GABRIEL SHENOUDA & BENJAMIN LAMBERT
ESILV - DIA1

Python for data analysis — 2021

ONLINE NEWS POPULARITY

CONTENTS

$\longrightarrow$

CONTEXT

- This dataset summarizes a heterogeneous set of features about articles published by Mashable in a period of two years.

- The goal is to predict the number of shares in social networks (popularity).

THE DATASET

—*The dataset is composed of 61 attributes*

— *It has 39797 instances*

— *It's area is business*

— *Relevant paper :*

K. Fernandes, P. Vinagre and P. Cortez. A Proactive Intelligent Decision – Support System for Predicting the Popularity of Online News. Proceedings of the 17th EPIA 2015 – Portuguese Conference on Artificial Intelligence, September, Coimbra, Portugal.

58 PREDICTIVE ATTRIBUTES

Such as :
- The number of words in the title,
- The number of videos,
- The day of the week it was published on, etc.

2 NON-PREDICTIVE

- The URL
- The timedelta (Days between the article publication and the dataset acquisition)

1 GOAL FIELD

- The number of shares (popularity)

FIRST STEPS

## 🌿 DATA PRE-PROCESSING

The first step is to clean our data :
- Deleting empty spaces in attributes' names
- Droping articles containing 0 word
- Droping the non-predictive features
- Droping the duplicates and NA
- Droping a part of highly corelated features ("n_non_stop_unique_tokens","n_non_stop_words","kw_avg_min")

