LDA_00                        LDA_01
##                  2.435867e+05                  2.427525e+05
##                         LDA_02                        LDA_03
```

```
##                 2.423606e+05                 2.431887e+05
##                       LDA_04           global_subjectivity
##                 2.431690e+05                 2.427689e+03
##    global_sentiment_polarity    global_rate_positive_words
##                 6.795300e+02                -1.402131e+04
##   global_rate_negative_words           rate_positive_words
##                 1.426761e+03                 2.416962e+03
##           rate_negative_words         avg_positive_polarity
##                 2.324254e+03                -1.653447e+03
##         min_positive_polarity         max_positive_polarity
##                -2.001325e+03                 3.687859e+02
##         avg_negative_polarity         min_negative_polarity
##                -1.730290e+03                 1.186208e+02
##         max_negative_polarity            title_subjectivity
##                -1.853354e+02                -8.147000e+01
##      title_sentiment_polarity        abs_title_subjectivity
##                 2.048240e+02                 4.957930e+02
##  abs_title_sentiment_polarity             weekday_is_sunday
##                 5.861683e+02                 3.016830e+02
```

The R^2 statistics suffers when adding variables to the model, in the sense that it increases as attributes are added. In fact, adding more variables causes R^2 increasing, even if there is not a really better performance. Moreover, the model containing all the predictors is more difficult to read and can be misleading also; thus, the path to follow in the best subset selection should be another.

Here below the R^2 increase over the number of predictors is graphical represented with his maximum peak:

```
p.max=which.max(summary(model_best)$rsq)

plot(summary(model_best)$rsq, main = "R^2 for n. of predictors (up to 55)",
     ylab="R^2 of the model")+
points(p.max, summary(model_best)$rsq[p.max], pch=20, col="red")
```



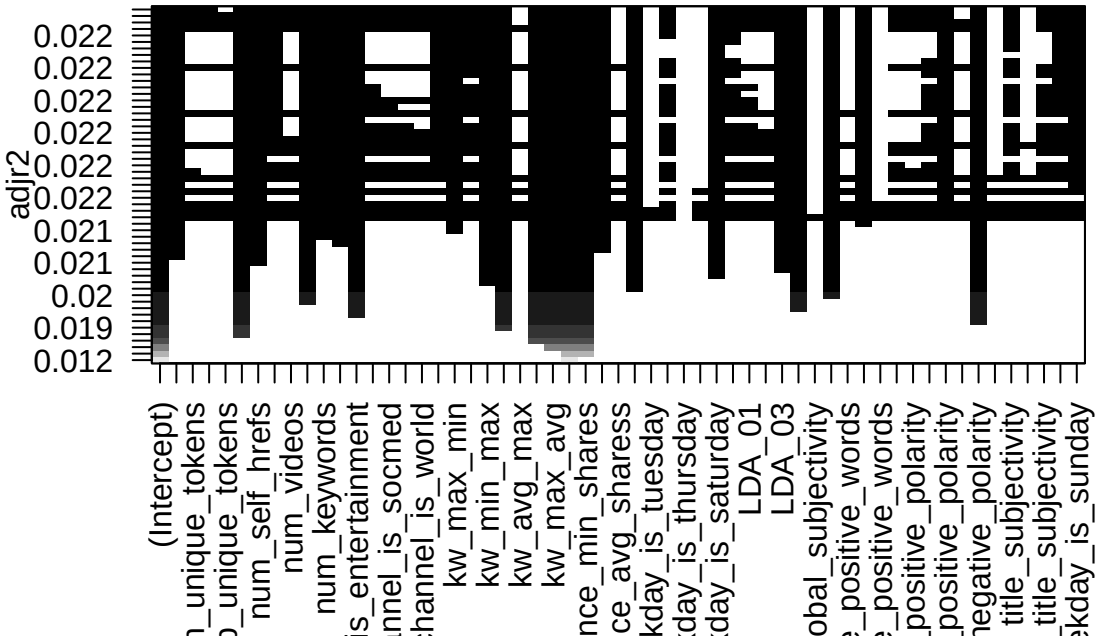

```
## integer(0)
```

```
#QUESTO VA ANCHE UN PO' BENE
```

What can be noticed is that the R^2 increasing does not follow always the same magnitude, although it increases as the number of predictors increase. Actually, there is an important R^2 rising only up to 30 predictors model approximately, at that point the increase continues but not in the important way as before. This could probably be a hint of where the best bias/variance trade off chosen model should be searched around, which is deeper demonstrated in the validation set approach part.

Meanwhile, here there is another representation of all the models depending on which variable is included according to the number of predictors and the *adjusted* R^2:

```
plot(model_best, scale="adjr2")
```



Validation Set approach

Now that one and more models are fitted on the whole dataset, it is time to pick the best based on the test error. Sometimes, there is no option to dispose of a training set and test set, thus estimate the test error of the model is difficult or impossible. This problem can be circumvent using measures like $C_p(AIC)$, BIC, or *adjusted* R^2. Another way to measure the test error is via the validation set approach. This approach involves to:

1. Divide randomly the whole dataset into two parts, a training set and a validation set;
2. Fit the model on the training set;
3. Use the fitted model to predict the responses in the validation set.

The results is assessed using MSE, who provides an estimate of the test error rate. This measure is more reliable than only assessing the best model considering the highest R^2 or the lowest RSS since both of these increase/decrease respectively when an higher number of predictors are included in the model.

To perform the validation set approach, as first the observations will be splitted into a training set and a test set. Here below a random vector called train is created which contains elements equal to TRUE if the corresponding observation is in the training set and FALSE otherwise. Another vector called test has a TRUE if the observation is in the test set and a FALSE otherwise instead:

```
set.seed(1)
```

```
train <- sample(c(TRUE,FALSE), nrow(dataset),replace=TRUE)

test <- (!train)
```

Now the train and the test set are created and the model can be trained and applied on. Go for create a model with the training set performing the best subset selection:

```
set.seed(1)

train.best.mod <- regsubsets(data=dataset[train,], shares~ 0 + .,
                             really.big=TRUE, nvmax=56,
                             method="forward")
```

```
## Reordering variables and trying again:
```

The following step is to build a matrix from data with model.matrix() function which will be fulfilled with the coefficient of the each size best model:

```
test.mat <- model.matrix(data=dataset[test,], shares ~ .)
```

Now run a loop that extract the coefficients from the model built with the train data and multiply them into the appropriate column of the test.mat model matrix to form the predictions and compute the test MSE all toghether:

```
MSE <- rep(NA, 54)

for(i in 1:54){
  coefi <- coef(train.best.mod, id=i)
  pred <- test.mat[,names(coefi)] %*% coefi
  MSE[i] <- mean((dataset$shares[test]-pred)^2)
}
```

Now is time to looking for the best model looking at the MSE which is contained in MSE variable. As said this should yields the best bias/variance trade off based model, because it compares the all the predicted output of the M_k model with the actual result and chooses the best one among them:

```
which.min(MSE)
```

```
## [1] 26
```

Effectively, the best model according to the MSE is the number 26, which contains 26 predictors + intercept:

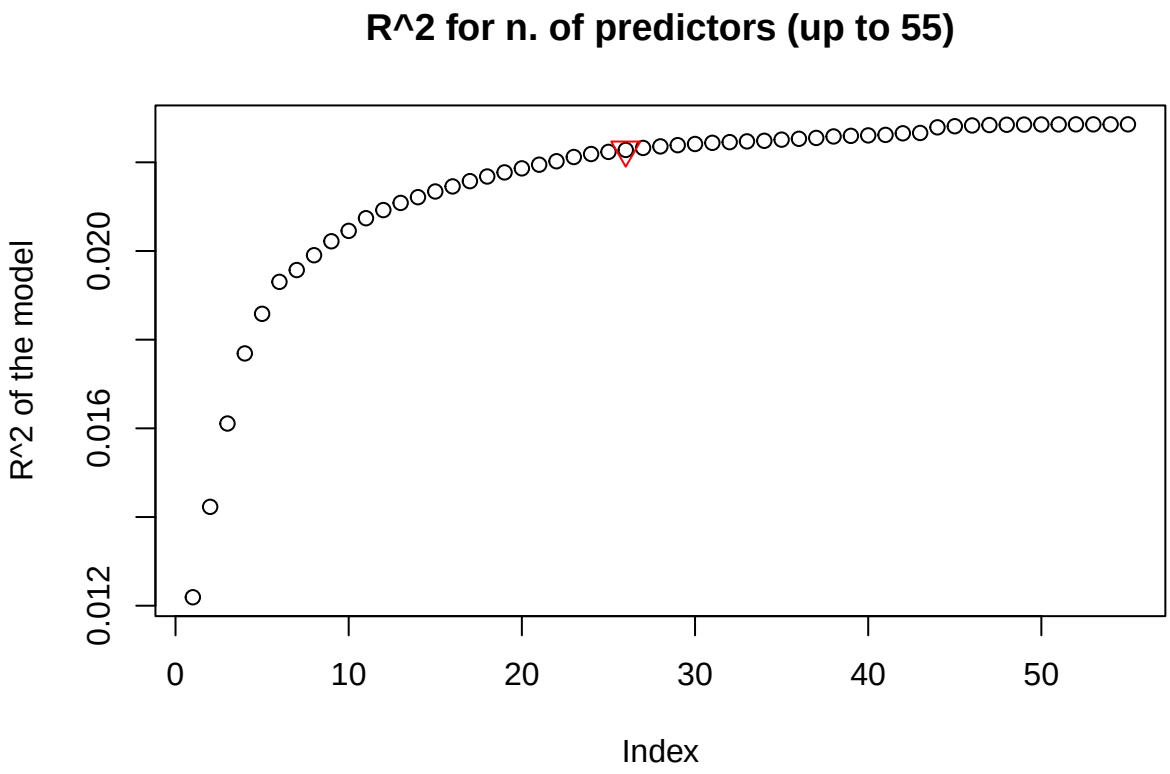
```
coef(train.best.mod, 26)
```

```
##                   (Intercept)                     num_hrefs
##                 -3.498688e+02                  4.635167e+01
##                num_self_hrefs          average_token_length
##                 -5.009434e+01                 -2.587040e+02
##     data_channel_is_lifestyle data_channel_is_entertainment
##                 -3.487563e+02                 -7.816976e+02
```

