Python for data Analysis

Saber Z.
Lalaina R.
Rizlane T.

the dataset

Online News Popularity Data Set

- 61 variables (58 predictive attributes, 2 non-predictive)

The dataset's description specifically considers the first two features url and timedelta as non-predictive. We remove them from the dataset. Nevertheless, we'll store them elsewhere for a potential alternative model to compare.

- 39644 observations

We rename the columns because they all have a space in their name.

- Target

Since the target shares is a quantitative variable, we're in a regression problem.

the dataset

missing and misc. values

It seems that there are neither *NA* nor duplicated values (How lucky we are !).

```
data_url.nunique() == len(data_url)
```

```
True
```

```
any(data.isna().sum() != 0)
```

```
False
```

However, when transforming some features, the dataset revealed some values that could be considered *NA* in spirit. For example, let's examine the *rate_positive_words* and *rate_negative_words* case.

the dataset

missing and misc. values

We observed that the sum of the two columns is always equal to 1. This is no coincidence as the following command showcases the following relationship :

$$y = 1 - x$$

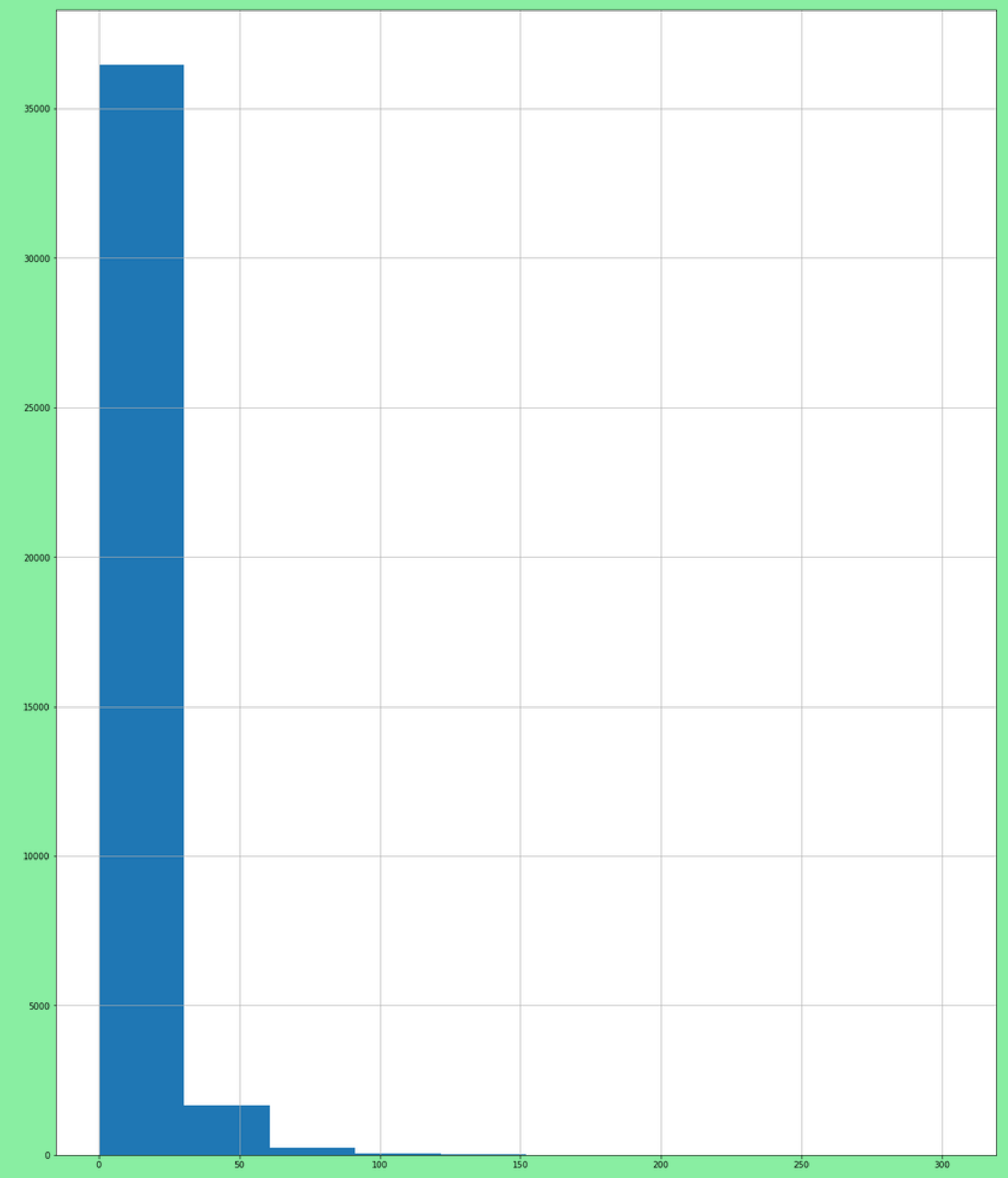
We find some rounding errors due to a mistake on either Python's or the source's end. Either way, we supposed that Python returns a correct result with 1% error and we decided to remove all the lines that have (0, 0) values for these two features. Our choice is consolidated by the fact these observations also have 0 values for the other features present in the dataset.

```
data[['rate_positive_words',
      'rate_negative_words']].head(10)
```

	rate_positive_words	rate_negative_words
0	0.769231	0.230769
1	0.733333	0.266667
2	0.857143	0.142857
3	0.666667	0.333333
4	0.860215	0.139785
5	0.523810	0.476190
6	0.827957	0.172043
7	0.846939	0.153061
8	0.600000	0.400000
9	0.562500	0.437500