05

HOW TO INTERPRET THE TARGET

- We tried regression models like linear regression or random forest regression (and others) but the results weren't good (MSE too big, score too low).
- The goal is to predict the popularity : Being able to say the article is popular or not is far more representative than a number without its context. Setting only 2 classes gives us a better accuracy too.
- We tested to set the thresh old value to set the popular/unpopular label with the mean of shares of every articles. => The accuracy was the best among every models we could find on the net but the recall and F1-score were around 0 (the model was able to recognize an article was unpopular very well because it had many of this examples, but sucked when it had to predict that a popular article was popular)
- So finally we choose to set this value to the Median which is 1400 and create a popularity column which takes 2 if the article is unpopular, else 4.
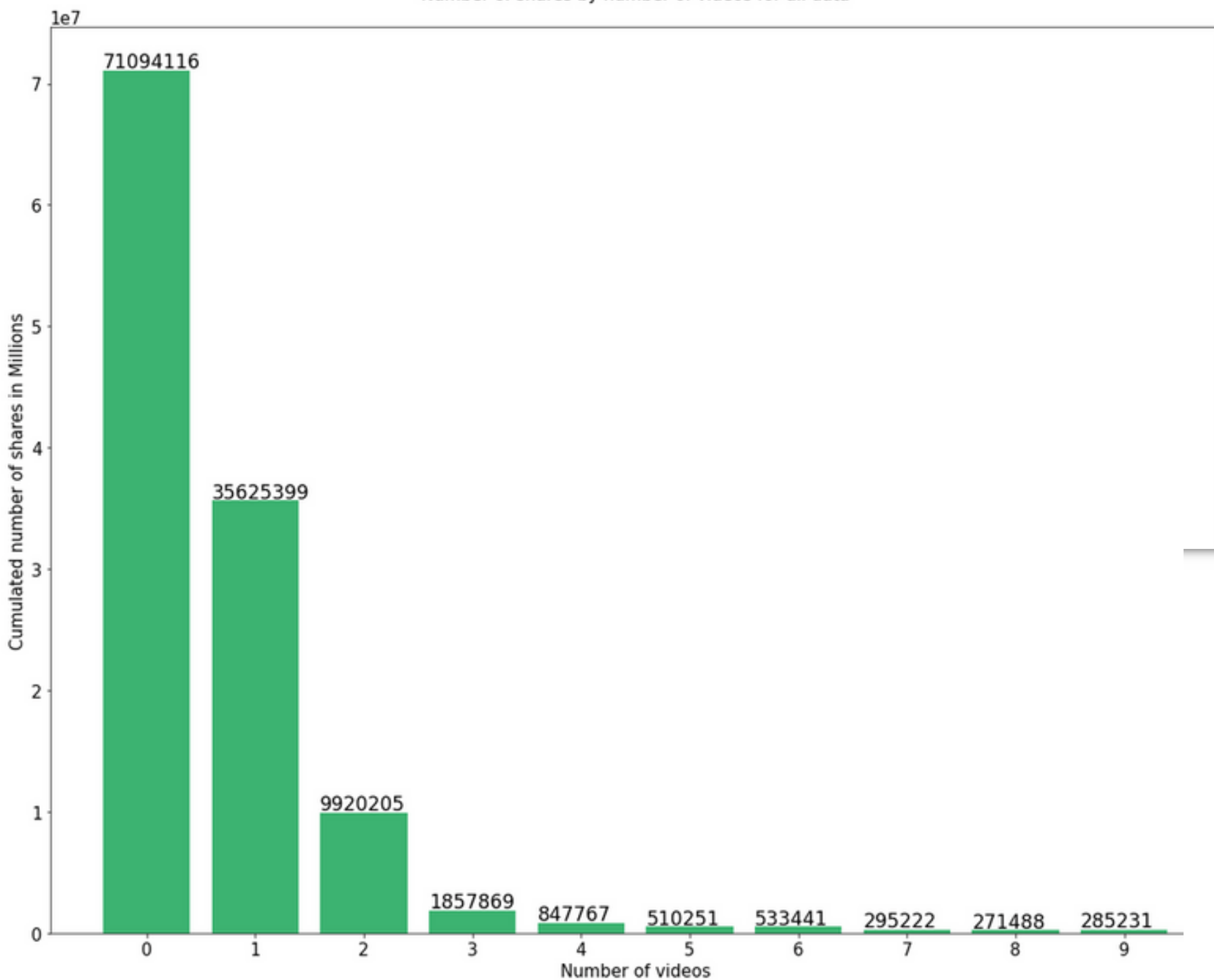
Number of shares by number of images for all data
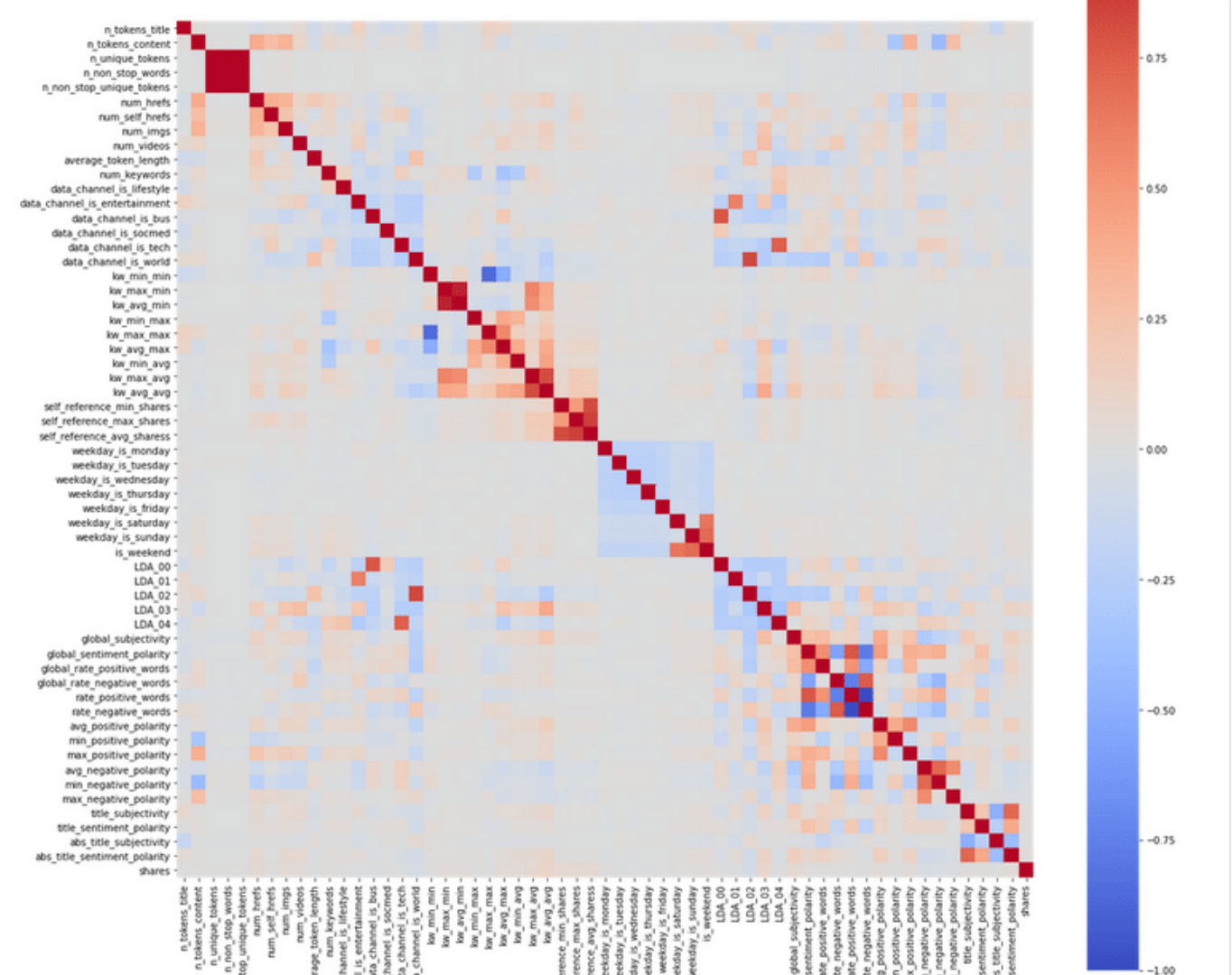
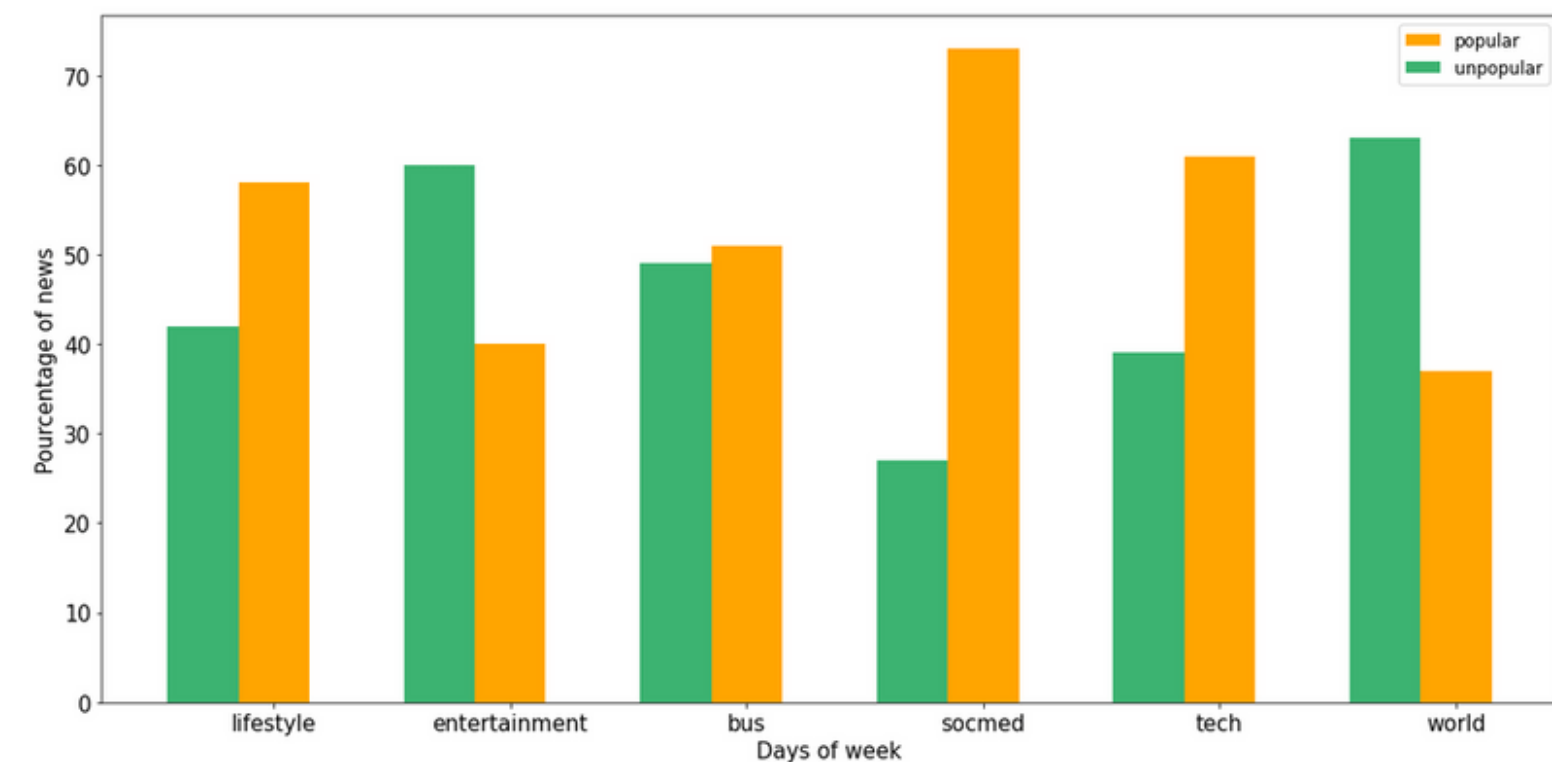

