Online News Popularity

PYTHON FOR
DATA ANALYSIS PROJECT 2022-2023

Audrey DJOUPE PENE
Christopher MUFU
Weize YAN

Summary

01 Introduction

02 Data pre-processing

03 Data exploratory analysis

04 Data modeling

05 Transformation of the model into an API

Introduction :

Dataset description - Online News Popularity :

- Set of features about articles published by Mashable in a period of two years.
- Composed of 39797 instances and 61 attributes

  (58 predictive attributes, 2 unpredictive attributes, 1 goal field)
- The goal field to predict is the number of shares in social networks (popularity)

Study problem :

What are the key factors on the popularity of articles ?

Data pre-processing

Data quality analysis

To start the pre-processing, we started by studying the quality of our dataset in order to have a global view on the transformations to be performed.

Main insights :

- Dataset composed only of numerical values
- No empty values observed in the attributes and target compositions

n_tokens_title	n_tokens_content	n_unique_tokens	num_hrefs	num_self_hrefs	num_imgs	num_videos
12.0	219.0	0.663594	4.0	2.0	1.0	0.0
9.0	255.0	0.604743	3.0	1.0	1.0	0.0
9.0	211.0	0.575130	3.0	1.0	1.0	0.0
9.0	531.0	0.503788	9.0	0.0	1.0	0.0
13.0	1072.0	0.415646	19.0	19.0	20.0	0.0
10.0	370.0	0.559889	2.0	2.0	0.0	0.0
8.0	960.0	0.418163	21.0	20.0	20.0	0.0
12.0	989.0	0.433574	20.0	20.0	20.0	0.0
11.0	97.0	0.670103	2.0	0.0	0.0	0.0
10.0	231.0	0.636364	4.0	1.0	1.0	1.0
9.0	1248.0	0.490050	11.0	0.0	1.0	0.0
10.0	187.0	0.666667	7.0	0.0	1.0	0.0
9.0	274.0	0.609195	18.0	2.0	11.0	0.0
9.0	285.0	0.744186	4.0	2.0	0.0	21.0
8.0	259.0	0.562753	19.0	3.0	9.0	0.0

Data pre-processing

Deleting rows and columns

The second step of our transformation consists in removing the rows and columns that are not relevant in the resolution of our problem in order to improve the performance of our predictive model.

Main deletions :

- Unpredictive attributes removing
- Removal of attributes that have too much correlation between them
- Deleting articles with empty content

```
# Here we drop the two non-preditive (url and timedelta) attributes.
df.drop(columns=['url','timedelta'], axis=1, inplace=True)
df.head()
```

```
# n_tokens_content represents Number of words in the conten
# However its minimum value to be 0 means that there are ar
# Such records should be dropped as their related attribute

# find number of rows that contain 0 for n_tokens_content
num_of_nowords=df[df['n_tokens_content']==0].index
print('Number of news with no words',num_of_nowords.size)

# Drop these items or rows with n_tokens_content = 0
df = df[df['n_tokens_content'] != 0]
```

```
Number of news with no words 1181
```

Data pre-processing

Attribute transfomations

We then intervened in the transformation of the attributes in order to facilitate our analysis tasks and also to facilitate the prediction potential of our model.

Main transformations :

- Grouping of binary attributes
- Transformation of the name of attributes
- Transformation of the target shares

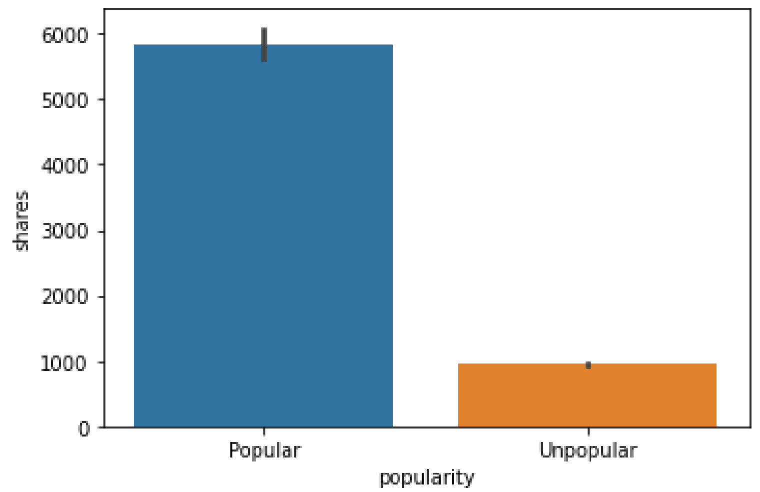
```
publishdayMerge=df[['weekday_is_monday','weekday_is_tuesday','
                   'weekday_is_thursday', 'weekday_is_frida
publish_arr=[]
for r in list(range(publishdayMerge.shape[0])):
    for c in list(range(publishdayMerge.shape[1])):
        if ((c==0) and (publishdayMerge.iloc[r,c])==1):
            publish_arr.append('Monday')
        elif ((c==1) and (publishdayMerge.iloc[r,c])==1):
            publish_arr.append('Tueday')
        elif ((c==2) and (publishdayMerge.iloc[r,c])==1):
            publish_arr.append('Wednesday')
```

```
0         593
1         711
2        1500
3        1200
4         505
         ...
38458    1800
38459    1900
38460    1900
38461    1100
38462    1300
Name: shares,
```

=>

```
sns.barplot(x='popularity', y="shares", data = df)
```

```
<AxesSubplot:xlabel='popularity', ylabel='shares'>
```



