Ecole Polytechnique de Louvain
2020-2021
LELEC2870 - Machine Learning

Project report : Predicting shares of articles

Authors: Groupe 38
LEBLANC Stéphane 8197-20-00
LE MEN Thibault 2113-15-00

April 30, 2022

1 Introduction

The purpose of this project is to predict the number of shares of an online article. To do so we use regression models optimized on a set of 19822 observations based on 58 features. Those observed features gather:

- features based on words characteristics (the number of words in the title, in the article, the average word length, and the rate of non-stop words, unique words, and unique non-stop words).
- features based on links characteristics (the number of links, of mashable article links, and minimum, average and maximum number of shares of mashable links).
- features based on media characteristics (the number of images and videos).
- features based on the time of the publication (day of the week, and during the week/week-end).
- features based on keywords characteristics (the number of keywords, the number of shares of worst average and best keywords), and the article category.
- features based on different natural language processing such as the title subjectivity or the rate of positive/negative words among non-neutral words.

We first describe how to apprehend the observation data then how the regression models are implemented and the error estimated. We gather in the conclusion the main result for each model and determine which model is adapted for the aimed prediction.

2 Features selection

Having 58 features at the beginning we want to reduce the dimension of the input space. Before comparing the correlation or mutual information between features we remark the redundancy in at least two features in time category. Indeed, the "is_weekend" feature is a linear combination of the "weekday_ is_saturday" and "weekday_is_sunday" features. So we decide to delete the feature "is_wekeend". Furthermore there are six different features for the article category (lifestyle, technology, ..). We gather all of it in only one feature (with a different value for each category).

Correlation If two feature have a high correlation between them, it's possible to drop one of two because they are redundant. We can mainly remark the biggest correlation rates (bigger than 0.9 in Table 1). We choose 0.9 as threshold because it was a value that was clearly delimiting some highly redundant features from the others but a lower threshold could also be used. Moreover, all these couples of features have as well a high mutual information. Two correlated features could have a low mutual information since correlation is simply a linear link. So correlated features with an high mutual information are clearly redundant and one of them can be dropped. In the table below, "non stop unique words", "Average word length" and "Maximum # shares of worst keywords" have been dropped.

Variance Moreover, it's possible to remove the low variance features. Nevertheless, we did not find features which have the same value for 90% or more for all of its values.

Feature 1	Feature 2	Correlation coeff.	MI coeff.
# unique words	# non stop unique words	0.93689	0.96273
# non stop words	Average word length	0.94345	0.80116
Av. # shares of worst keywords	Max. # shares of worst keywords	0.95304	0.73717

Table 1: Correlation higher than 0.9

Mutual information We compute now the mutual information between features and the output Y and remark some low values below a threshold of 0.009. We took this threshold because there was a clear gap between the mutual information below and above this one. So we decided to drop all the features that have a mutual information with the output below the threshold because they are really not dependant to the output and they don't bring a lot of information.

Feature	MI with output	Feature	MI with output
n tokens title	0.001	polarity abs title sentiment polarity	0
n unique tokens	0.008	weekday is friday	0
n non stop words	0	weekday is saturday	0.004
num videos	0.006	weekday is sunday	0.004
kw avg min	0.005	global subjectivity	0.006
weekday is monday	0.006	global sentiment polarity	0.005
weekday is tuesday	0.005	global rate positive words	0.004
weekday is wednesday	0	global rate negative words	0
weekday is thursday	0.004	rate negative words	0.006
avg positive polarity	0.002	title subjectivity	0.003
min negative polarity	0.005	title sentiment polarity	0.007
max negative polarity	0.005		

Table 2: Mutual information between features and output

3 Regression models description

We were asked to train 4 regressors: linear model, KNN regressor, multi layer perceptron and a random forest regressor that we choose. All these regressors have been tested with the bootstrap 632 error, provided in the mlxtend package, which has the advantage to have no bias and a low variance.