Popularity by number of words
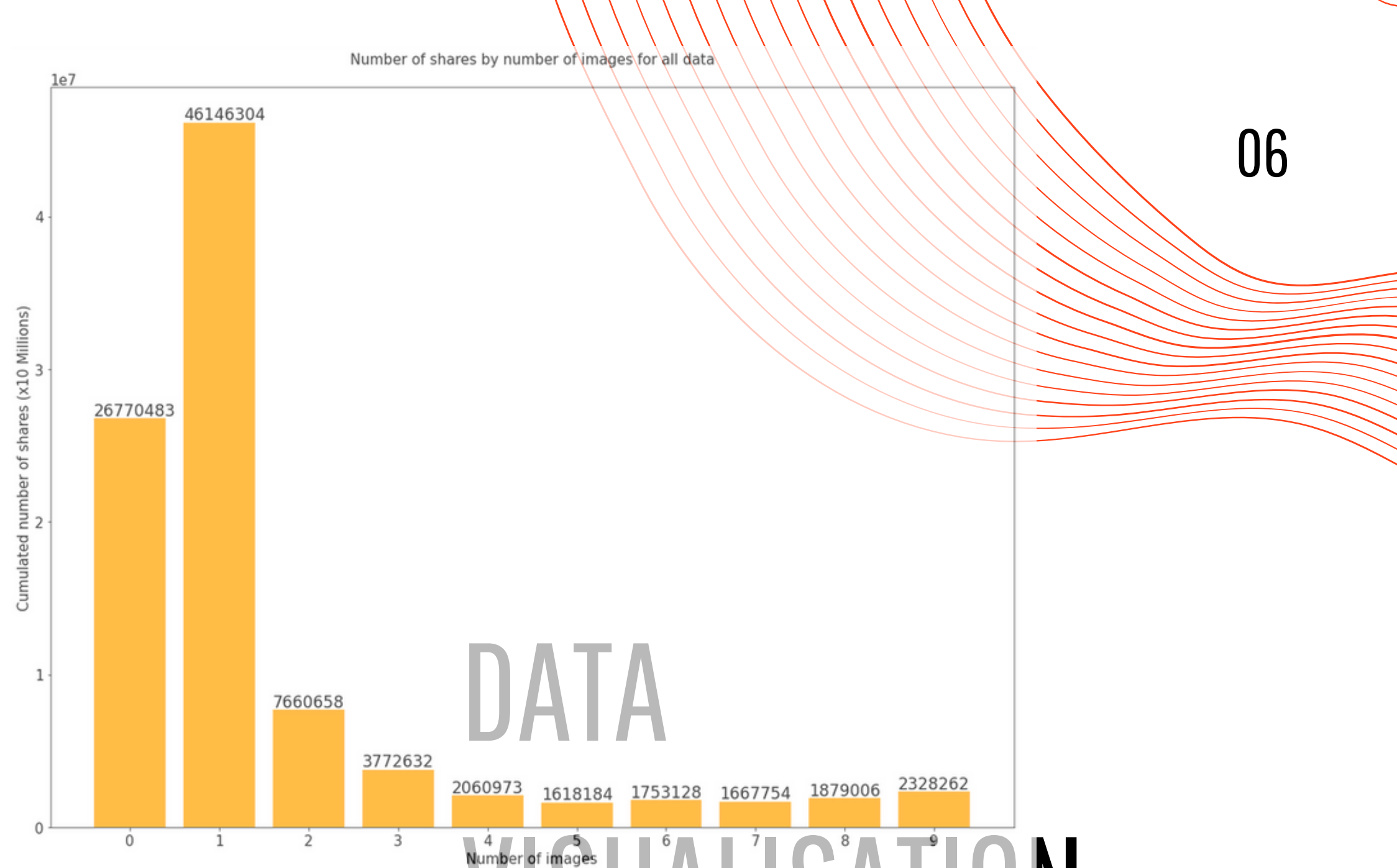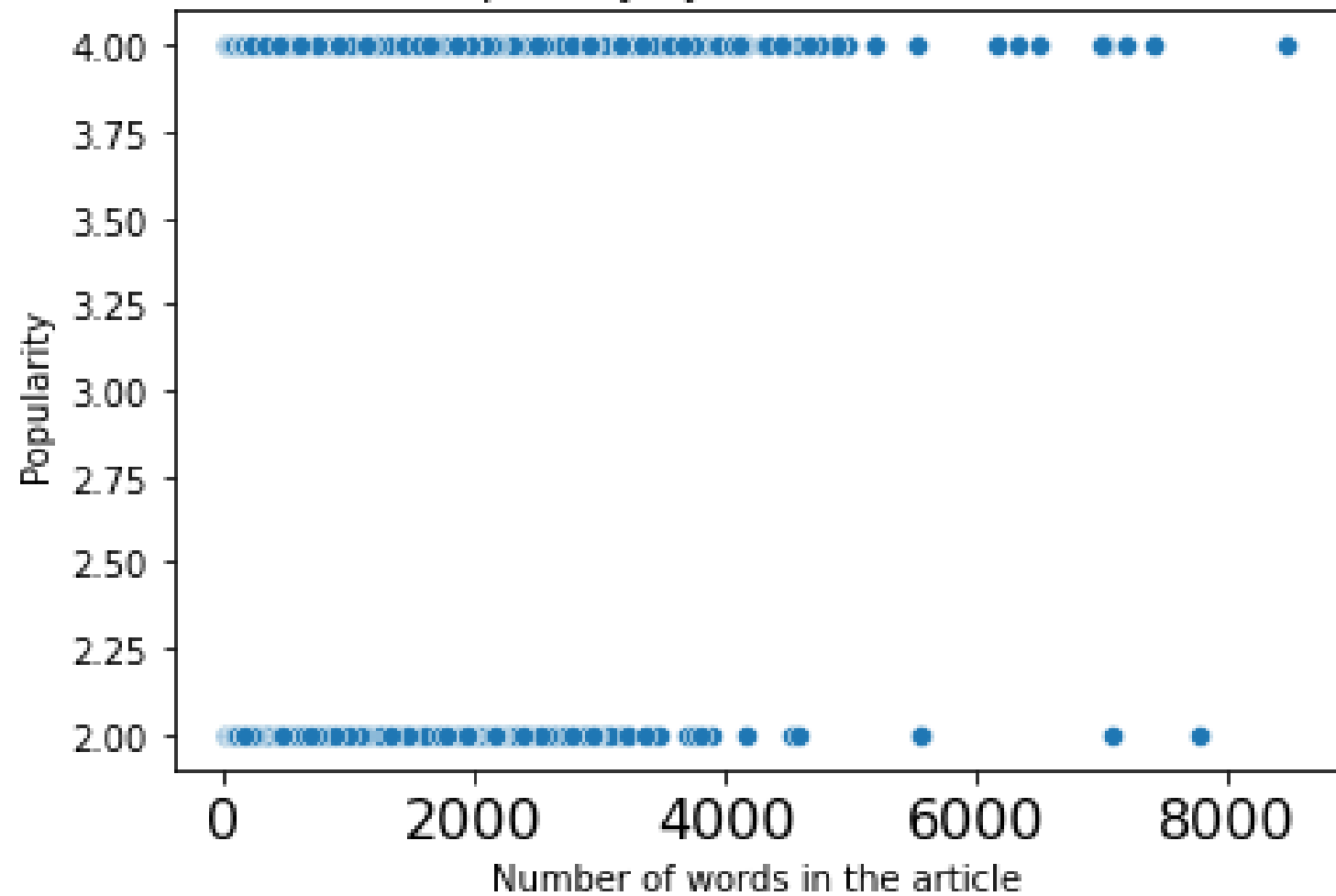
DATA
VISUALISATION

