Université Catholique de Louvain

LELEC2870

Machine learning : regression, deep networks and dimensionality reduction

Predicting shares of articles

François Gouverneur 3736-1700
Romain Graux 2868-1700

Teacher:
Michel Verleysen

December 20, 2020

Info: All scores obtained in this report can be found on our GitHub : `https://github.com/RomainGrx/LELEC2870-Project`.

Introduction

Machine Learning methods can be used to solve many practical problems in a wide range of applications such as weather forecast, customer clustering, medical diagnostics, spam blocking, financial time series prediction or signal denoising,...

In this project we will apply different machine learning methods in order to predict the number of shares of an online articles based on its components. The components of an article are the number of words in the title, the day of the week it was published, the rate of unique words and a whole bunch of others.

First we select our features based on different criteria. Then we implement 4 methods : `Linear Regression`, `K-Nearest Neighbour (KNN)`, `MLP` and our chosen method, `Ensemble Learning`. After that we train these models and validate them. Finally we select the best model and try to predict $Y2$ on the data $X2$ given to us.

1 Features Processing

Before starting the project, it is important to have a general overview of the data we are working with. For some of the figures in this report we will use numbers to represent each feature, they are visible with their *mean* and *standard deviation* on the Table below.

For all features, we applied a standard scaler to standardize all data respect to $\mathcal{N}(0,1)$. We also exclude some outliers to focus on the general pattern and be biased by the outliers. To do so, we could use a N standard deviation remover, that will keep the data within the N range of standard deviation but with our sparse data, the model won't be able to well predict the *viral hits.*

So to avoid such behaviour, we decided to use an isolation forest that detect the anomaly (outliers) based on the training data. This ensure that *viral hits* are kept if there are obviously enough of them.

		mean	std			mean	std
data_channel_is_bus	(1)	0.159	0.365	global_subjectivity	(2)	0.443	0.117
data_channel_is_entertainment	(3)	0.181	0.385	avg_positive_polarity	(4)	0.353	0.104
max_negative_polarity	(5)	-0.108	0.098	data_channel_is_tech	(6)	0.187	0.390
n_non_stop_unique_tokens	(7)	0.672	0.154	data_channel_is_world	(8)	0.205	0.403
data_channel_is_lifestyle	(9)	0.052	0.222	abs_title_subjectivity	(10)	0.340	0.189
LDA_00	(11)	0.184	0.262	rate_positive_words	(12)	0.682	0.189
data_channel_is_socmed	(13)	0.059	0.235	abs_title_sentiment_polarity	(14)	0.158	0.225
LDA_02	(15)	0.211	0.279	max_positive_polarity	(16)	0.758	0.248
LDA_01	(17)	0.142	0.220	LDA_04	(18)	0.236	0.292
weekday_is_monday	(19)	0.169	0.374	global_sentiment_polarity	(20)	0.119	0.097
is_weekend	(21)	0.133	0.339	title_subjectivity	(22)	0.286	0.325
average_token_length	(23)	4.545	0.845	n_tokens_title	(24)	1.042e1	2.118
weekday_is_thursday	(25)	0.184	0.388	weekday_is_saturday	(26)	0.062	0.241
weekday_is_friday	(27)	0.142	0.349	num_self_hrefs	(28)	3.308	3.993
weekday_is_tuesday	(29)	0.185	0.388	min_negative_polarity	(30)	-0.524	0.289
num_imgs	(31)	4.609	8.440	num_hrefs	(32)	1.088e1	1.113e1
title_sentiment_polarity	(33)	0.075	0.265	num_keywords	(34)	7.235	1.919
num_videos	(35)	1.275	4.104	weekday_is_wednesday	(36)	0.187	0.390
kw_min_min	(37)	2.610e1	6.965e1	kw_avg_avg	(38)	3.139e3	1.328e3
kw_avg_min	(39)	3.129e2	6.769e2	kw_min_avg	(40)	1.115e3	1.137e3
n_tokens_content	(41)	5.492e2	4.769e2	kw_max_avg	(42)	5.621e3	5.905e3
kw_max_min	(43)	1.153e3	3.995e3	self_reference_avg_shares	(44)	6.429e3	2.383e4
self_reference_min_shares	(45)	3.953e4	1.860e4	self_reference_max_shares	(46)	1.051e4	4.225e4
kw_min_max	(47)	1.358e4	5.777e4	kw_avg_max	(48)	2.600e5	1.349e4
kw_max_max	(49)	7.538e5	2.120e5	rate_negative_words	(50)	0.288	0.155
min_positive_polarity	(51)	0.095	0.069	avg_negative_polarity	(52)	-0.261	0.129
global_rate_negative_words	(53)	0.017	0.011	global_rate_positive_words	(54)	0.040	0.017
n_non_stop_words	(55)	0.970	0.170	n_unique_tokens	(56)	0.530	0.137
weekday_is_sunday	(57)	0.071	0.256	LDA_03	(58)	0.227	0.296

