January
2022

Python for Data Science

VIDAL Sara & TESTU Constantin

Online news popularity

dataset

01	INTRODUCTION	Context and why we chose this dataset.
02	THE PROBLEM	What are we looking at with this dataset.
03	THE FEATURES	A glimpse into the many features of the dataset
04	OUR STUDY	Our approach to the problem
05	CONCLUSION	Our results

Introduction

In the context of our fourth year of engineering studies and our first year specialized in data & AI we had a python project for the end of the semester: Apply what we learned in both python and machine learning on a dataset.

We chose this particular dataset as it was about Natural Language Processing (which was a field we did not know much about), and because it had a lot of features, so it seemed to require a lot of work.

VIDAL Sara & TESTU Constantin

The Problem

The dataset contains informations about online articles from a numeic media called Mashable. Each line of the dataset is an article. The target feature here is the number of shares the article had. Most explanatory feature is linked to the content of the article, often linked to NLP (Natural Language Processing), but also to its publication but we will see this in details later.

Thus the objective here is to predict the popularity, represented by its number of shares. To do so we will test different algorithms and try to construct a predictive model.

VIDAL Sara & TESTU Constantin

Online News Popularity Dataset

NUMBER OF INSTANCES	NUMBER OF ATTRIBUTES	DATE DONATED	ASSOCIATED TASKS:	AREA:
39797	61	2015-05-31	Classification, Regression	Numerical Medias

The Features

We have 61 different features in the dataset:
58 predictive attributes, 2 non-predictive, 1 goal field.

The URL of the article is not predictive as well as the date at which each article was published (timedelta).

We do not want to use the timedelta as a predictive variable because if the website got more popular with time, there would be a strong correlation between the number of share and the publication date, that would maybe hinder our work on trying to understand how the content of the article affects the popularity.

VIDAL Sara & TESTU Constantin

List of features:

0. url: URL of the article (non-predictive)
1. timedelta: Days between the article publication and the dataset acquisition (non-predictive)
2. n_tokens_title: Number of words in the title
3. n_tokens_content: Number of words in the content
4. n_unique_tokens: Rate of unique words in the content
5. n_non_stop_words: Rate of non-stop words in the content
6. n_non_stop_unique_tokens: Rate of unique non-stop words in the content
7. num_hrefs: Number of links
8. num_self_hrefs: Number of links to other articles published by Mashable
9. num_imgs: Number of images
10. num_videos: Number of videos
11. average_token_length: Average length of the words in the content
12. num_keywords: Number of keywords in the metadata
13. data_channel_is_lifestyle: Is data channel 'Lifestyle'?
14. data_channel_is_entertainment: Is data channel 'Entertainment'?
15. data_channel_is_bus: Is data channel 'Business'?
16. data_channel_is_socmed: Is data channel 'Social Media'?
17. data_channel_is_tech: Is data channel 'Tech'?
18. data_channel_is_world: Is data channel 'World'?
19. kw_min_min: Worst keyword (min. shares)
20. kw_max_min: Worst keyword (max. shares)
21. kw_avg_min: Worst keyword (avg. shares)
22. kw_min_max: Best keyword (min. shares)
23. kw_max_max: Best keyword (max. shares)
24. kw_avg_max: Best keyword (avg. shares)
25. kw_min_avg: Avg. keyword (min. shares)
26. kw_max_avg: Avg. keyword (max. shares)
27. kw_avg_avg: Avg. keyword (avg. shares)
28. self_reference_min_shares: Min. shares of referenced articles in Mashable
29. self_reference_max_shares: Max. shares of referenced articles in Mashable
30. self_reference_avg_sharess: Avg. shares of referenced articles in Mashable
31. weekday_is_monday: Was the article published on a Monday?
32. weekday_is_tuesday: Was the article published on a Tuesday?
33. weekday_is_wednesday: Was the article published on a Wednesday?
34. weekday_is_thursday: Was the article published on a Thursday?
35. weekday_is_friday: Was the article published on a Friday?
36. weekday_is_saturday: Was the article published on a Saturday?
37. weekday_is_sunday: Was the article published on a Sunday?
38. is_weekend: Was the article published on the weekend?
39. LDA_00: Closeness to LDA topic 0
40. LDA_01: Closeness to LDA topic 1
41. LDA_02: Closeness to LDA topic 2
42. LDA_03: Closeness to LDA topic 3
43. LDA_04: Closeness to LDA topic 4
44. global_subjectivity: Text subjectivity
45. global_sentiment_polarity: Text sentiment polarity
46. global_rate_positive_words: Rate of positive words in the content
47. global_rate_negative_words: Rate of negative words in the content
48. rate_positive_words: Rate of positive words among non-neutral tokens
49. rate_negative_words: Rate of negative words among non-neutral tokens
50. avg_positive_polarity: Avg. polarity of positive words
51. min_positive_polarity: Min. polarity of positive words
52. max_positive_polarity: Max. polarity of positive words
53. avg_negative_polarity: Avg. polarity of negative words
54. min_negative_polarity: Min. polarity of negative words
55. max_negative_polarity: Max. polarity of negative words
56. title_subjectivity: Title subjectivity
57. title_sentiment_polarity: Title polarity
58. abs_title_subjectivity: Absolute subjectivity level
59. abs_title_sentiment_polarity: Absolute polarity level
60. shares: Number of shares (target)