— *We tried to demonstrate visually many interesting points in order to better understand the data set and the links with the variable shares*
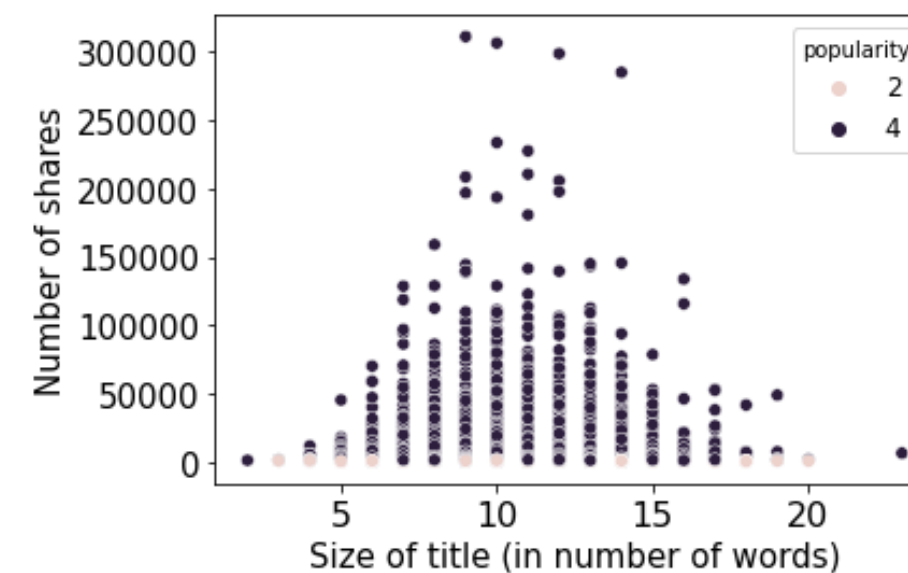