Table 1: Mean & Standard deviation for each features (58 in total) before standardization

2 Feature Selection

Feature selection allows us to select most useful features from inputs. In a first place we classify features in terms of relevance between inputs and output. Then we choose which features to select minimizing the redundancy.

We tried to get the best features from a Sequential Forward/Backward Selection but the problem with this wrapper is that it really depends on the model used to predict the output to select the features. So we just tried a SFS with a ridge for the experiment but we did not put it in this report because it is not relevant in our case.

2.1 Mutual Information (MI)

In order to classify features in terms of relevance we could have use a correlation matrix but it has the disadvantage of only measuring the linear relation. Meaning that a zero-correlation is not enough to conclude about a pair input/output. That's why we prefer to use a Mutual Information method to select features.

In a first place we have classify the features according to their mutual information with the output (# shares) as we can see on Figure 2. This figure shows us the *relevancy* of our features but we cannot just select the best ones to train our model because of the *redundancy*. That's why we will use a Mutual Information matrix in the next point.

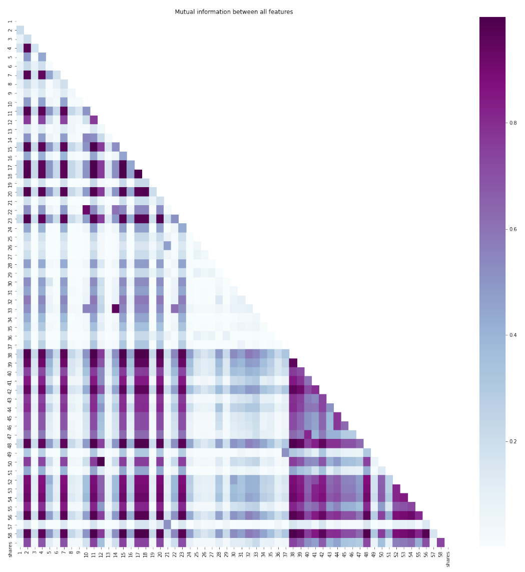

Figure 1: Mutual Information matrice

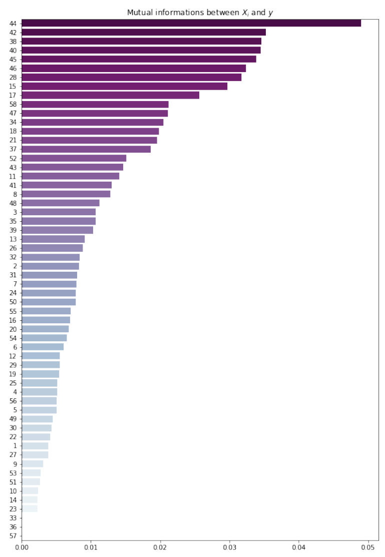

Figure 2: Mutual information between each feature (X_i) and the number of shares (Output Y)

2.2 Minimum Redundancy Maximum Relevancy (MRMR)

Based on the Mutual Information matrix [Fig. 1] above, we can choose which feature to drop from the relevancy list. Depending on the number of features we want to work with, these are the ones we select for our models [Tab. 2].