- You can notice features linked to the conditions of publication of the article, like the category in which it appeared on Mashable (*13. data_channel_is_lifestyle*), the day of the week it was published (*31. weekday_is_monday*), etc...

- There are features related to the format of the article, such as the number of images in it (*9. num_imgs*), or the number of words in the title (*2. n_tokens_title*).

- Finally, most of the features were related to NLP: they were tools measuring the subjectivity or the positivity of the content or the title (*56. title_subjectivity*), or evaluating key words (*19. kw_min_min*) to which scores were given (e.g. a word like "terrorism" will have a much stronger score than a word like "and")

Our Study

using python

DATA VISUALIZATION

A first visualization that would allow us to better deal with the data set.

SPLITTING THE DATA

We used the sklearn splitter with 80% of the data in the training set, and 20% in the validation set.

SCALING THE DATA

We used a robust scaler for the scaling.

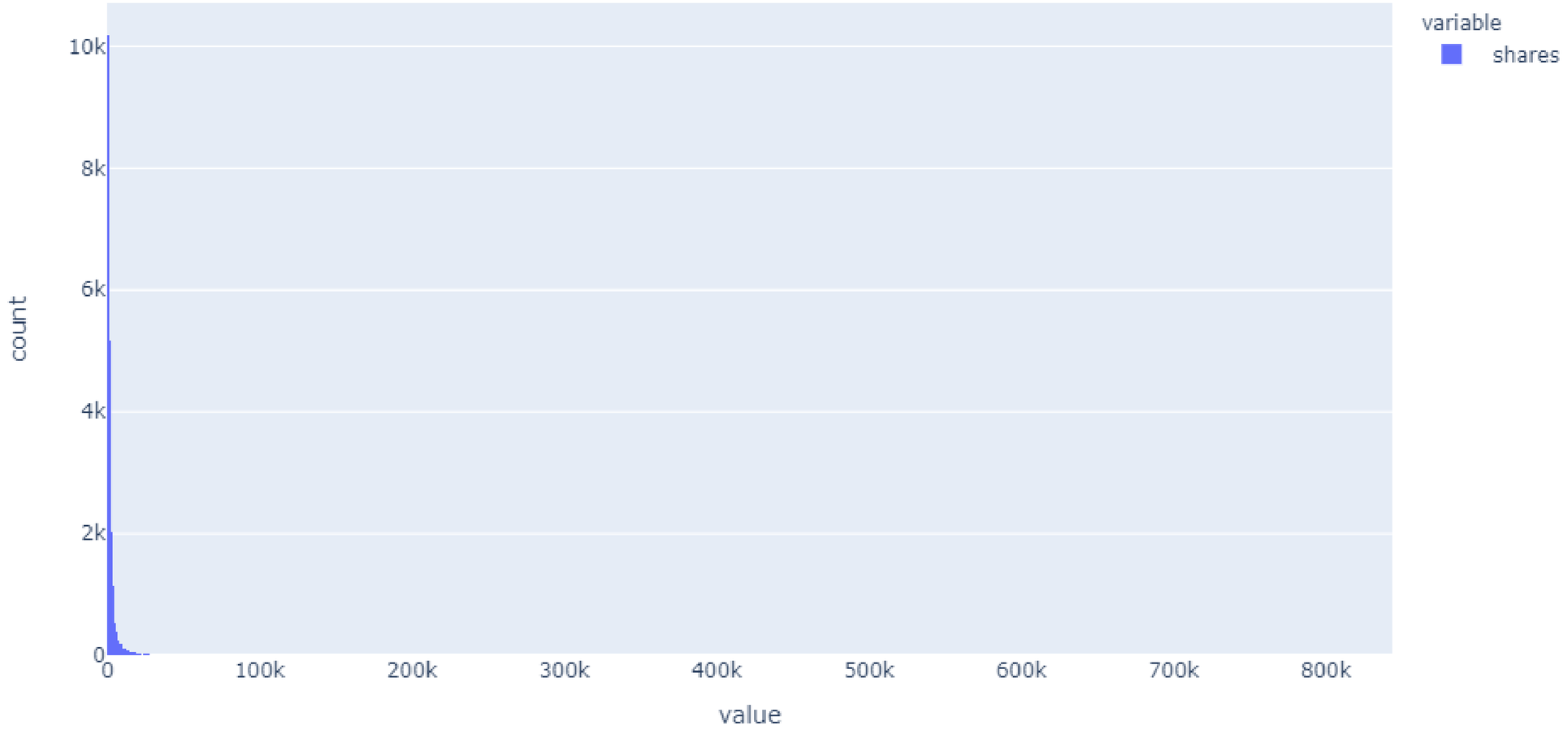
TRAINING THE MODELS

We tested several models for both regression and classification.