the dataset

Histogram of num_hrefs pretransform



Dataset's outliers

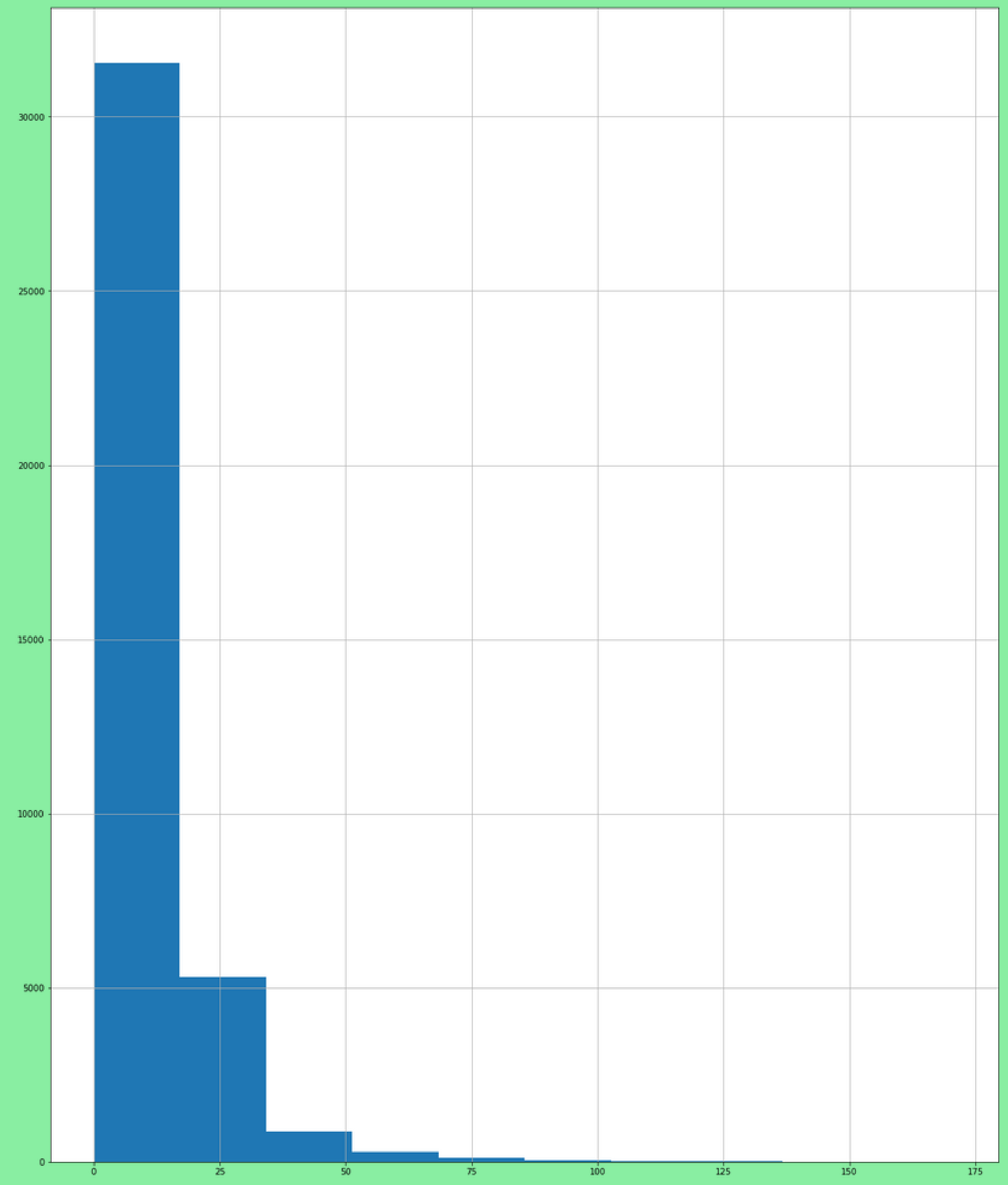
We decided to consider than an outlier is a value that have a 1 in 10^50 chance to exceed the mean, that is to say, the D values such as :

$$|D - \mu_{feature}| > 15\sigma_{feature}$$

The outlier criteria is intrinsically related to its standard deviation. Thus, we won't consider the features who are already in the [–1,1] interval but only those whose histograms are compressed due to surprising extreme values. After that, we'll check that the number of outliers in shares hasn't significatively change, this would mean that to confirm our suspicisions about the outliers' nature.

the dataset

Histogram of num_hrefs posttransform



Dataset's outliers

Undeniably, we have an easier time to estimate the different distributions, for example *num_hrefs* appears to be following an exponential distribution whereas in the precedent histogram it looked like one range of values dominated the rest outside that range.

the dataset

Aftermath

```
1 data.shares.describe()
```

```
count     38237.000000
mean       3321.762534
std       11050.341900
min           1.000000
25%         944.000000
50%        1400.000000
75%        2700.000000
max      843300.000000
Name: shares, dtype: float64
```

```
1 pd.read_csv("OnlineNewsPopularity.csv")[' shares'].describe()
```

```
count     39644.000000
mean       3395.380184
std       11626.950749
min           1.000000
25%         946.000000
50%        1400.000000
75%        2800.000000
max      843300.000000
Name:  shares, dtype: float64
```

After these changes and the drop of missing values, we can see that the original dataset doesn't really differ to the transformed one. We've lost 3% of the dataset and 5% of standard deviation which is most likely the result of noise as we've explained.