3.1 Linear regression

This model is the simplest one. It estimates a straight line by minimising the mean squared error to compute the weight for each feature. The root mean squared error is similar to the bootstrap error so the model seems not to overfit. Moreover, the model has also been trained with all the features and gets a very high bootstrap error so it seems very sensitive to the input space.

Features	Error
Selected	11762.99
Full	179146228.36

Table 3: Boostrap 632 error for the linear model

3.2 K-Nearest Neighbour regression

KNN, often used for clustering can also be used for regression. The boostrap error has been compared for different value of neighbours. As expected, the error for the full input space is in general greater than the error for the selected input space. Indeed, euclidiean distance is not a very good metric in a high dimensional space. It's very sensitive to noise due to the squared terms and can thus select wrong neighbours. See Table 3 for the minimum error for the two KNN models.

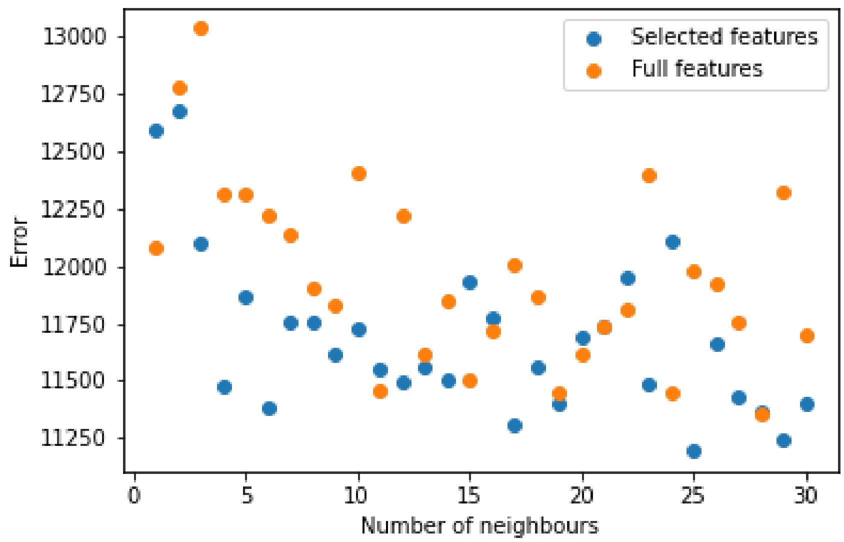

Figure 1: Bootstrap error for KNN model

Features	Min error	Value of K
Selected	11194.52	25
Full	11355.24	28

Table 4: Boostrap 632 error for the KNN model

3.3 Multiple Layer Perceptron regression

The multi layer perceptron, characterized by a lot of hyperparameters such as the number of hidden layers and the number of neurons in each hidden layer, takes the features as inputs to propagate these in the network to output an estimation for the regression problem.

For this model, we did not compare the error with the full input space because of lack of time but the expected results are that a MLP with full features should have a lower error. It should naturally decrease the weight for the most useless features but it should also take a longer time to train.

We trained the model with different number of hidden layers and different numbers of neurons in each hidden layer (same number of neurons for each layer). We made this with two nested loops that is an easy but not performing way. So we stopped the computation for 1 to 13 neurons each trained in 1 to 19 hidden layers. We did not test the number of epochs because a bigger number of epochs should let the model converges and so find a lower error. We thus let the default number of epochs in sklearn which is 200.