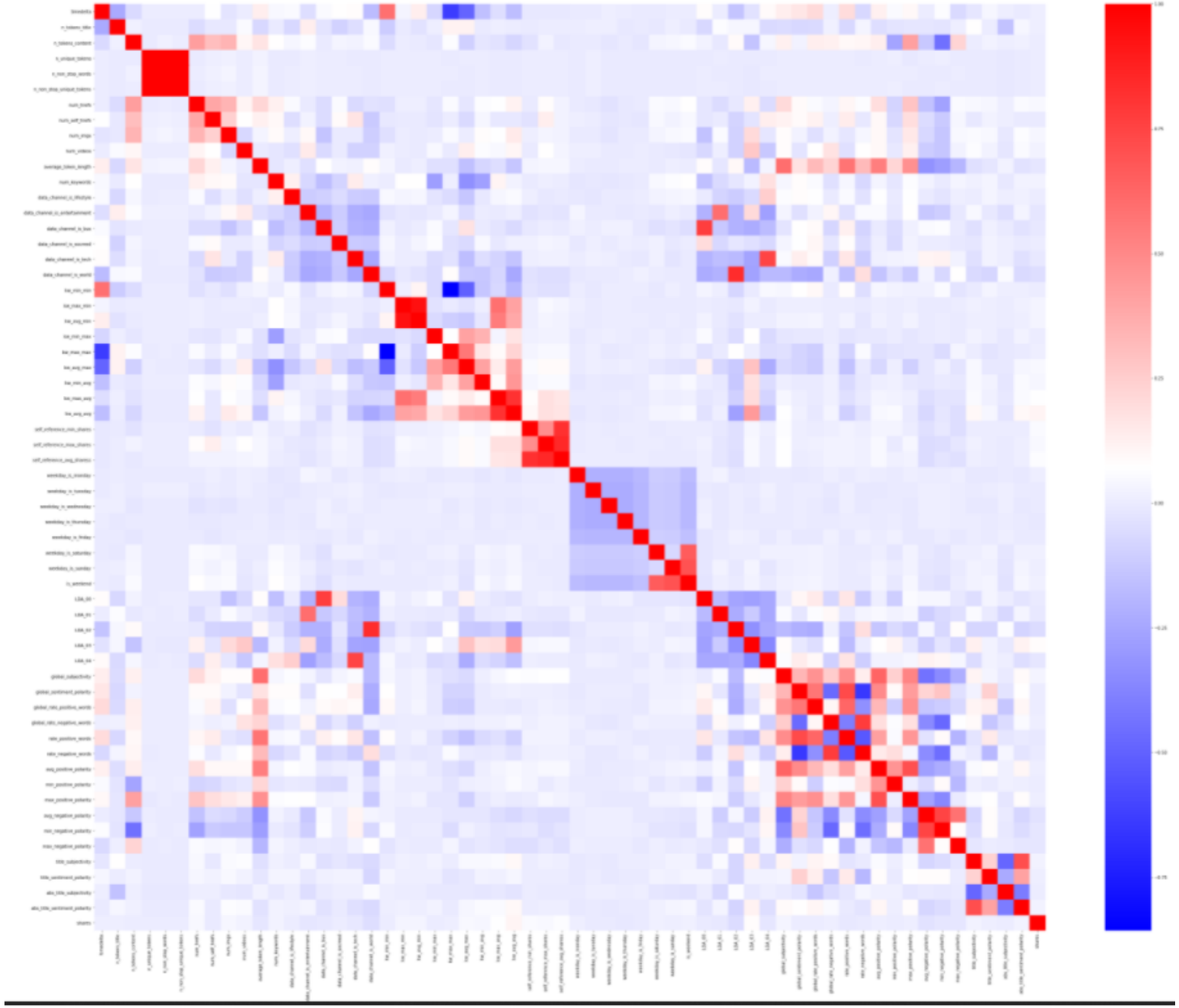
OPTIMIZATION

We used a Grid search for optimizing the hyperparameters

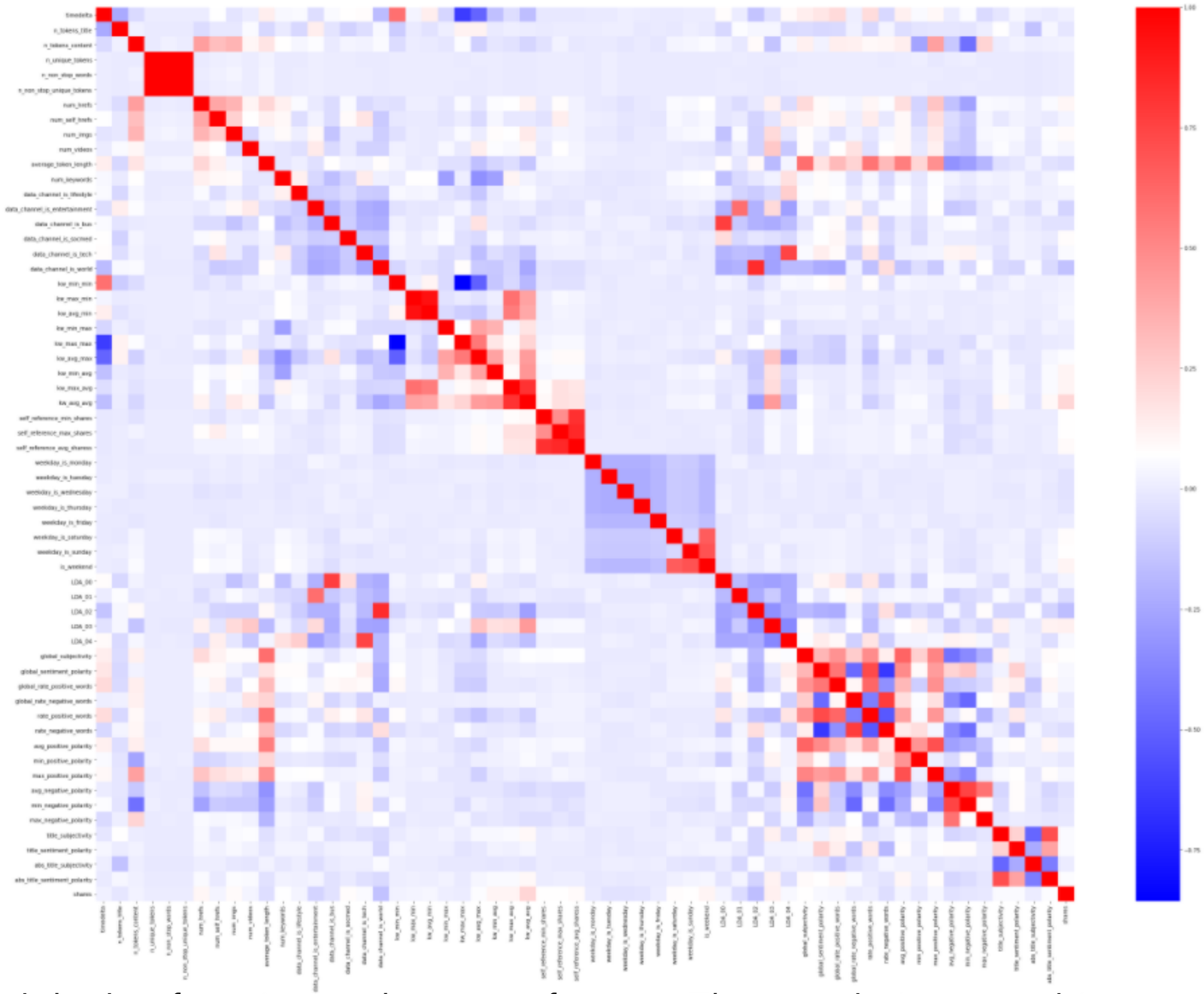
Data visualization



This represents the number of instances there are for each bins of shares (target value). We realized that there were many outliers.