dataset preprocessing

Generic preprocessing methods

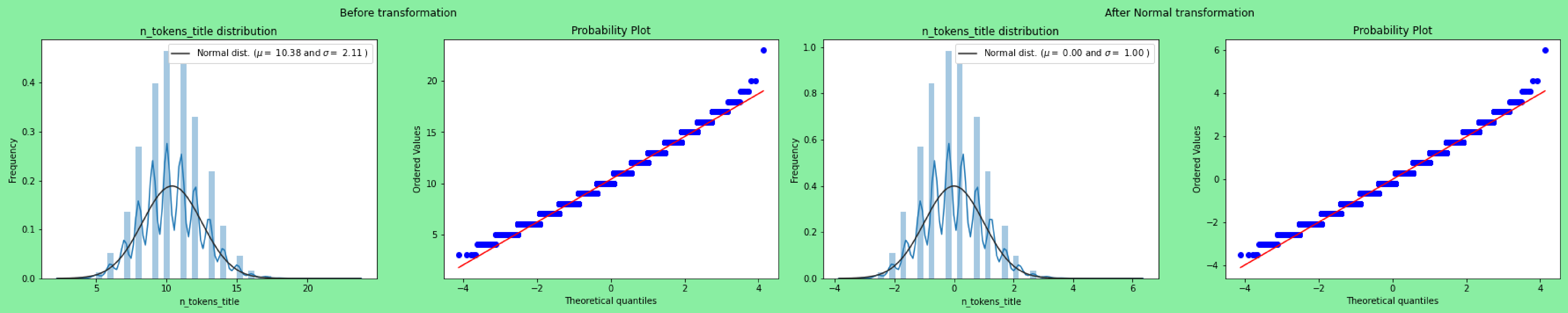
In order to make the best out of the preprocessing, we write the different methods that enable us to plot histograms with density plots, correlation plots, probability-probability plots as well as to normalize data following various probability laws : binomial, exponential and even the beta distribution. The target is expressed in the thousands. However, most of the features' values are contained in the [-1, 1] interval. To avoid overfitting and underfitting certain variables, we'll transform all features whose values surpass the previously mentionned interval. As for the target variable, scaling is only helpful if its values could cause a memory overflow, which was not our case.

dataset preprocessing

Binomial distribution

It seems fair to assume the probability distribution of *n_tokens_title* follows a binomial law, which converges to a normal law. Let's standardize it to follow $N(0,1)$ law in order to restrain its values as much as possible to the $[-1, 1]$ interval.

$$Z = \frac{X - \mu}{\sigma} \hookrightarrow \mathcal{N}(0,1)$$

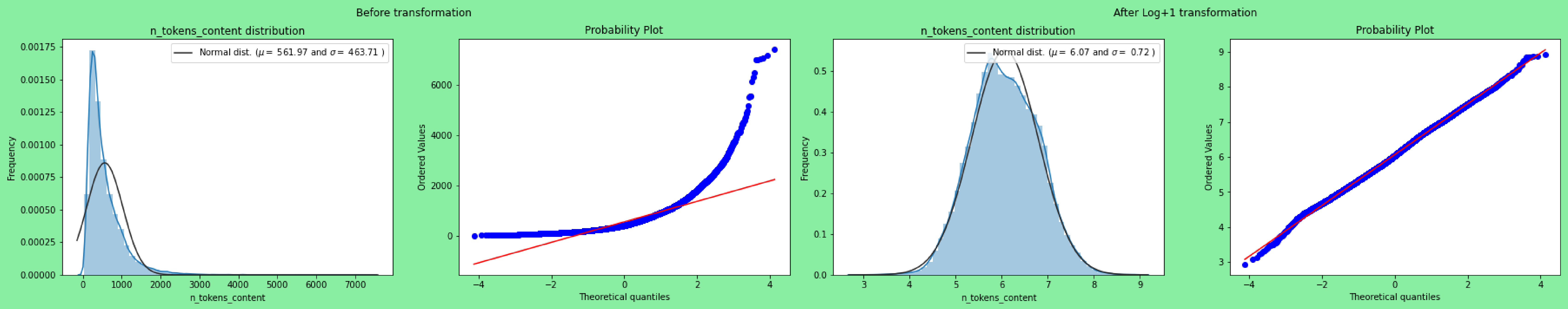


dataset preprocessing

Exponential distributions

n_tokens_content follows an exponential law. We standardize it by applying the logarithm function. Nevertheless, to avoid the log(0) error. We'll use variants of the log transformation.

$$Z = \begin{cases} log(X+2) & \text{if } -2 < min(X) \leq -1 \\ log(X+1) & \text{if } -1 < min(X) \leq 0 \\ log(X) & \text{otherwise} \end{cases}$$



dataset preprocessing

Exponential distributions

Other features following an exponential distribution are :

num_hrefs, *num_self_hrefs*, *num_imgs*, *num_videos*, *kw_min_min*, *kw_max_min*, *kw_min_max*, *kw_max_max*, *kw_min_avg*, *kw_max_avg*, *self_reference_min_shares*, *self_reference_max_shares*

dataset preprocessing

Beta distribution

n_non_stop_words follows what appears to be a beta law. Indeed, normalization through gaussian and exponential methods were not satisfying. We decided to apply the inverse probability function of a beta distribution. A beta distribution requires two parameters such as :

$$E(X) = \frac{\alpha}{\alpha + \beta} \qquad V(X) = \frac{\alpha \cdot \beta}{(\alpha+\beta)^2 \cdot (\alpha+\beta+1)}$$

Thus, through the méthode des moments, we've estimated E(X) and V(X) by their common estimators Xnbar and Sn^2 and established the parameters by the following relations :