# features	*Selected features*
5	kw_avg_avg (38) - max_positive_polarity (16) - weekday_is_wednesday (36) max_negative_polarity (5) - abs_title_subjectivity (10)
10	+ data_channel_is_socmed (13) - self_reference_min_shares (45) rate_negative_words (50) - data_channel_is_lifestyle (9) - weekday_is_saturday (26)
15	+ LDA_01 (17) - num_imgs (31) - weekday_is_sunday (57) - num_videos (35) - kw_min_max (47)
20	+ kw_min_min (37) - n_tokens_title (24) - weekday_is_friday (27) data_channel_is_bus (1) - min_positive_polarity (51)

Table 2

3 Linear Regression

As part as a baseline, we first try a simple linear regression. The linear regression tries to estimate the function $f(x)$ with an estimator set as $\hat{f(x)} = w^T x$. The matrix w is computed by minimizing the mean square error: $\min_w ||y - w^T x||_2^2$. With this model we obtain these scores on both the training and the validation set.

Model	*5 features*	*10 features*	*15 features*	*20 features*	*All features*
Training set	0.467	0.470	0.476	0.476	0.487
Validation set	0.476	0.480	0.484	0.485	0.485

Table 3

We observe that the best estimation is without any surprise the one when all features are selected, both in training and validation set. The less features we have, the worst our error. However, we obtain almost the same score for both the training and the validation set. It tells us that the model tends to underfit (it is obvious regarding the linear relation between input/output), but the model is at its best.

4 K-Nearest Neighbour (KNN)[1]

In this model we use features similarity to predict values of any new data points. This means that the new point receives a value based on how closely it resembles the points in the training set.

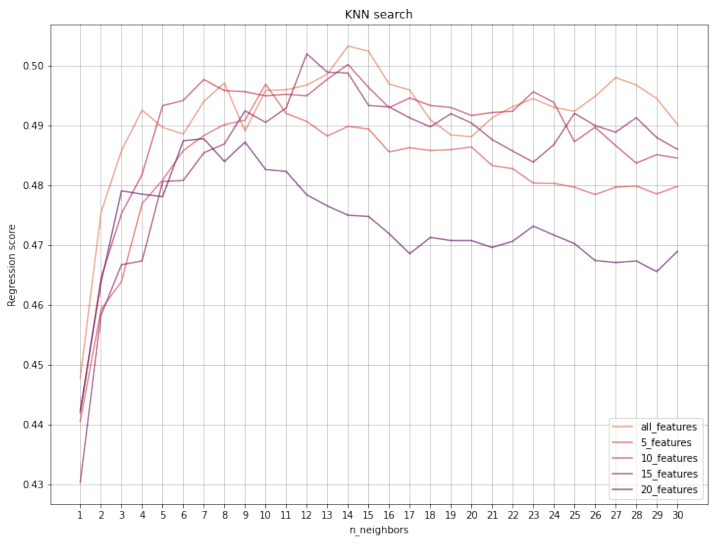

Figure 3: KNN regression score regarding the number of neighbors with Eucledian distance (= Minkowski with $p = 2$)

[1]https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/

Depending on the number of neighbors we select, the regression score varies. It rises quickly with the number of neighbors ($= k$) when it is low before slowly decreasing after $k > 10$ (approximately). When k is small, the model overfits on the training data, which leads to a low regression score on the validation set. But when k is too large, the model performs also poorly. We see on the above figure [Fig. 3] that regarding the number of features selected, the optimal value for k is not the same but still more or less similar ($7 \leq k_i \leq 14$).