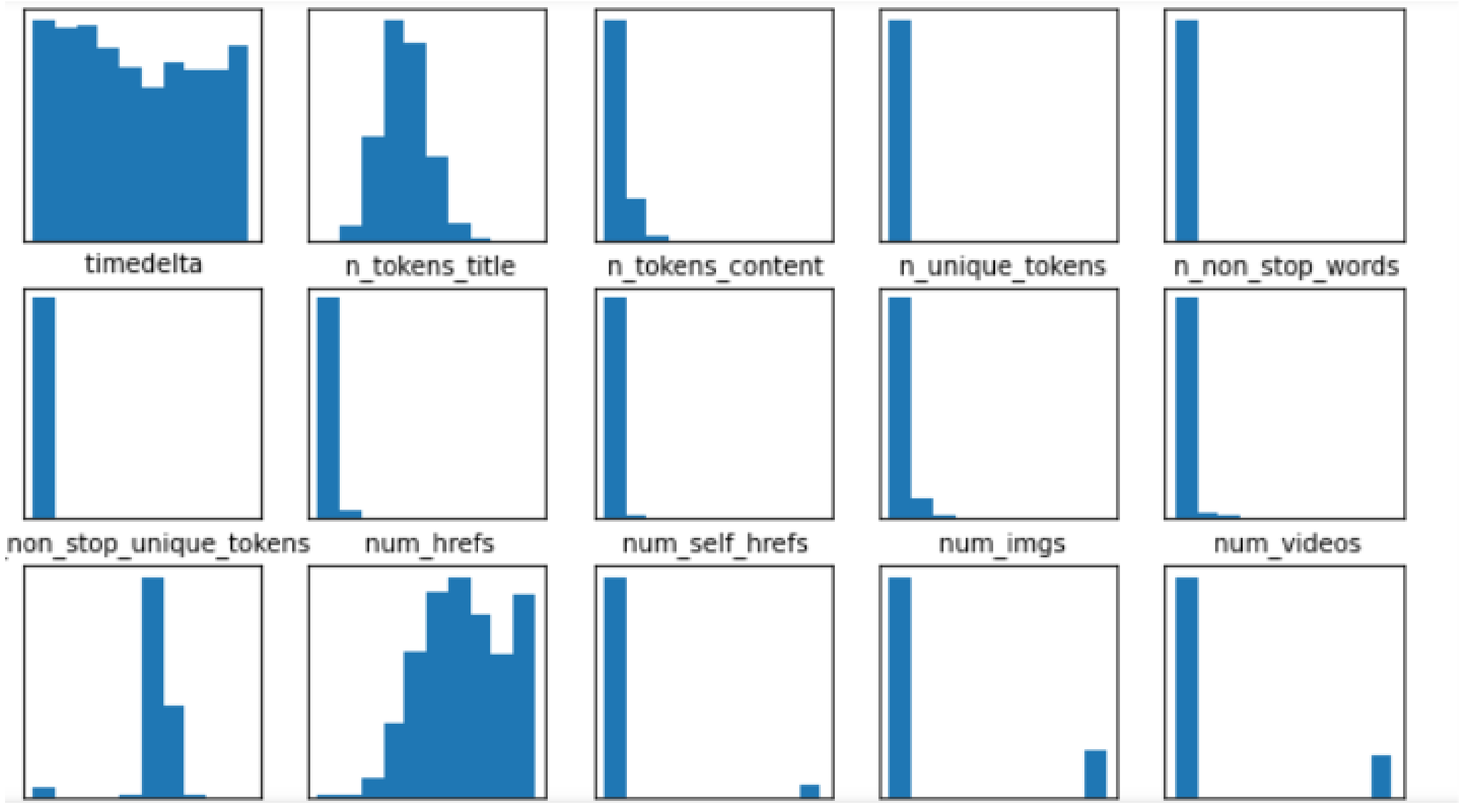


This correlation matrix shows us that the target feature has very few correlation with the other features (very light colours on the bottom line)



Here we applied the log function to the target feature. The correlation are a bit more contrasted (bottom line) so we shall want to keep that log applied.

Scaling



Those are the bar plots of the different features. We understood here that there were a lot of outliers. this is why we chose the robust scaler for scaling our data as it uses the median values instead of the means. That way the scaling is not unbalanced by the outliers.

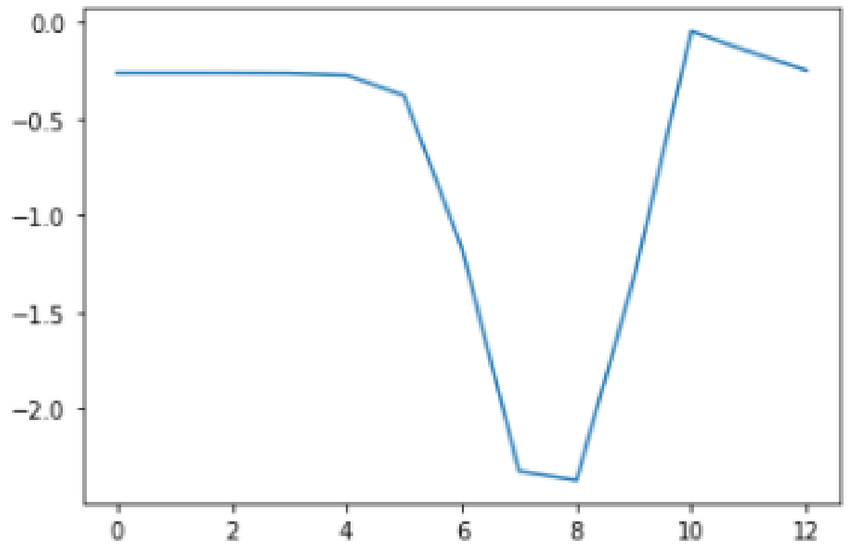
Training the models:

We tried to apply a regression to the dataset. First a Ridge Regression, then a Random Forest Regression but the results were really bad, even when we applied the log function to the target feature:

Ridge Regression:

```
Entrée [19]: plt.plot(val_score.mean(axis=1))

Out[19]: [<matplotlib.lines.Line2D at 0x108410ec820>]
```



Random Forest:

```
Entrée [25]: predicted_test = model.predict(X_test)
             Metric(y_test, predicted_test)

Mean Absolute Error: 3100.673736660987
Mean Squared Error: 73348008.01187077
Root Mean Squared Error: 8564.345159547855
```

Then, by reading the scientific paper of those who studied the dataset, we realized that they were actually doing a classification : each article was either popular or not popular based on a threshold number of shares.