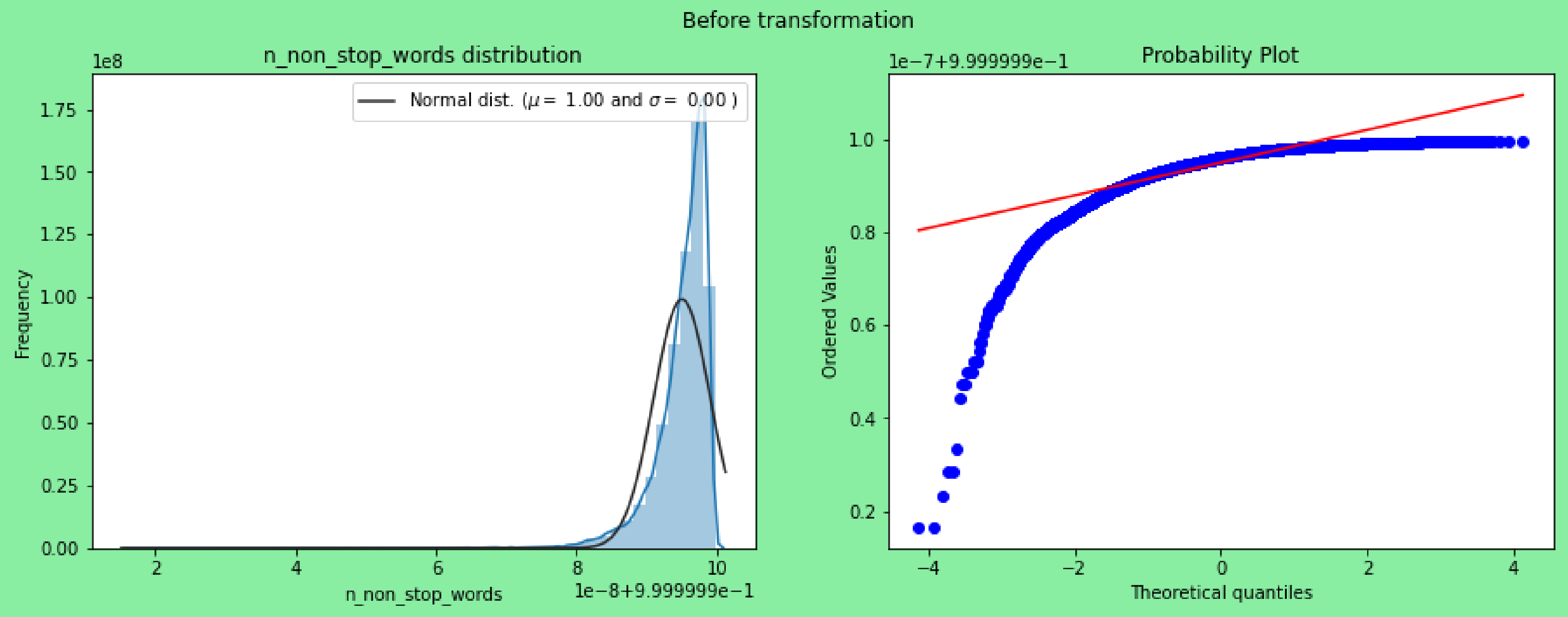
$$\alpha = k \cdot \beta \qquad \beta^2 + \frac{1}{k+1}\beta - \frac{k}{(k+1)^3 \cdot V(X)} = 0 \qquad k = \frac{E(X)}{1-E(X)}$$

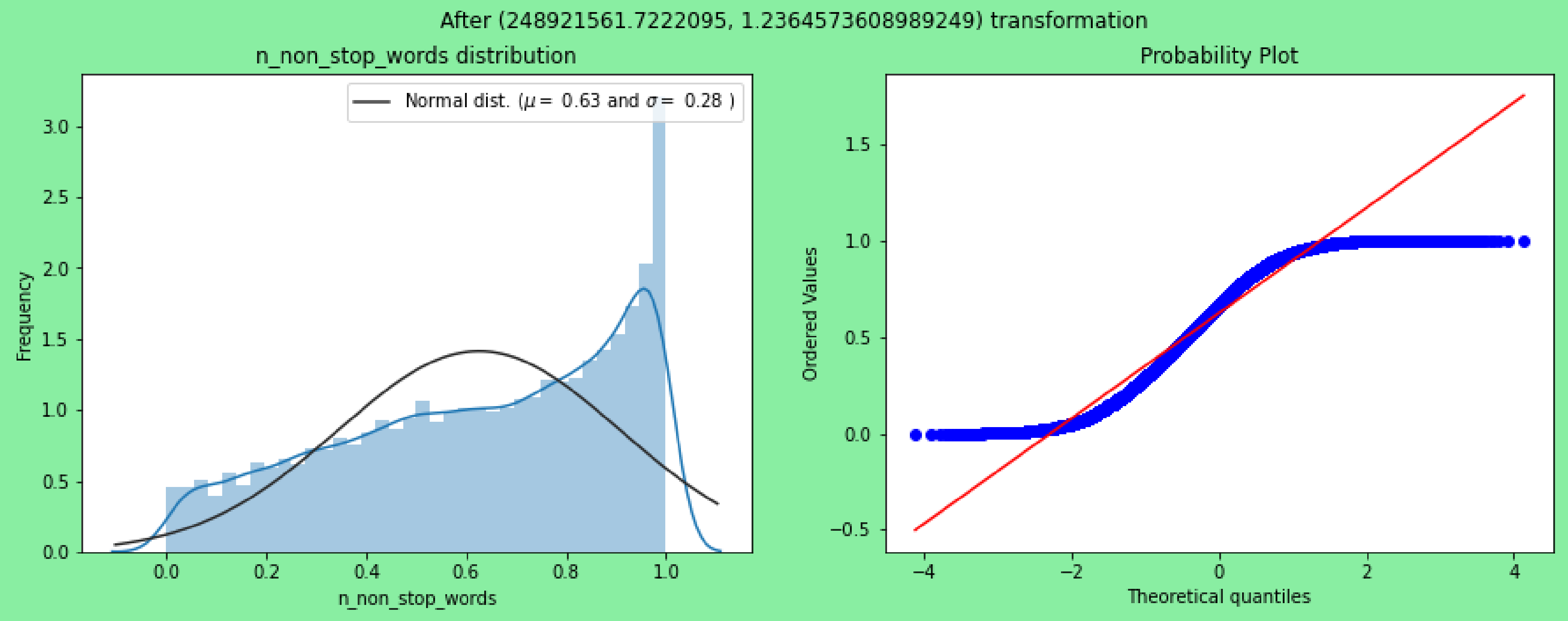
dataset preprocessing

Beta distribution

The estimated parameters were alpha = 2.10^6 and beta = 1.24. By observing some plots of the Beta distribution, these values do not surprise us, especially when we consider the length of the interval of values containing the feature (10^-8)