The number of selected features is not the only parameter influencing the optimal value for k. The method to calculate the distance between 2 points also influence the optimal k. In the table below [Tab. 4] we try with 4 different methods to compute the distance between 2 points ($x, y \in \mathbb{R}^n$) :

- Minkowski : $(\sum_i^n |x_i - y_i|^p)^{1/p}$ with param. p
- Manhattan : $\sum_i^n |x_i - y_i|$
- Eucledian : $\sqrt{\sum_i^n (x_i - y_i)^2}$
- Chebyshev : $\max_{i \in \{1,..,n\}} |x_i - y_i|$

Model	*5 features*		*10 features*		*15 features*		*20 features*		*All features*	
	k	score	k	score	k	score	k	score	k	score
Minkowski ($p = 5$)	11	0.4946	12	0.4992	9	0.4942	8	0.4854	15	0.4993
Eucledian	10	0.4969	14	0.5002	12	0.5020	7	0.4878	14	0.5033
Manhattan	10	0.4931	17	0.5016	14	0.5035	7	0.4816	14	0.5099
Chebyshev	8	0.4932	6	0.4953	9	0.4892	6	0.4777	16	0.4964

Table 4: Optimal number of neighbors (k) and corresponding regression score regarding distance measurement & number of selected features

5 Multilayer Perceptron (MLP)

For this model, we decided to stack 3 dense layers with ReLU as activation function in the hidden layers and of course a linear activation for the output layer. We set N neurons in the first layer, then $N//2$ and finally $N//4$ in the last layer ($//$ being the integer division).

For the training, we added a dropout of 10% for each layer to avoid overfitting as much as possible and we used a batch size of 256 on 50 epochs. We used an Adam optimizer with a learning rate of $1e^{-3}$ and mse as the loss function.

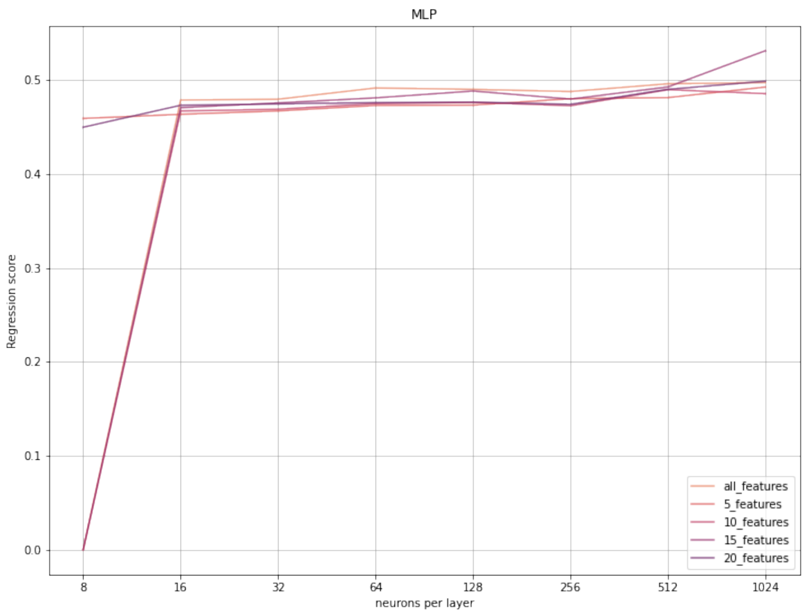

Figure 4: The regression score regarding the number of neurons

On the figure above [Fig. 4] and the table below [Tab. 5], we can see that the more the neurons we have, the best the validation score. But we have to be careful that the more neurons we have, the more variables will allow us to overfit on the training set. This is why the choice of the number of epochs will be more important with more neurons (or we use an early stop based on another validation set). Furthermore, we obtain a minor change in the score between the number of features. This is normal regarding the structure of the MLP; it is able itself to set its weights to 0 if it considers a feature not relevant (the backpropagation will set to 0 the feature that does not contribute to the output prediction).

Neurons/layer	*5 features*	*10 features*	*15 features*	*20 features*	*All features*
8	0.4590	0.0	0.0	0.4496	0.0
16	0.4634	0.4669	0.4705	0.4730	0.4786
32	0.4669	0.4688	0.4754	0.4745	0.4795
64	0.4728	0.4747	0.4809	0.4758	0.4914
128	0.4732	0.4762	0.4881	0.4764	0.4890
256	0.4800	0.4725	0.4796	0.4740	0.4876
512	0.4812	0.4896	0.4926	0.4900	0.4959
1024	0.4924	0.4854	0.5311	0.4986	0.4972

Table 5: Score regarding the number of neurons/layer and the number of selected features

6 Ensemble Learning

For the custom model, we decided to use an ensemble learning model based on the AdaBoost method itself composed of 50 decision tree regressor. It will allow us to use several weak regressor to obtain a better regressor. The main problem with the AdaBoost is that it tends to overfit because with each iteration, it will contribute to reinforce "borderline" cases. The more depth we have in our decision trees, the more the model overfits.