```
##                   kw_min_min                   kw_max_min
##                 2.266074e+00                 1.662384e-01
##                   kw_avg_min                   kw_min_max
##                -5.584623e-01                -2.479771e-03
##                   kw_max_max                   kw_min_avg
##                -1.204345e-03                -3.581571e-01
##                   kw_max_avg                   kw_avg_avg
##                -1.977055e-01                 1.733937e+00
##    self_reference_min_shares    self_reference_max_shares
##                 5.256650e-02                 1.387664e-02
##   self_reference_avg_sharess            weekday_is_monday
##                -2.579270e-02                 6.107845e+02
##          weekday_is_saturday                       LDA_03
##                 7.840809e+02                 6.596463e+02
##                       LDA_04    global_sentiment_polarity
##                 1.610257e+02                 6.868710e+02
##        max_positive_polarity        avg_negative_polarity
##                 6.070494e+02                -1.991612e+03
##        min_negative_polarity  abs_title_sentiment_polarity
##                -1.934702e+02                 2.245638e+02
##            weekday_is_sunday
##                 2.339547e+01
```

The previous plot, in fact, already suggested that the R^2 mostly increased up to the model which contains 30 variables, or something less. Looking closer at what happens to R^2, this increases a lot up to the model containing 26 variables, after which the increase seen is not more important:

```
plot(summary(model_best)$rsq, main = "R^2 for n. of predictors (up to 55)",
     ylab="R^2 of the model")+
points(26, summary(model_best)$rsq[[26]], pch=25, col="red", cex=1.5)
```

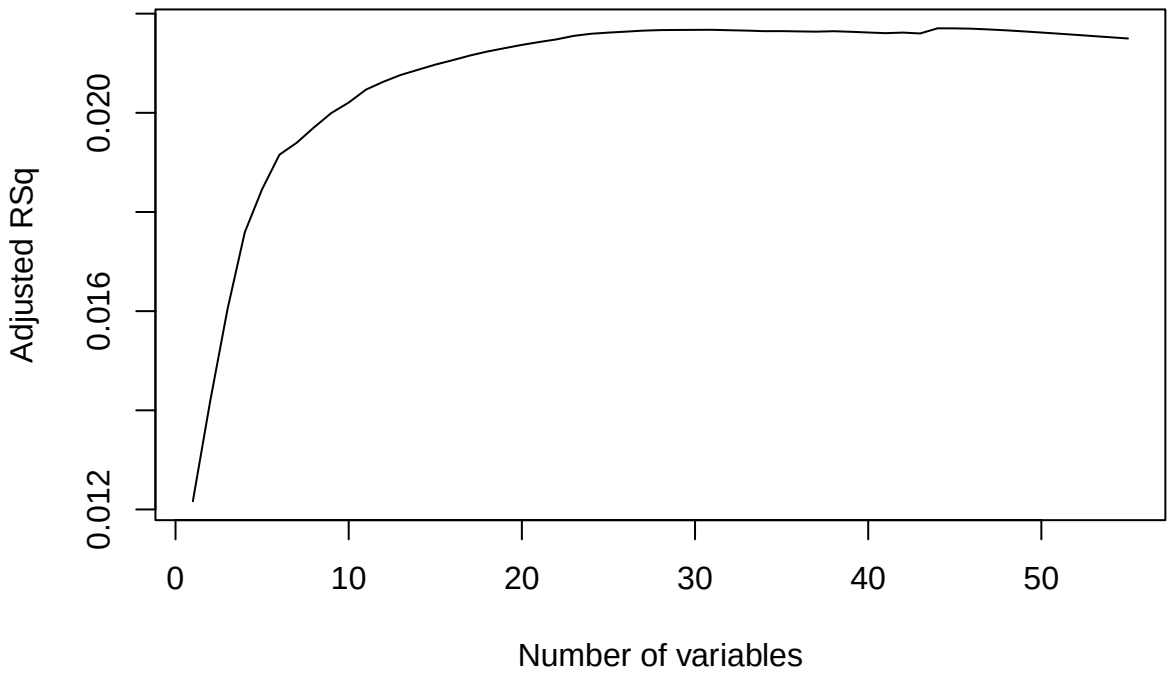