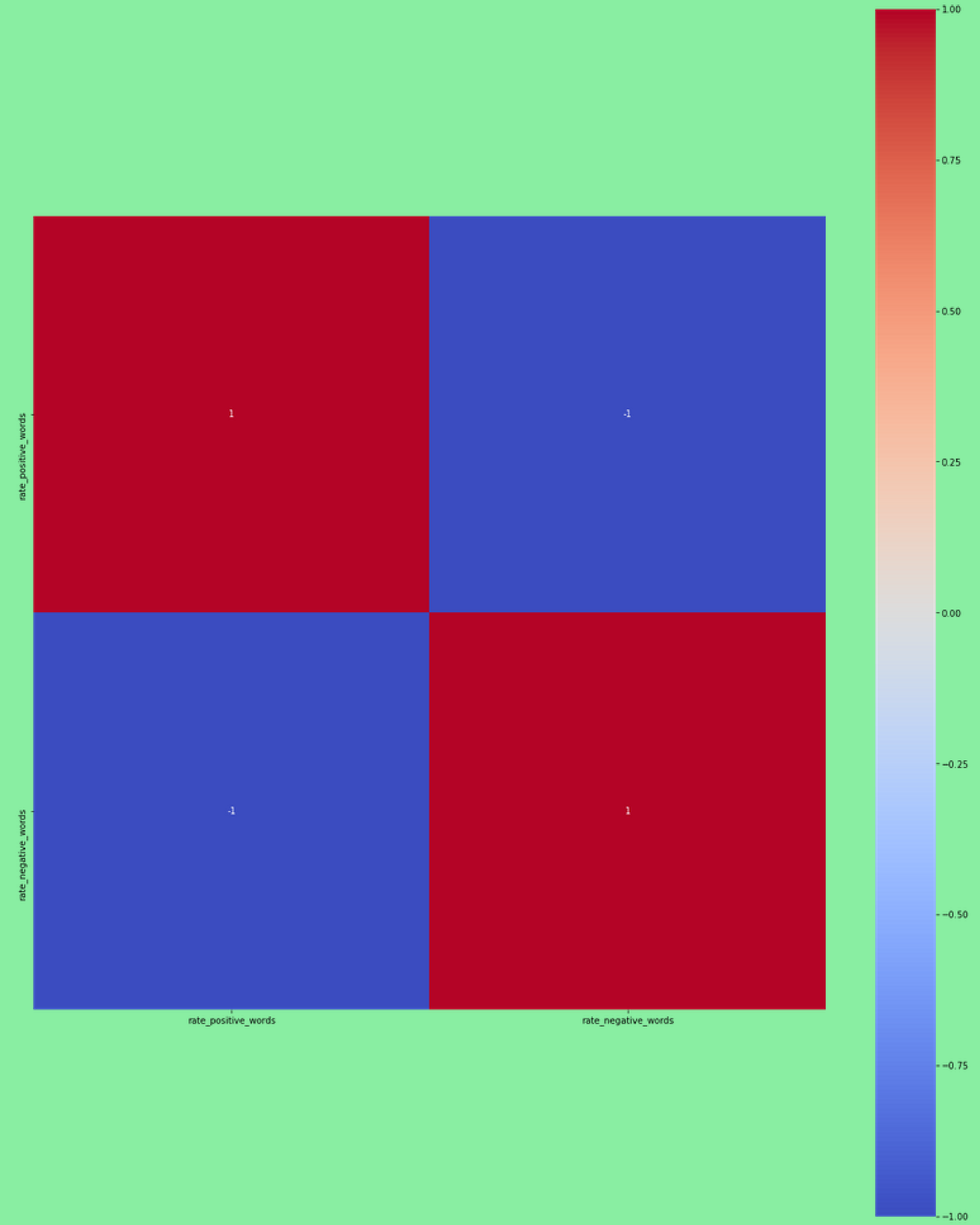
$$Z = f^{-1}_{\beta(\alpha,\beta)}(X)$$





dataset preprocessing

Perfect correlation between the pair of features



Correlated features

Multicolinearity is an ineluctable problem when assuming independancy between variables. For some sets of features, we've revealed through Pearson's and Spearman's methods linear dependencies inbetween features. In order to avoid overfitting by making the model distribute unreasonnably its share of variance to correlated features, we've decided to remove features who presented a high linear correlation with others. The most prevalent example was our case study of *rate_positive_words* and *rate_negative_words* case

dataset preprocessing

Correlated features

Other sets of features correlated are :

Day of the week : *weekday_is_monday*, *weekday_is_tuesday*, *weekday_is_wednesday*, *weekday_is_thursday*, *weekday_is_friday*, *weekday_is_saturday*, *weekday_is_sunday*, *is_weekend*. We've decided to remove *is_weekend* and *weekday_is_sunday* since the latters are evidently determined by the values of the other variables.

Keywords : *kw_min_min*, *kw_max_min*, *kw_avg_min*, *kw_min_max*, *kw_max_max*, *kw_avg_max*, *kw_min_avg*, *kw_max_avg*, *kw_avg_avg*. We've decided to remove *kw_avg_....* since they are intrinsically bound to the min and max features variant. Moreover, the arithmetic mean is heavily penalized by extreme values, thus both min and max are significant for estimating *kw_avg_....*.

dataset preprocessing

Correlated features

Other sets of features correlated are :

Global : *global_subjectivity*, *global_sentiment_polarity*, *global_rate_positive_words*, *global_rate_negative_words*. We've removed *global_sentiment_polarity*.

Self_reference : *self_reference_min_shares*, *self_reference_max_shares*, *self_reference_avg_shares*. We've removed *self_reference_avg_shares*.

Title : *title_subjectivity*, *title_sentiment_polarity*, *abs_title_subjectivity*, *abs_title_sentiment_polarity*. We've removed *abs_title_subjectivity* and *abs_title_sentiment_polarity*.