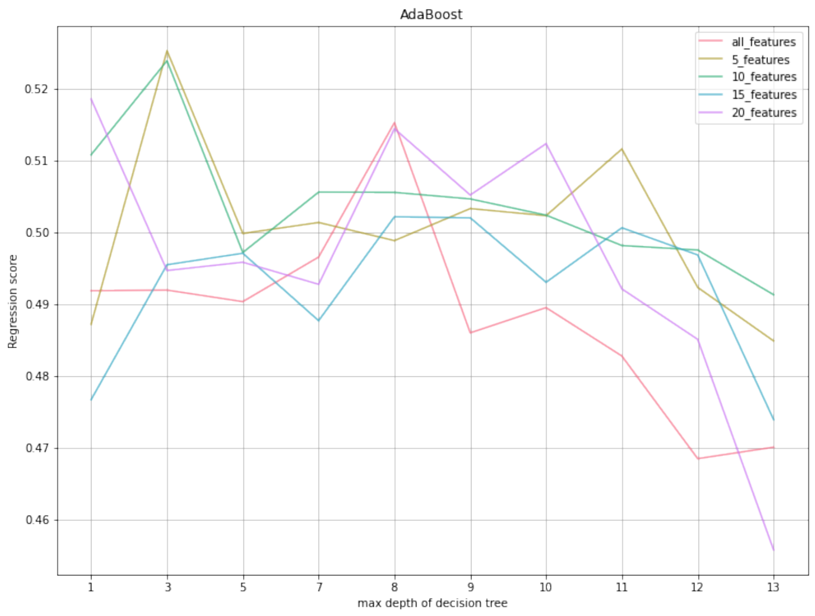

Figure 5: Regression score regarding the max depth in the decision trees

	5 features	*10 features*	*15 features*	*20 features*	*All features*
Training set	0.5209	0.5288	0.4993	0.4987	0.4953
Validation set	0.5253	0.5239	0.4955	0.4947	0.4920

Table 6: Regression score for 3 max depth in the decision trees

We can observe on the figure and the table above [Fig. 5, Tab. 6] that we obtain a tradeoff between overfit and underfit because we obtain almost the same scores for training and validation set. Furthermore, the best validation score is obtain with 5 features. We can conclude that the best the features are chosen, the best the regression score.

7 Comparison

Model		*5 features*	*10 features*	*15 features*	*20 features*	*All features*
Linear Regression		0.476	0.480	0.484	0.485	0.485
KNN	Minkowski ($p = 5$)	0.494	0.499	0.494	0.485	0.499
	Eucledian	0.497	0.500	0.502	0.488	0.503
	Manhattan	0.493	0.501	0.503	0.482	**0.510**
	Chebyshev	0.493	0.495	0.489	0.478	0.496
MLP		0.492	0.485	**0.531**	**0.499**	0.497
Ensemble Learning		**0.525**	**0.524**	0.495	0.495	0.492

Table 7: Summarize of (best) regression score for every model regarding the number of features selected

To conclude, we have seen that MLP and Ensemble Learning have the best scores. When we have a few features (≤ 10), the Ensemble Learning method is better than the MLP one, while when the number of features is ≥ 15, it is the opposite. But as we said, the mlp is able to cancel the influence of features so it maybe uses less features than we the one given as input.

In order to predict the value for $Y2$ we might be tempted to choose the MLP model with 15 features because it is the one with the best score of all. However, we prefer to choose the **Ensemble Learning** model with **5 features** and **3 max depth** because this model is more constant while the MLP one has only one score above 0.5 and it is more favorable to randomness on the training process.