Min error	Number of neurons	Number of hidden layers
11252.71	1	4

Table 5: Minimum boostrap 632 error for the multi layer perceptron model

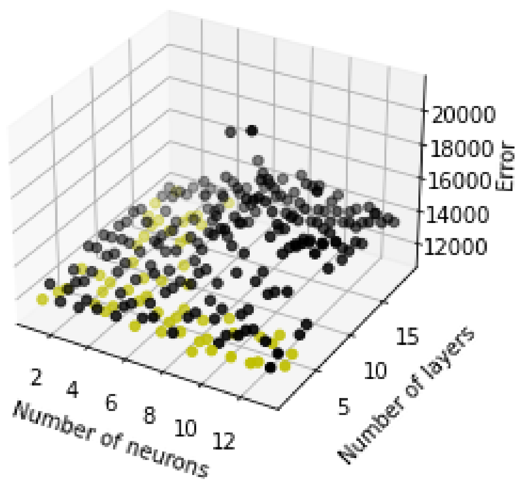

Figure 2: Bootstrap error for multi layer perceptron

The yellow points represent an error below 12000. It seems that the MLP has a lower bootstrap error with a low number of layers. Adding hidden layers should be the way to better approximate the dataset but it should so have a too high polynomial order and overfit.

3.4 Random forest regression

Random forest is an ensemble learning method (ie: a method that builds a aggregation of models to obtain a better accuracy). It is constructed as many decision trees that perform regression or classification by recursively asking questions that have a boolean value. For regression, at the end, the algorithm outputs the average of all trees predictions. Nevertheless, we need uncorrelated trees that wont output the same value. So in order to do that, the bagging method can be used. It refers to random sampling with replacement such as trees have a different sample of the training set from the each others.

For this model, different numbers of trees have been tested (1 to 100). We didn't compare the model with the full input space because of lack of time. From 20 trees, the error doesn't seem to have a decreasing tendency with a minimum value for 59 trees. Moreover, without the model selection, this model was overfitting the dataset. Indeed, for different numbers of trees, the average root mean squared error was near 4500 that is very far from other models and from the bootstrap error.

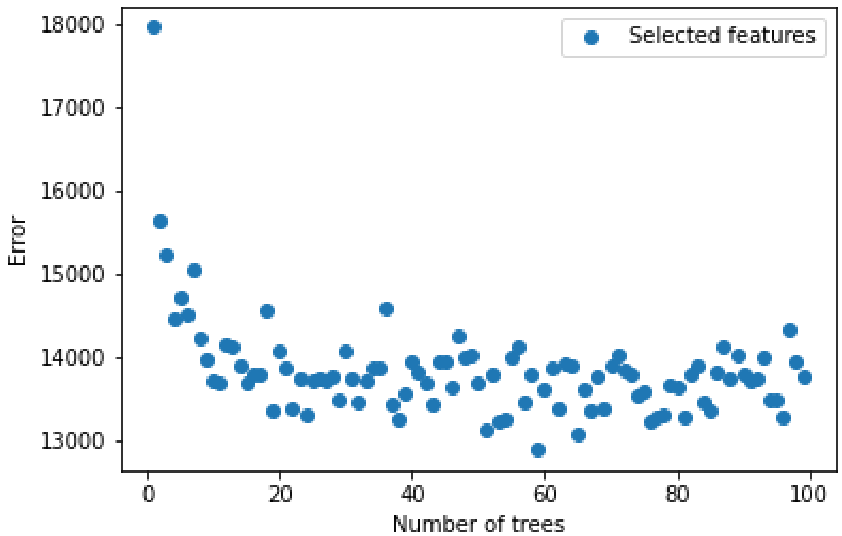

Figure 3: Bootstrap error for random forest model

error	Number of trees
12902.05	59

Table 6: Minimum boostrap 632 error for the random forest model

4 Conclusion

Between the four model, KNN was giving the lowest bootstrap error so it will be with this model that we are going to use to make predictions. Here is a recap of errors for the four models.

Model	Error
Linear model	11762.99
KNN	11194.52
Multi layer perceptron	11252.71
Random forest	12902.05

Table 7: Boostrap 632 errors for the selected models

A new csv file named Y2 is created with the predicted values of the KNN model. The model gives an accuracy of 0.439 for the predicted values. This score doesn't seem to be especially very good for this dataset because by just testing with an MLP with the default parameters of Sklearn, we obtain an accuracy near 0.5. Nevertheless, the KNN model gets the lowest bootstrap error and so should be more robust on any data of the real world compared to the other models. As he bootstrap 632 error tries to to estimate the error on all possible data, the score on Y2 should be similar to the score on Y1.