Polarity : *avg_positive_polarity*, *min_positive_polarity*, *max_positive_polarity*, *avg_negative_polarity*, *min_negative_polarity*, *max_negative_polarity*. We've removed *avg_positive_polarity* and *avg_negative_polarity*.

dataset preprocessing

Correlated features

One final correlated set of features is the set of *LDA_k*. The Latent Dirichlet Allocation (LDA) is a statistical method that allows the "gentrification" of the individuals following different topics. Here we suppose that every article is described by its "allegiance" to 5 different topics. These topics are mathematically established. A more known example in the academic field would be the Principal Component Analysis, which constructs artificial variables that suffice to explain the dataset, most of the time, the PCA reveals that most of the variance is explained by fewer number of variables than there are. As it is a probability, the law of total probability applies, thus, we have:

$$LDA_0 = 1 - \sum_{k=1}^{4} LDA_k$$

There isn't enough evidence to claim that two topics are related. It isn't a surprise since the LDA method constructs topics that are independent. Nevertheless, we'll remove *LDA_00* as it's a linear combination of the four others.

regression models

Metrics

We used the RMSE and MAE for our metrics. These metrics are expressed in the same units as the target shares, their values should give more insight than other metrics such as the Mean Square Error as well as the Mean Percentage Error. Moreover, the MAE gives the easier metric to interpret but is less sensitive to outliers as opposed to the RMSE. We don't know the requirements of the prediction problem, thus, for safety measures we'll only keep the MAE for insight but not for feature selection.

regression models

Decision Tree

```
RMSE: 14872.88
R_squared: -1.58
```

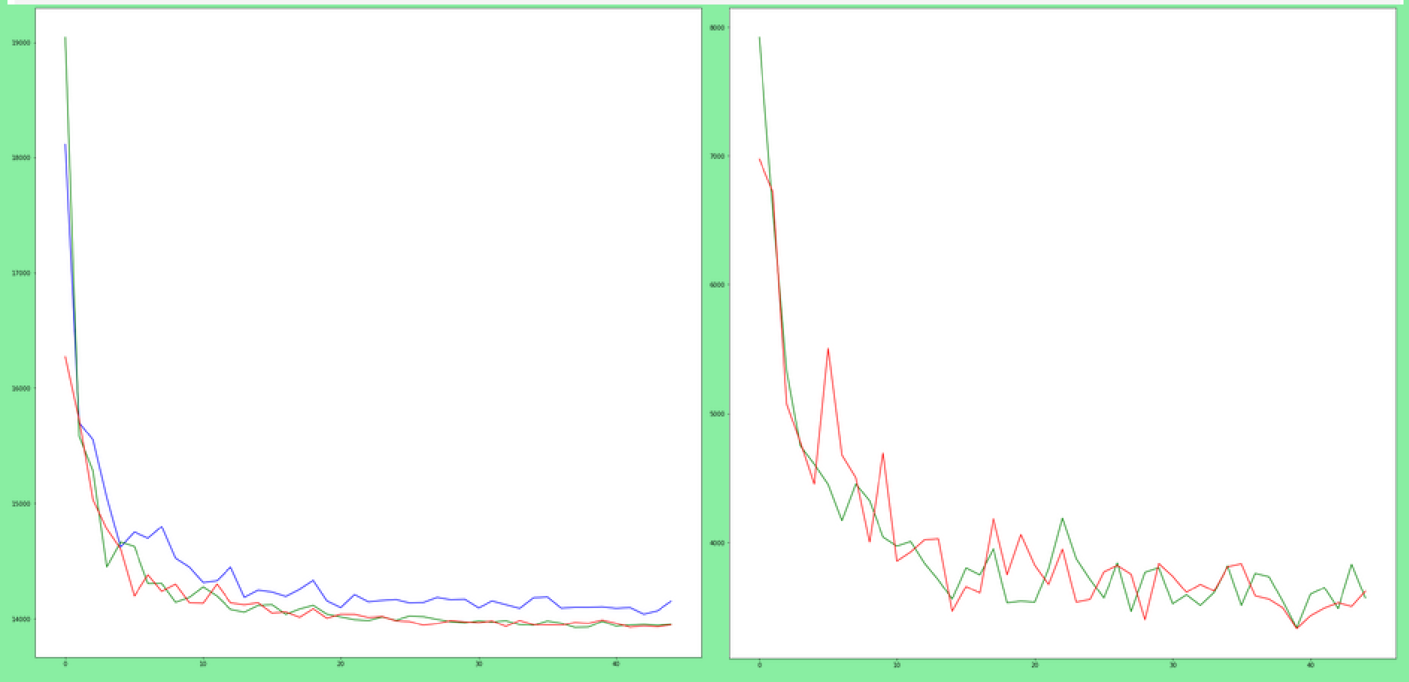
Our first model is a Decision Tree based of the DecisionTreeRegressor method from the sklearn library. Nevertheless, due to the negative R^2 score and our overall reluctance to pursue further with such an undocumented method. We pursued other models.

regression models

Random Forest

Number of trees vs. scores of the models

	trees	rmse_auto_train	rmse_auto_test	autoscore	rmse_sqrt_train	rmse_sqrt_test	sqrtscore	rmse_log_train	rmse_log_test	logscore
0	1.0	8827.530314	18117.069542	-0.687401	7920.751888	19045.088609	-0.864697	6974.639568	16270.598846	-0.360973
1	2.0	6073.380796	15701.456239	-0.267425	6576.184954	15580.697078	-0.248004	6721.330534	15727.812485	-0.271683
2	3.0	4989.018288	15549.860388	-0.243069	5332.373095	15280.679867	-0.200405	5075.616321	15031.839927	-0.161627
3	4.0	5170.124022	15049.521440	-0.164361	4746.117731	14447.011539	-0.072997	4773.907771	14776.715148	-0.122530
4	5.0	4551.223903	14619.213691	-0.098728	4605.444454	14662.103535	-0.105185	4451.905689	14601.507871	-0.096069
5	6.0	4608.693791	14749.054896	-0.118332	4448.443733	14623.840512	-0.099424	5505.929896	14195.092901	-0.035902
6	7.0	4991.080208	14695.413066	-0.110212	4168.114961	14304.139352	-0.051879	4675.853396	14378.772475	-0.062884
7	8.0	4671.343681	14797.452911	-0.125683	4451.518955	14304.845984	-0.051983	4500.825872	14236.172257	-0.041907
8	9.0	3986.697412	14522.438265	-0.084230	4322.692298	14141.781562	-0.028136	4002.464527	14298.023553	-0.050980
9	10.0	4133.028929	14445.354019	-0.072750	4041.501287	14184.013690	-0.034286	4691.180861	14138.268752	-0.027625