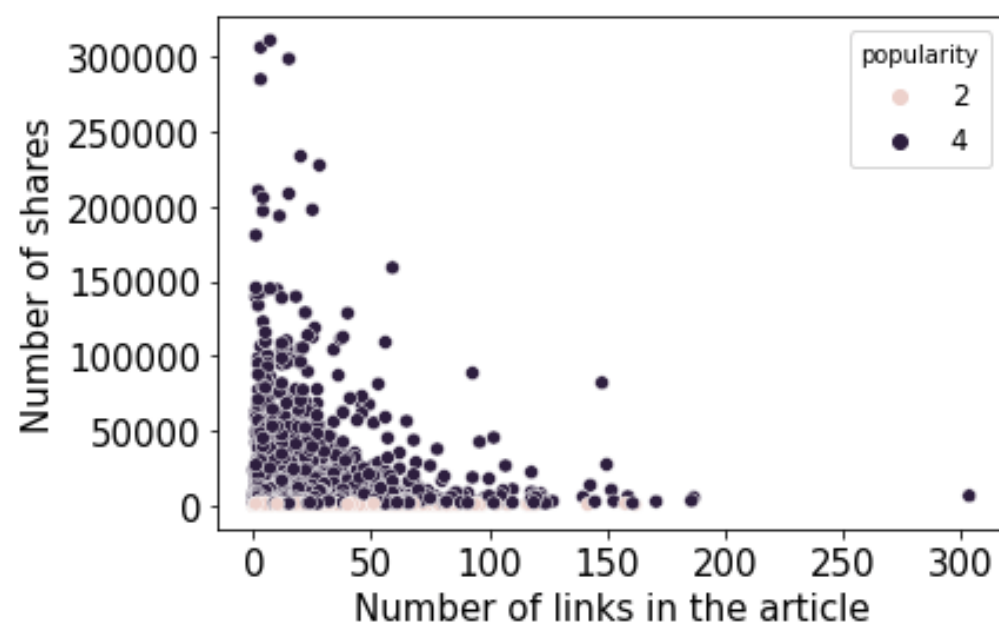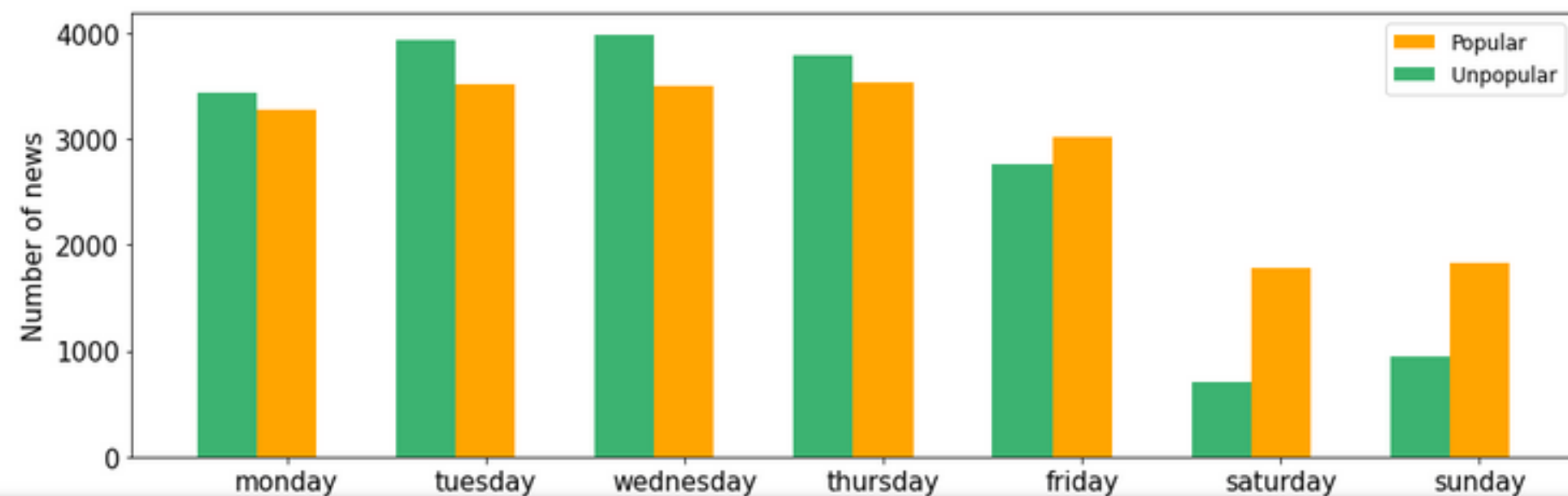
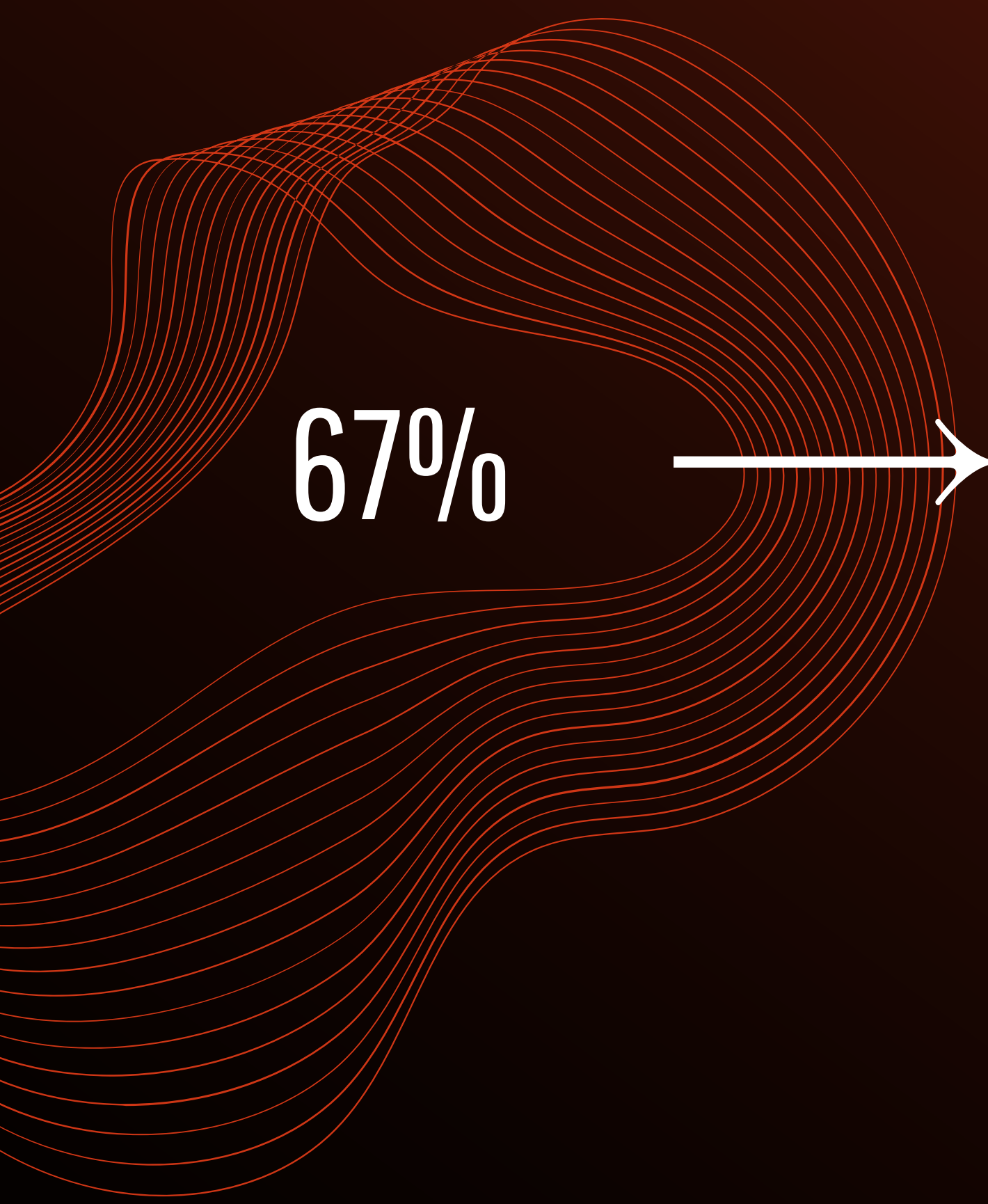Number of shares by number of videos for all data