The mean value is close to the 80th percentile value. Thus if a number of share is above average, it means that it is better than 80% of the articles of the dataset.

We will then set our threshold to 3395 to sort the articles between not popular (0) and popular (1)

Entrée [35]:

```
popularity_threshold = 3395
df["popular"] = np.where(df[" shares"]>popularity_threshold,1,0)
df.head()
```

We tried 4 different classification models:

1. Random Forest
2. Adaptive Boosting Classifier
3. K-Nearest-Neighbours (KNN)
4. Naive Bayes

As the target feature had about 80% '0' and 20% '0', using the accuracy as a score would have been pointless, so we used ROC, and AUC to evaluate our models.
Here are the AUC of the ROC for each model:

	Model	Score AUC
0	Random Forest	0.704491
1	Adaptive Boosting	0.704720
2	KNN	0.620429
3	Naive Bayes	0.579315

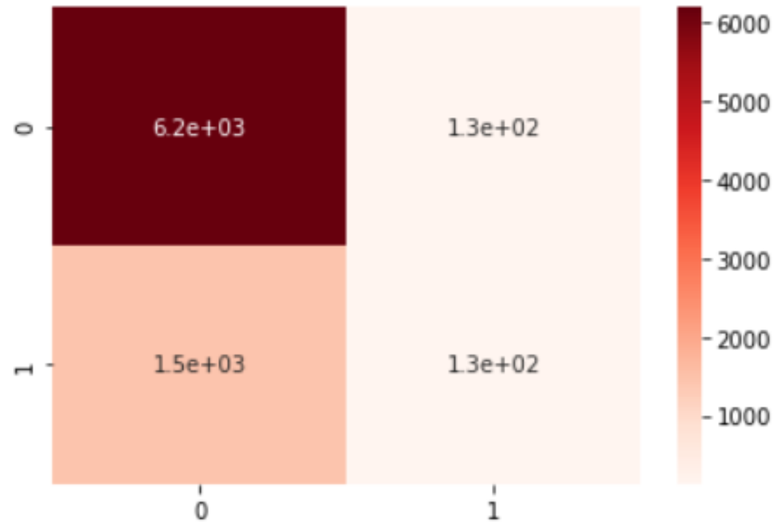
So we selected Adaboost as our model as it had the best score. We optimized it with a gridsearch and found 100 as the best number of estrimators.

Conclusion

After failing with the regression, we eventually managed to have good results (about 0.70 score of AUROC) with an adaboost classification.

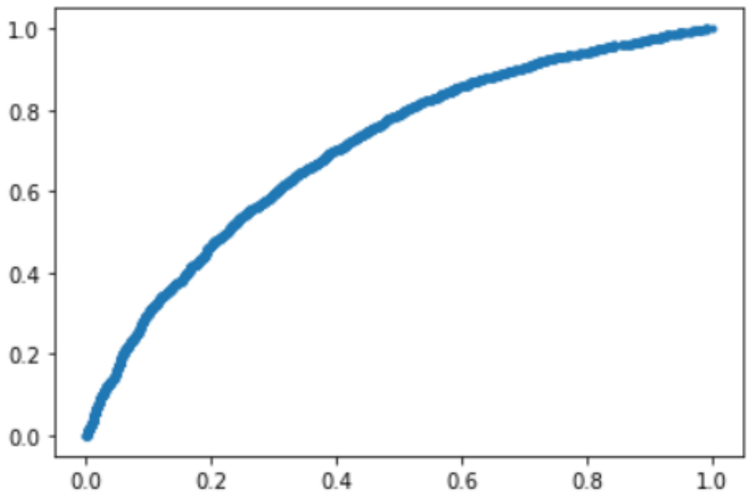
Here are the confusion matrix and ROC of our final model:

```
In [77]: ConfusionMatrix(confusion_matrix(y_test, grid_ada.best_estim
```




```
Final Model : AUROC = 0.706

Out[78]: [<matplotlib.lines.Line2D at 0x1083aed0430>]
```



VIDAL Sara & TESTU Constantin