Test scores and Train scores

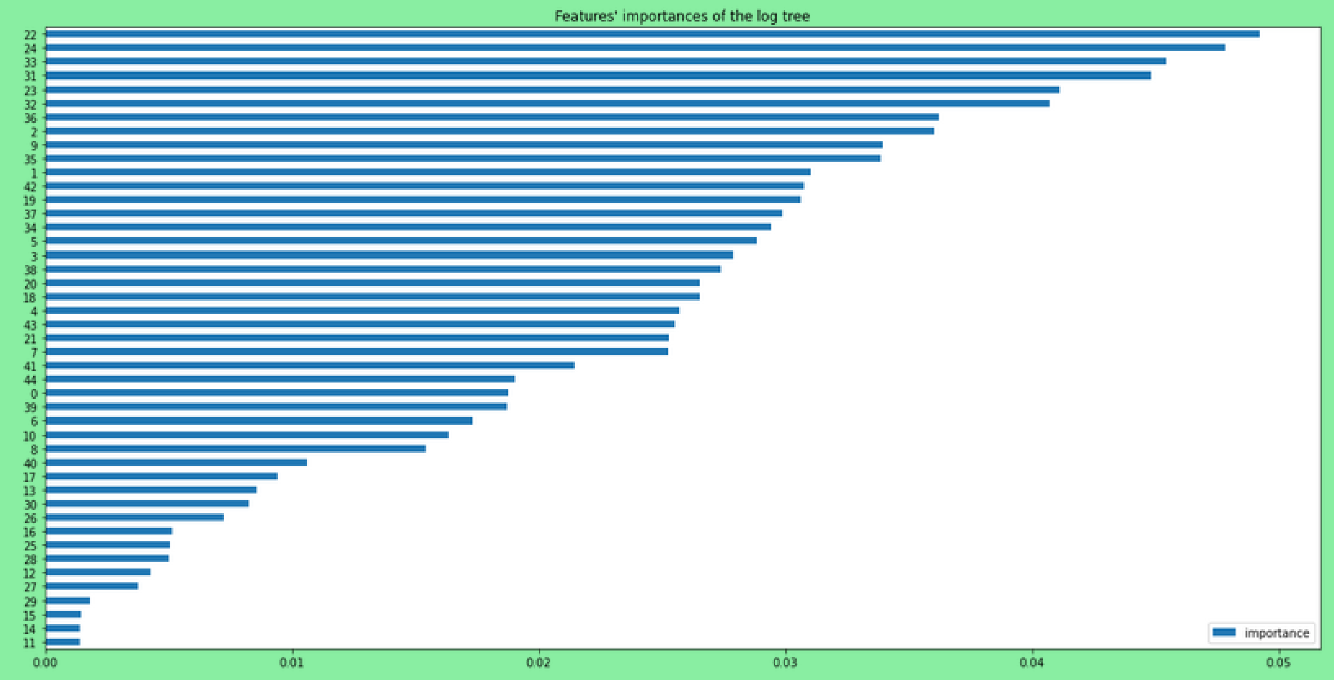
(Blue all features, Green log, Red sqrt)

Our second model was based of the same library with the RandomForestRegressor method. However, we did achieved better results thanks to the tuning parameters. We've tried for all possible values of trees and feature selection (all vs log vs sqrt features selected at every branch of the tree). We can see that limiting the number of features selected at every branch reduces overfitting. When plotting the train scores of the log and sqrt trees, the results were similar, we'll keep the log one.

regression models

Random Forest

Features' importance in the log tree



According to this model, the following features were the most significant in explaining the dataset :

kw_min_avg

self_reference_min_shares

LDA_02

weekday_is_saturday

kw_max_avg

LDA_01

regression models

Linear Regression

```
Coefficients:
 [   175.90083148   1271.44894633  -2850.00628519  -5678.96452739
  -1213.82244274    693.27872152  -1206.17325775    154.86412504
    613.4228829     146.44085535     39.19459217  -1335.35175143
  -1941.10072101  -1421.96498391  -1058.10482112   -794.19844864
   -963.34176543     61.3930096      50.51175021     76.71487694
   -229.00605901   -103.3159223     944.18034624   -224.42810929
    430.46304857    247.23357074   -408.90132091   -104.35740839
   -366.47106293   -194.41318493    568.43490459   -471.74343619
  -1776.50713924    -93.65295759   -711.52462341   2625.66155775
  -1270.66656989 -17422.88782181   -994.5017028   -3421.9955127
     25.88094201   -745.91226915    -64.83040913     26.12280706
    306.85129163]
Mean squared error: 83647529.6640169
Variance score: 0.025002277526087746
```