Proportion of popular/unpopular news by data channel

Number of popular and unpopular news by day

07

CLASSIFICATION

— We tried : Gradient Boosting, Random Forest, Ada Boost, Bagging, Logistic Regression, Naive bayes and KNN,
and tuned them to obtain the best accuracy possible. We don't forget to look at other metrics like recall, precision, F1-score, AUC... We also scaled the data with a robust and a standard scaler when the models needed it.

— 67% of accuracy is the best result we obtained using Gradient Boosting Classification with the threshold set to 1400. The recall and F1-score are good in this case (so is the precision and AUC).

67%

CLASSIFICATION - CONTINUED

AN EXAMPLE OF HOW WE TUNED THE MODELS

We try to find the best number of estimators for the model to have the best accuracy

AN EXAMPLE OF HOW WE TUNED THE MODELS
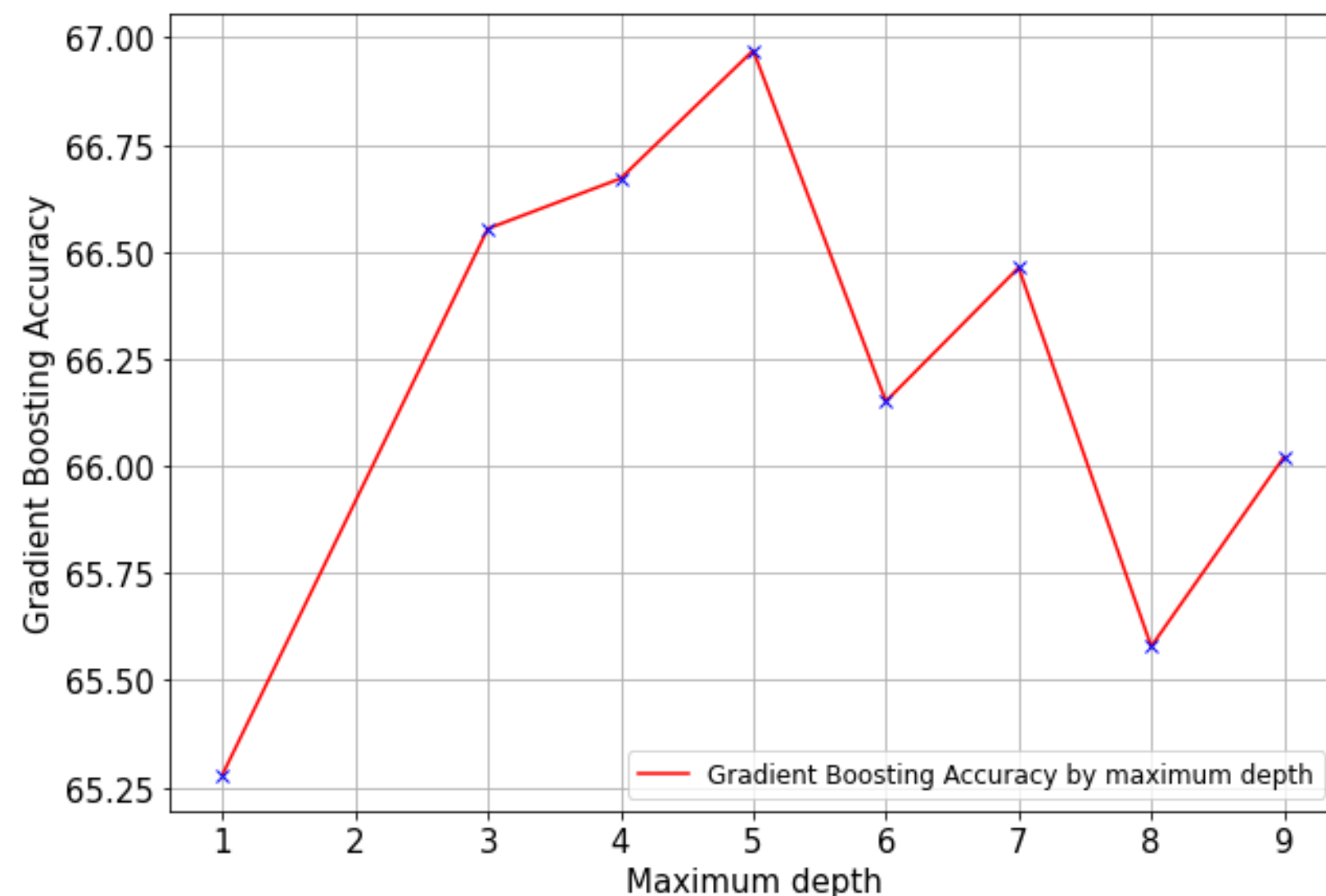
```
Gradient Boosting Classifier Accuracy : 0.6696997270245678
Gradient Boosting Classifier AUC : 0.66677052358857
Gradient Boosting Classifier Recall : 0.711393022684606
Gradient Boosting Classifier Precision : 0.68226485727556
Gradient Boosting Classifier F1-score : 0.696245431744894
```


We print the scores using our metrics.

Then we look at the importance of each parameter the model found.
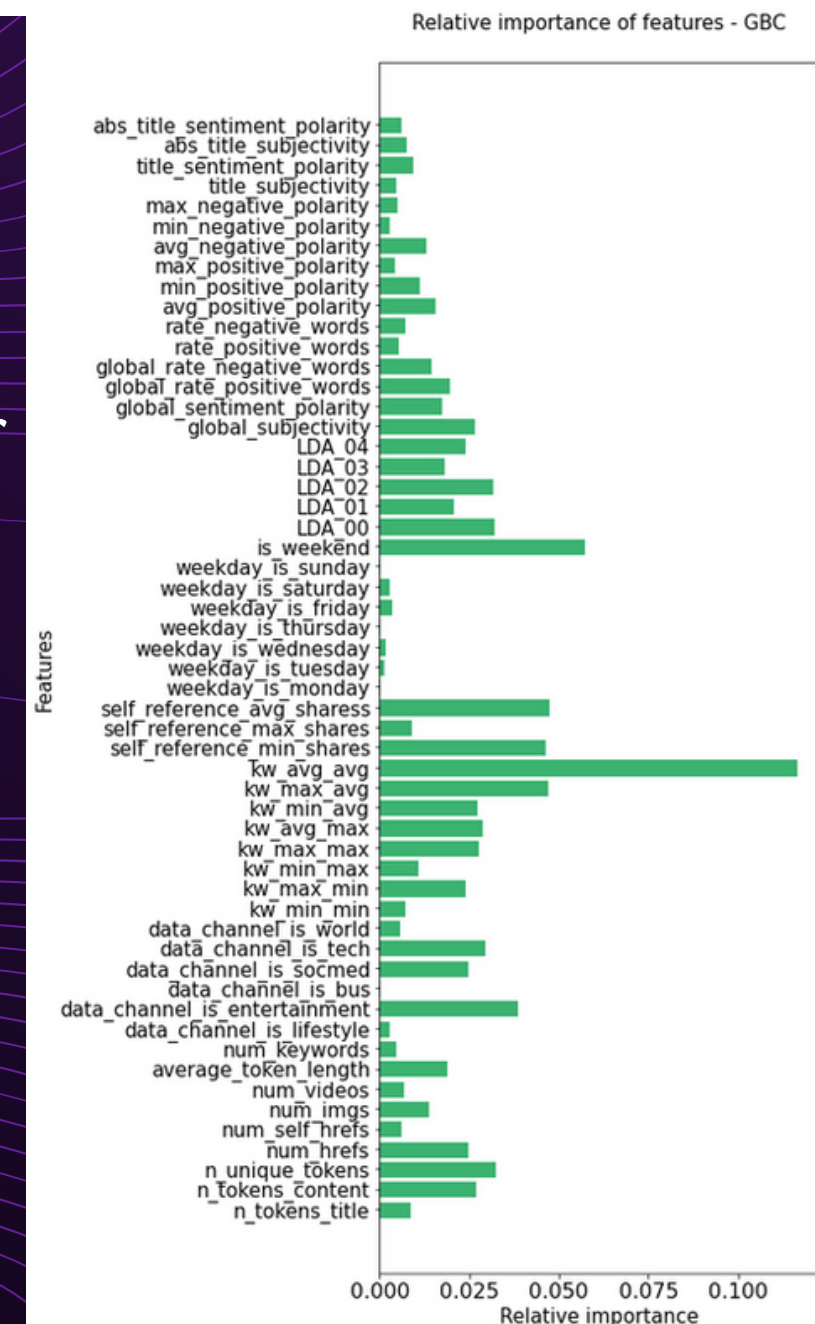
Then we try to find the best maximum depth using the number of estimators we just found