```
## integer(0)
```

Chosing the best model according to indirect measures like C_p, BIC or *adjusted* R^2 instead, can yield different results. Just for the record let's plot these statistics looking for differences. The first metric is the *adjusted* R^2, which is the most similar to the R^2 statistic and that try to take a count of the variables' increasing in the model while it is computed. Differently to the R^2, the *adjusted* R^2coefficient increase slower than the previous one:

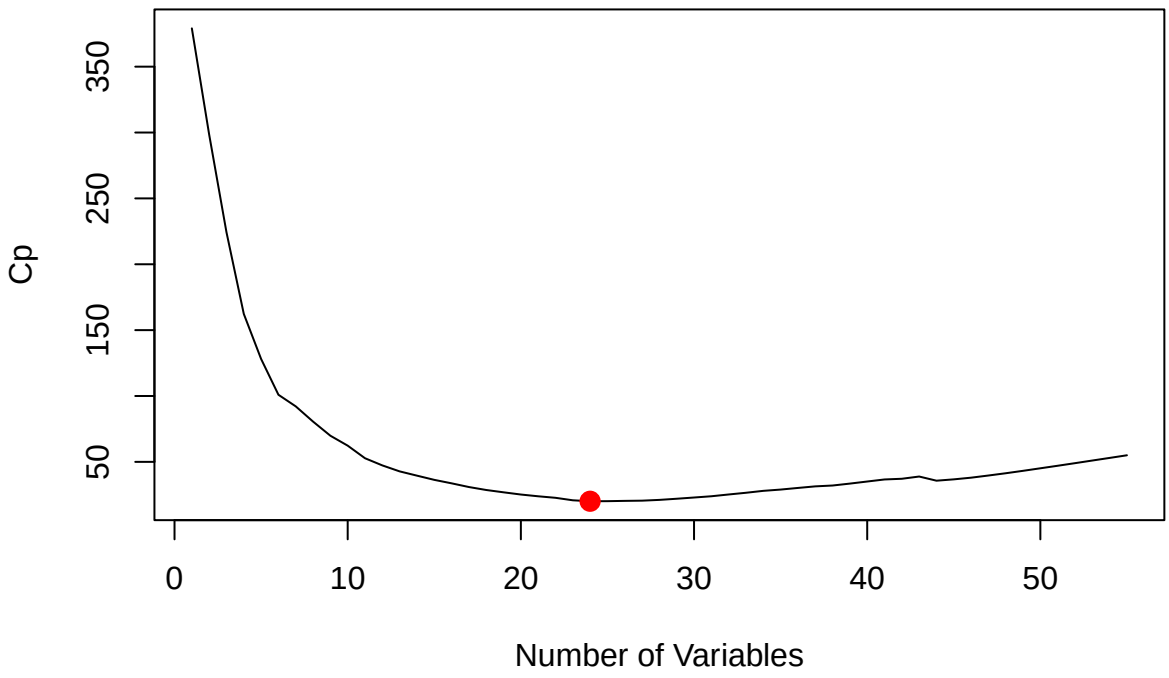
```
plot(summary(model_best)$adjr2, xlab="Number of variables",
     ylab="Adjusted RSq", type="line")
```

```
## Warning in plot.xy(xy, type, ...): il tipo plot 'line' sarà troncato al primo
## carattere
```



The C_p is represented here below instead:

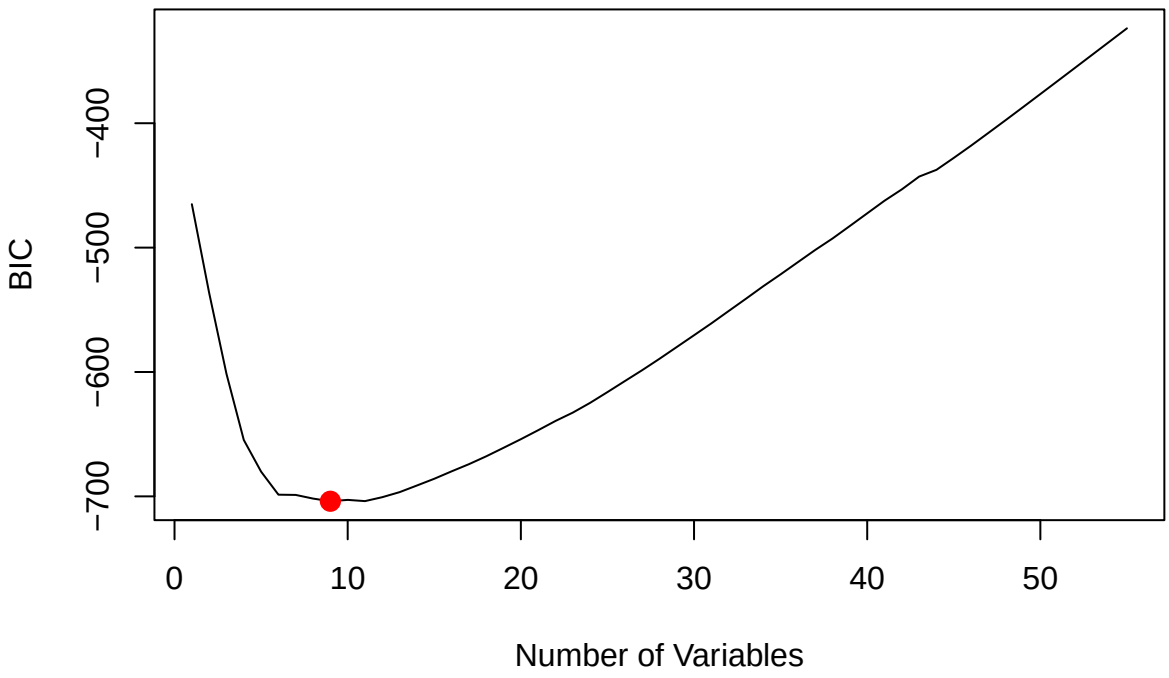
```
pimin <- which.min(summary(model_best)$cp)

plot(summary(model_best)$cp,xlab="Number of Variables", ylab="Cp",
     type="line")+
points(pimin, summary(model_best)$cp[pimin], col ="red", cex =2, pch=20)
```