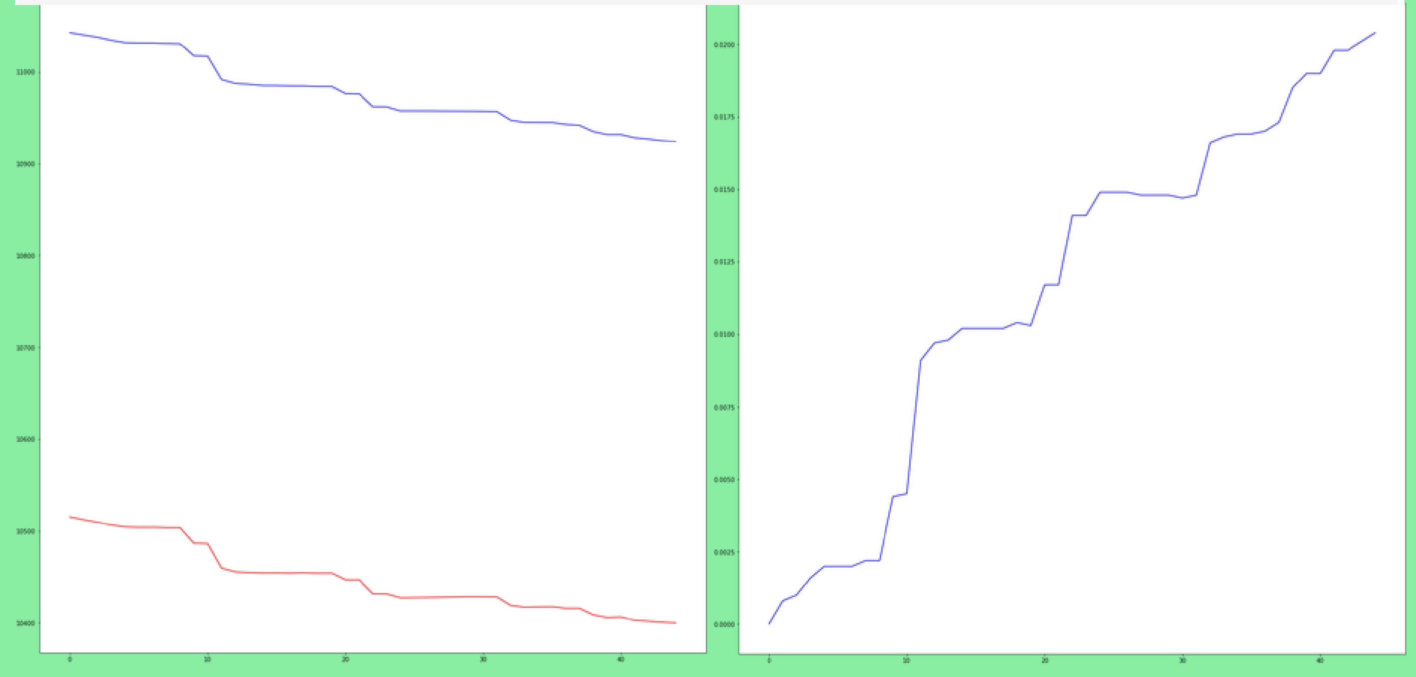
Our third model was based of the same library with the LinearRegression method. Despite our many efforts to make sense of the method, including recalculating the MSE to see if it matched the one plotted, the coefficients presented remain a mystery. Moreover, the R^2 score and missing p-values did not raise our hopes to get more opportunities for interpretation than the next models.

regression models

Linear Regression with 10CV FS

Feature selection steps and scores

	feature_selected	rsquared	rmse_train_score	rmse_test_score	mae_train_score	mae_test_score	rsquared_adj
0	title_sentiment_polarity	NaN	11042.851868	10515.587862	3099.778863	3099.960020	0.0000
1	title_subjectivity	NaN	11040.345859	10512.504732	3096.895150	3097.027703	0.0008
2	max_negative_polarity	NaN	11037.952607	10509.823581	3096.388607	3096.737927	0.0010
3	min_negative_polarity	NaN	11034.589857	10506.678996	3094.018396	3094.463835	0.0016
4	max_positive_polarity	NaN	11032.026287	10504.595971	3090.926883	3091.596028	0.0020
5	min_positive_polarity	NaN	11031.629144	10504.062130	3090.621290	3091.364502	0.0020
6	rate_positive_words	NaN	11031.622008	10504.163054	3090.775684	3091.532594	0.0020
7	global_rate_negative_words	NaN	11031.089500	10503.799569	3088.925289	3089.930620	0.0022
8	global_rate_positive_words	NaN	11030.778265	10503.673657	3089.746965	3090.688780	0.0022
9	global_subjectivity	NaN	11017.616593	10486.836799	3077.941595	3078.840376	0.0044
10	LDA_04	NaN	11017.089513	10486.388119	3080.421366	3081.470996	0.0045



CV scores (Left) and Rsquared adjusted (Right)
(Blue train RMSE, Red test RMSE)

Our ultimate model revolves around a time-consuming yet understandable approach : we exploited the statsmodels library to practice a 10-fold forward selection linear regression on the dataset. Though the R^2 is often used as the CV score, we changed it for the test score as out aim was to limit overfitting too many features and give the best prediction.

regression models

Linear Regression with 10CV FS

The maximal Rsquared adjusted value is 0.0204. This is 2% of the variance explained as did the other models. We did try other methods such as the Lasso regularization but we didn't bother consider them as separate models since the modules used are sparsely documented and the lack of possible interpretations overshadow any significant change in Rsquared. This model was used for our API model since it was the most complete.

regression models

Conclusion

We've learned much more about the regression models thanks to this project. Giving the opportunity to students to confront an unknown dataset and its challenges gave us a data scientist vibe that no other project could ever procure.

We came to the conclusion that this dataset, despite its complex nature, introducing features extracted from NLP algorithms, lacks real significant features. The scores were so low that kept questionning our methods, which was not a bad idea since we've put ten times more effort in our data preprocessing, however, to no avail.

Such a complex target that revolves around social interaction, human behavior, cultural interests and time undeniably needs a more complex approach than the regressions we've seen so far. Especially since the time dimension of the dataset has been removed.

api flask

For the API, we used Flask. We create a new model of simple regression with only three variables which were the number of worlds in the title, the number of image in the articles and the number of videos. Then we saved the model thanks to the library Joblib and we used it with flask to create our API. To have a more realistic model we decided to add in the form the URL, the theme and the day of the week even if they don't play a role in the prediction and after completing the form it gives us the result which correspond to the prediction of the number of shares according to the parameters that we entered.