RESULTS

Best accuracy results of our models over scaled and unscaled results
Std stands for "with standard scaling" / Rob stands for "with robust scaling"

Best results of our models over scaled and unscaled results
Std stands for "with standard scaling" / Rob stands for "with robust scaling"

BONUS

```python
df2 = df.iloc[:,:56]
liste = list()
for x in df2['shares']:
    if x >= 3355 :
        liste.append(4)
    else:
        liste.append(2)
df2["popularity2"] = liste
```

As you can see, setting the threshold value to 3355 on a random forest classifier gives us an amazing accuracy of 80.3%. But recall and F1-score are very bad.

```python
clf = RandomForestClassifier(n_estimators = 150, max_depth=15, random_state=0, bootstrap = True, n_jobs=-1)
clf.fit(X2_train, Y2_train)
Y2_pred = clf.predict(X2_test)
```

```
Random Forest Classifier Accuracy : 0.8029377356037957
Random Forest Classifier AUC : 0.5178392258754849
Random Forest Classifier Recall : 0.04218040233614536
Random Forest Classifier Precision : 0.6190476190476191
Random Forest Classifier F1-score : 0.0789793438639125
```

API

— *Our API takes 3 inputs : A title, a text, and a categorie It then uses our model and gives you a prediction whether it will be popular or not.*

⚕ WE TRAINED THE MODEL ON THIS PARAMETERS =>

'n_tokens_title',

'n_tokens_content',

'n_unique_tokens',

'average_token_length',

'n_non_stop_unique_tokens',

'num_hrefs',

'global_subjectivity',

'avg_positive_polarity',

'global_sentiment_polarity', 'data_channel_is_world',

"data_channel_is_tech", "data_channel_is_socmed",

"data_channel_is_bus",

"data_channel_is_entertainment",

"data_channel_is_lifestyle

```python
def tokenizetext(text):
    return word_tokenize(text)
def words(text):
    l = [word for word in word_tokenize(text) if word.isalpha()]
    return l
def unique_words(text):
    return list(set(words(text)))
def rate_uni_words(text):
    uni_words = len(unique_words(text))/len(words(text))
    return uni_words
def avglengthtoken(text):
    w = words(text)
    sum = 0
    for item in w:
        sum+=len(item)
    avglen = sum/len(w)
    return avglen
def n_non_stop_unique_tokens(text):
    uw = unique_words(text)
    n_uw = [item for item in uw if item not in stopwords]
    w = words(text)
    n_w = [item for item in w if item not in stopwords]
    rate_nsut = len(n_uw)/len(n_w)
    return rate_nsut
def numlinks(article):
    return len(BeautifulSoup(article).findAll('link'))
def get_subjectivity(a_text):
    return a_text.sentiment.subjectivity
def get_polarity(a_text):
    return a_text.sentiment.polarity
def word_polarity(words):
    pos_words = []
    ppos_words = [] # polarity of pos words
    neg_words = []
    pneg_words = [] # polarity of negative words
    neu_words = []
```

And we created the functions that allows us to create every values corresponding to this parameters with only 3 inputs which are : a title, a text and a categorie.

API

Hello

Welcome to our website, try to predict the popularity of an article!

Enter the article's title:

Enter the article's text:

What kind of article? World(1) Tech(2) Social(3) Business(4) Entertainement(5) Lifestyle(6)

Submit

CONCLUSION

We found that articles containing :
- Less than 4500 words
- 1 image (or 0)
- 0 video (or 1)
- Between 0 and 40 links
- Posted on week-end
- The categorie social media, tech or lifestyle
- Title between 6 and 16 characters

Tend to be more popular than the others

The accuracy alone isn't an accurate metric as we saw with the case of the threshold of the popularity set to 3355 shares. It gave us an amazing accuracy but very poor recall and F1-score.
We think that a 2 class classification suits the best this dataset but more data could also helps for regression or multiclass classification.

We could get a better accuracy by getting more data but also using deep learning and neural networks. We found on the net someone getting a 98% accuracy using Keras.
A better features selection might bring less biais since we didn't really tested that out.

Popularity might change with people and with time, so we think that the articles liked and popular today might not be the same as the one at the time of the dataset (2013-2015).
A new dataset with actual data might increase the performance of the API.