```
## integer(0)
```

According to C_p , the best model has approximately the same number of variables also. The comparison between this one and the results of the best subset selection says that the model find is quite reliable, since there are two metrics to compare with. Let's see what happens with BIC instead:

```
pimin.bic <- which.min(summary(model_best)$bic)

plot(summary(model_best)$bic, xlab="Number of Variables", ylab="BIC", type="line")+
points(pimin.bic, summary(model_best)$bic[pimin.bic], col="red", cex=2, pch=20)
```




```
## integer(0)
```

While choosing the best model, the lowest BIC should be considered. Unfortunately, this coefficient leads towards a model with 10 variables aproximately instead of 26, which is the number "advised" by the validation set approach and by the C_p estimate. This happens because of the way in which BIC estimate is built, who chooses the best model considering the RSS summed to a penalty that get higher as the number of variables in the model increase. For this reason, places a really heavier penalty when the number of variables are too high and lead the choice towards a model with less variables. Since there is many variables potentially to be considered significant in predicting the output, choose the model with less variables may not be the best practice.

Regularized Regression (LASSO)

Another approach to find the best test-error proof model is using a regularization method, that create a model using all p predictors but constraining the coefficients estimates towards zero. What turns out is a significantly reduction of their variance. As the ordinary least squares, the regularization method want to minimize the RSS , but this time a penalty is added:

$$RSS + \lambda l(B_1, \ \dots \ , B_p)$$

The term $\lambda l(B_1, \ \dots \ , B_p) \geq 0$ is the shrinkage penalty term, who tends to shrink the coefficients and the predictors too towards small values when the minimum value of the above expression is pursued. The tuning parameter λ serves to control the impact of the regression coefficient estimates and it is selected based on cross-validation.

The most commonly known two techniques of regularization are the Ridge Regression and the Lasso Regression, but here only the Lasso Regression is used. What Lasso Regression does is to minimize the quantity:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_i x_{ij})^2 \ + \lambda \sum_{j=1}^{p}|\beta_j| \ = \ RSS + \lambda \sum_{j=1}^{p}|\beta_j|$$

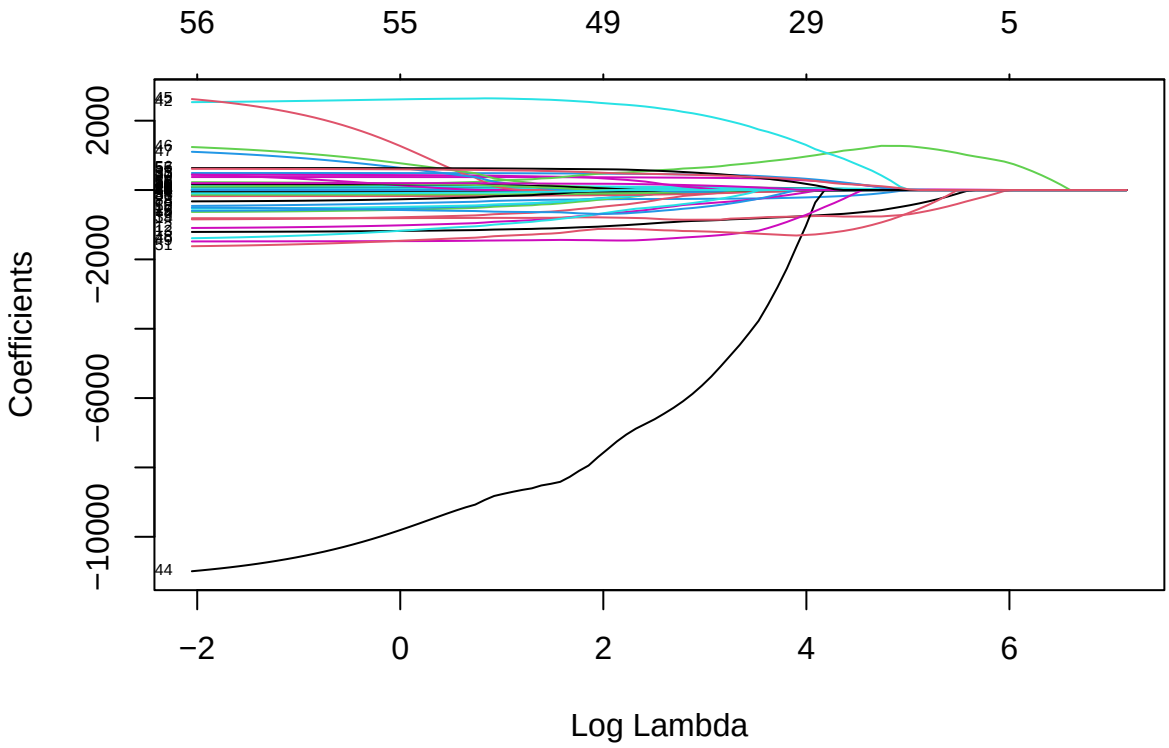
The only difference with Ridge regression is that $|\beta_j|$ is changed with β_j^2 . However, in the case of the Lasso, the penalty force some of the coefficient estimates to be exactly equal to zero when the tuning parameter is large, while with Ridge the coefficients only tends to zero making the models more difficult to interpret than those produced by Lasso regression. Roughly speaking, Lasso regression yields a variable selection much like best subset selection.

Before performing Lasso regression is necessary to prepare the arguments of the function that will be used later. Differently from the lm() and the regsubsets() function, using *glmnet*() function an x matrix and a y vector must be pass in to work. For create x object the previously used model.matrix() function will be instanced:

```
x <- model.matrix(data=dataset,shares ~ .)[, -1]

y <- dataset$shares
```

Now the Lasso regression will be performed as follows:

```
library(glmnet)

set.seed(1)

Lasso <- glmnet(x,y, alpha=1)

plot(Lasso, xvar="lambda", label=TRUE)
```



Where the alpha attribute in the function determines the algorithm which will be used, if Ridge or Lasso. SCRIVERE

```
log(Lasso$lambda[29])
```

```
## [1] 4.552595
```

```
coef(Lasso)[,29] #questi sono i coefficienti quando log(lambda)=4.5526
```

```
##                   (Intercept)                 n_tokens_title
##                  3.604765e+02                  2.189559e+01
##               n_tokens_content                n_unique_tokens
##                  0.000000e+00                  0.000000e+00
##               n_non_stop_words      n_non_stop_unique_tokens
##                  0.000000e+00                  0.000000e+00
##                     num_hrefs                 num_self_hrefs
##                  2.186349e+01                 -7.805196e+00
##                     num_imgs                    num_videos
##                  7.274546e+00                  0.000000e+00
##          average_token_length                  num_keywords
##                 -1.392920e+02                  3.705123e+01
##     data_channel_is_lifestyle data_channel_is_entertainment
##                  0.000000e+00                 -6.375142e+02
##           data_channel_is_bus        data_channel_is_socmed
##                  0.000000e+00                  0.000000e+00
```

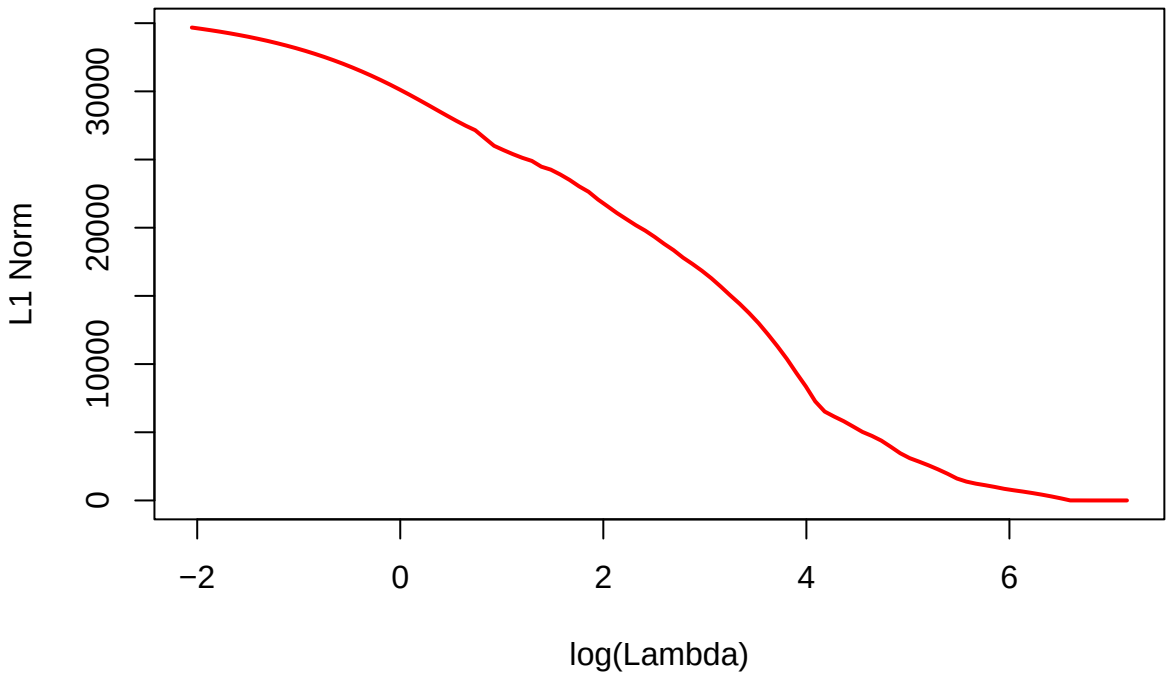
```
##          data_channel_is_tech         data_channel_is_world
##                  0.000000e+00                  0.000000e+00
##                    kw_min_min                    kw_max_min
##                  1.025883e+00                  0.000000e+00
##                    kw_avg_min                    kw_min_max
##                  0.000000e+00                 -8.182141e-04
##                    kw_max_max                    kw_avg_max
##                  0.000000e+00                  0.000000e+00
##                    kw_min_avg                    kw_max_avg
##                  0.000000e+00                 -1.906603e-02
##                    kw_avg_avg     self_reference_min_shares
##                  7.331318e-01                  1.838894e-02
##     self_reference_max_shares    self_reference_avg_sharess
##                  0.000000e+00                  3.739338e-03
##             weekday_is_monday           weekday_is_tuesday
##                  1.420816e+02                  0.000000e+00
##          weekday_is_wednesday          weekday_is_thursday
##                  0.000000e+00                  0.000000e+00
##             weekday_is_friday          weekday_is_saturday
##                  0.000000e+00                  1.535236e+02
##             weekday_is_sunday                        LDA_00
##                  0.000000e+00                  0.000000e+00
##                        LDA_01                        LDA_02
##                  0.000000e+00                 -7.634910e+02
##                        LDA_03                        LDA_04
##                  1.195907e+03                  0.000000e+00
##           global_subjectivity     global_sentiment_polarity
##                  6.754543e+02                  0.000000e+00
##    global_rate_positive_words    global_rate_negative_words
##                  0.000000e+00                  0.000000e+00
##           rate_positive_words           rate_negative_words
##                  0.000000e+00                  0.000000e+00
##         avg_positive_polarity         min_positive_polarity
##                  0.000000e+00                  0.000000e+00
##         max_positive_polarity         avg_negative_polarity
##                  0.000000e+00                 -1.075273e+03
##         min_negative_polarity         max_negative_polarity
##                  0.000000e+00                  0.000000e+00
##            title_subjectivity      title_sentiment_polarity
##                  0.000000e+00                  0.000000e+00
##        abs_title_subjectivity  abs_title_sentiment_polarity
##                  0.000000e+00                  1.452665e+02
```

One more relevant aspect is the L^1 *norm* which is the quantity:

$$||x||_1 = \sum_{j=1}^{n} |x_j|$$

that represent, for a vector $x = (x_1, \ ... \ x_n)$, the distance of (x_1, x_2, x_3, x_4) from the origin $(0, 0)$ of the Cartesian plan for $x = (x_1, x_2, x_3, x_4)$. What Lasso does is to add to the OLS function a penalty term directly related to the L^1 *norm*, which became smaller when a large value of λ is used. This means that as the lambda increases the L^1 *norm* decreases, since the distance among x's decreases because these tends to be 0:

```
L1 <- apply(coef(Lasso)[-1,],2,function(x)sum(abs(x)))

plot(log(Lasso$lambda),L1,type="line",xlab="log(Lambda)",ylab="L1 Norm",
     lwd=2, col="red")
```

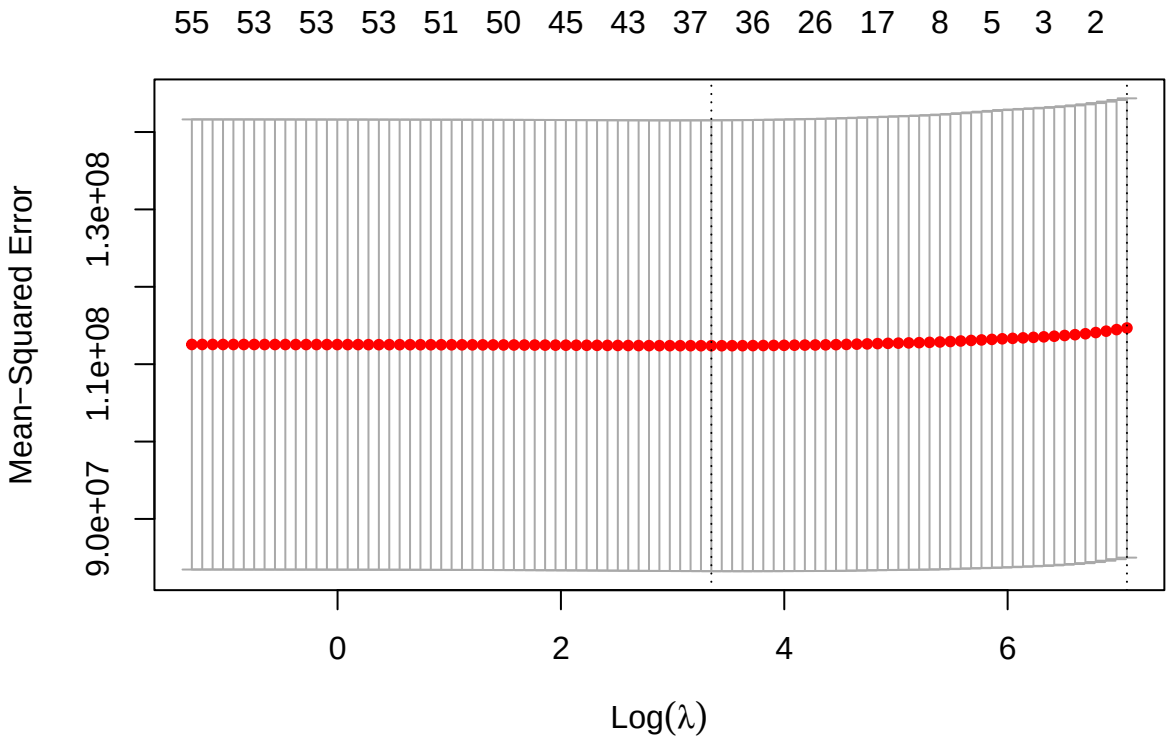


To find the best lambda coefficient, the cross-validation procedure is following made. Before performing the cross-validation, the dataset will be divided in a training set and in a test set:

```
set.seed(1)

train <- sample(1:nrow(x),nrow(x)/2)

test <-(-train)

y.test=y[test]
```

Then the cross-validation can be used to choose the best λ tuning parameter. The function which allows to compute the cross-validation (k-fold) is cv.glmnet(), that produces a plot and return a value for lambda:

```
set.seed(1)

cross.Lasso <- cv.glmnet(x[train,], y[train], alpha=1, nfolds=10)

plot(cross.Lasso)
```



The best lambda (log) is find here below:

```
names(cross.Lasso)
```

```
##  [1] "lambda"     "cvm"        "cvsd"       "cvup"       "cvlo"
##  [6] "nzero"      "call"       "name"       "glmnet.fit" "lambda.min"
## [11] "lambda.1se" "index"
```

```
migliorLambda <- cross.Lasso$lambda.min

log(migliorLambda)
```

```
## [1] 3.346904
```

While the worst (log - within one SE):

```
peggiorLambda <- cross.Lasso$lambda.1se

log(peggiorLambda)
```

```
## [1] 7.068253
```

For finding the test MSE associated with the best λ:

```
Lasso.pred <- predict(cross.Lasso, s=migliorLambda, newx=x[test,])

mean((Lasso.pred-y.test)^2)
```

```
## [1] 181052661
```

This obtained value has to be compared with the MSE of OLS regression, for choosing the best one.

Now it's time to fit the Lasso regression model on the full dataset but using the best value of λ chosen using cross validation and assess the coefficient estimates (only the first 20 coefficient estimates will be shown):

```
Lasso.def <- glmnet(x, y, alpha=1)

# migliorLambda = 23.59031

pwithLambda <- predict(Lasso.def, type="coefficients", s = migliorLambda)

pwithLambda[1:20,]
```

```
##                   (Intercept)                 n_tokens_title
##                  -874.70771457                   69.18244432
##               n_tokens_content                n_unique_tokens
##                     0.12294436                    0.00000000
##               n_non_stop_words      n_non_stop_unique_tokens
##                     0.00000000                    0.00000000
##                      num_hrefs                 num_self_hrefs
##                    26.04206171                  -40.10791500
##                       num_imgs                    num_videos
##                    10.80936197                    2.69222633
##           average_token_length                  num_keywords
##                  -246.30208294                   53.62340748
##     data_channel_is_lifestyle data_channel_is_entertainment
##                  -307.80209595                 -835.46483060
##            data_channel_is_bus        data_channel_is_socmed
##                   -43.21419309                    0.00000000
##           data_channel_is_tech         data_channel_is_world
##                     0.00000000                    0.00000000
##                     kw_min_min                    kw_max_min
##                     2.22915026                    0.01086197
```

Comparison and conclusion

Choosing the best model means choosing the model which shows the lowest MSE among the model trained on the test set. Here below, the OLS regression model, the Best Subset Selection and the Lasso regression model's MSE are compared in order to select the model who can explain better than the others the data.

Here, a model with OLS estimates is built first and the Mean Squared Error is extracted but represented using a logarithmic scale for make him more readable:

```
model_OLS <- predict(cross.Lasso, s=0, newx=x[test,], exact=TRUE, x=x[train,],
                     y=y[train])
```

```r
MSE_OLS <- mean((model_OLS-y.test)^2)

MSE_OLS
```

```
## [1] 308390656
```

```r
log(MSE_OLS)
```

```
## [1] 19.54688
```

The Mean Squared Error of the model find using Best Subset Selection approach:

```r
MSE_Best_Subset <- MSE[[26]]

MSE_Best_Subset
```

```
## [1] 96209223
```

```r
log(MSE_Best_Subset)
```

```
## [1] 18.38204
```

Lastly, the MSE of the Lasso regression computed via cross-validation approach:

```r
MSE_Lasso <- cross.Lasso$cvm[[44]]

MSE_Lasso
```

```
## [1] 112388531
```

```r
log(MSE_Lasso)
```

```
## [1] 18.53747
```

As regard using another way to estimate the coefficients of the model's predictors instead of OLS procedure, it is important to use the most appropriate especially considering the number of observations n and the number of predictors p. Sometimes the number of observations is much larger than the predictors, making the least squares estimation method more accurate compared to the condition in which the number of osbervations n is not much larger than the number of predictors p, where the variance measured on the test observations is higher. Moreover, OLS more difficult yields a coefficients' estimates equal to zero enforcing them to be included in the model, in some sense and making the model more complex. Differently, the regularized regressions shrink the coefficients' estimates towards zero trying to delete them from the model, obtaining a model which is more easy to understand.

In this case, the number of observation is much larger than the number of predictors and the assessment of the models on the test set shows that the best model here is the model find with Best Subset Selection algorithm, which has 26 variables and a $log(MSE) = 18.38204$. Compared to the Lasso, the Best Subset algorithm here is performing better.