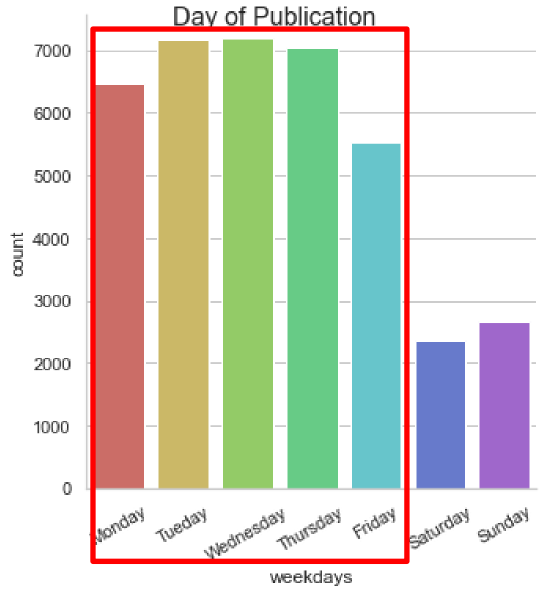
Data exploratory analysis

Global analysis

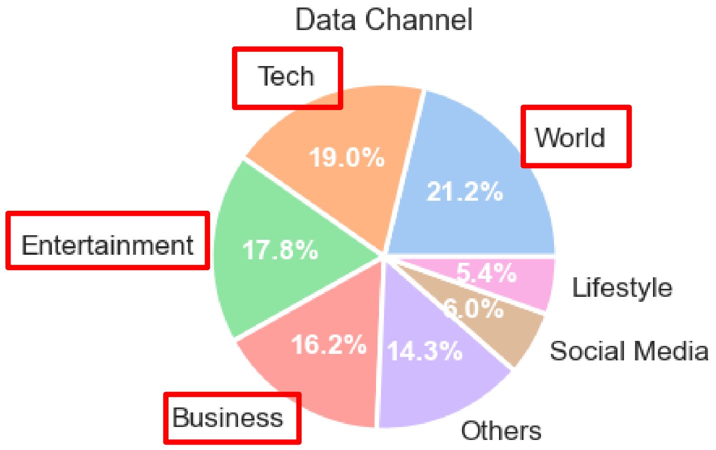
In order to get a better understading of our dataset to solve our problematic, we proceeded to a global study including an analysis of our categorical attributes as well as an illustration of the correlations between them.

Main analysis :

- Understanding of the distribution of articles according to the day of the week and the theme.
- No linear correlation between target share and other numerical attributes.



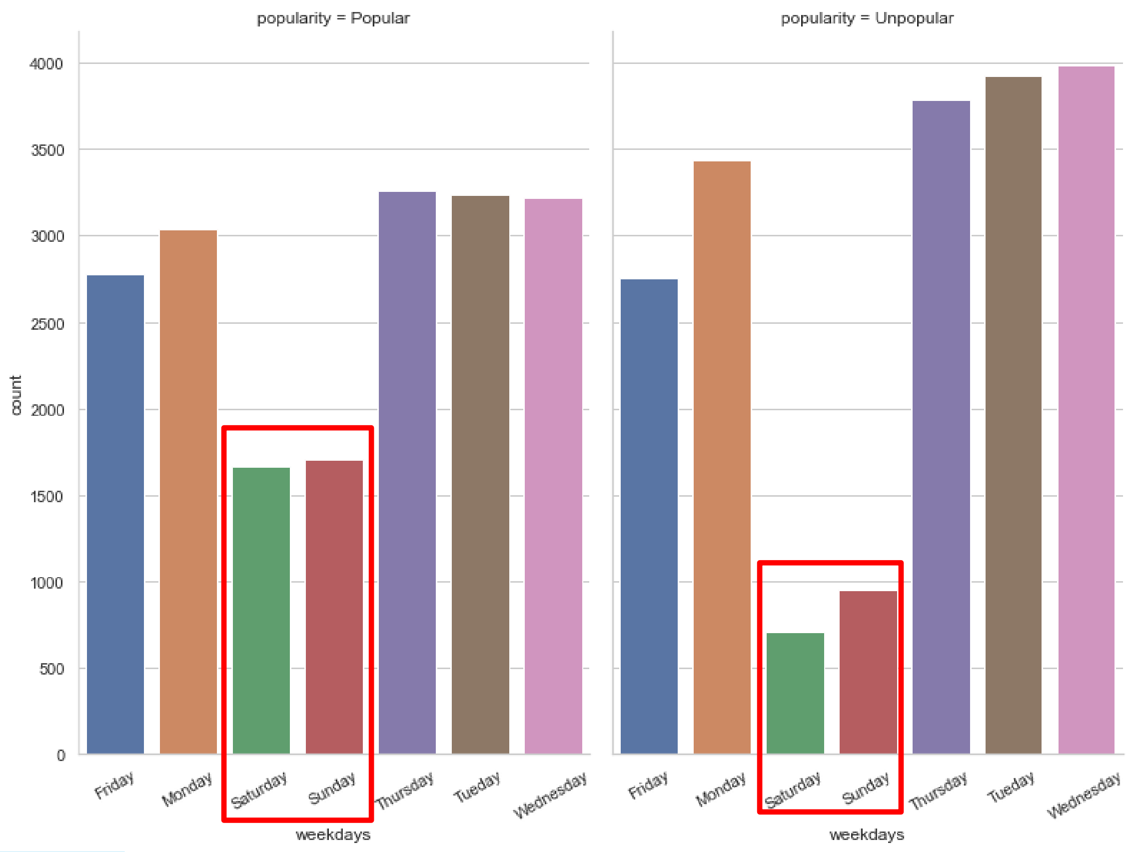
Most of the articles was not released on weekends.



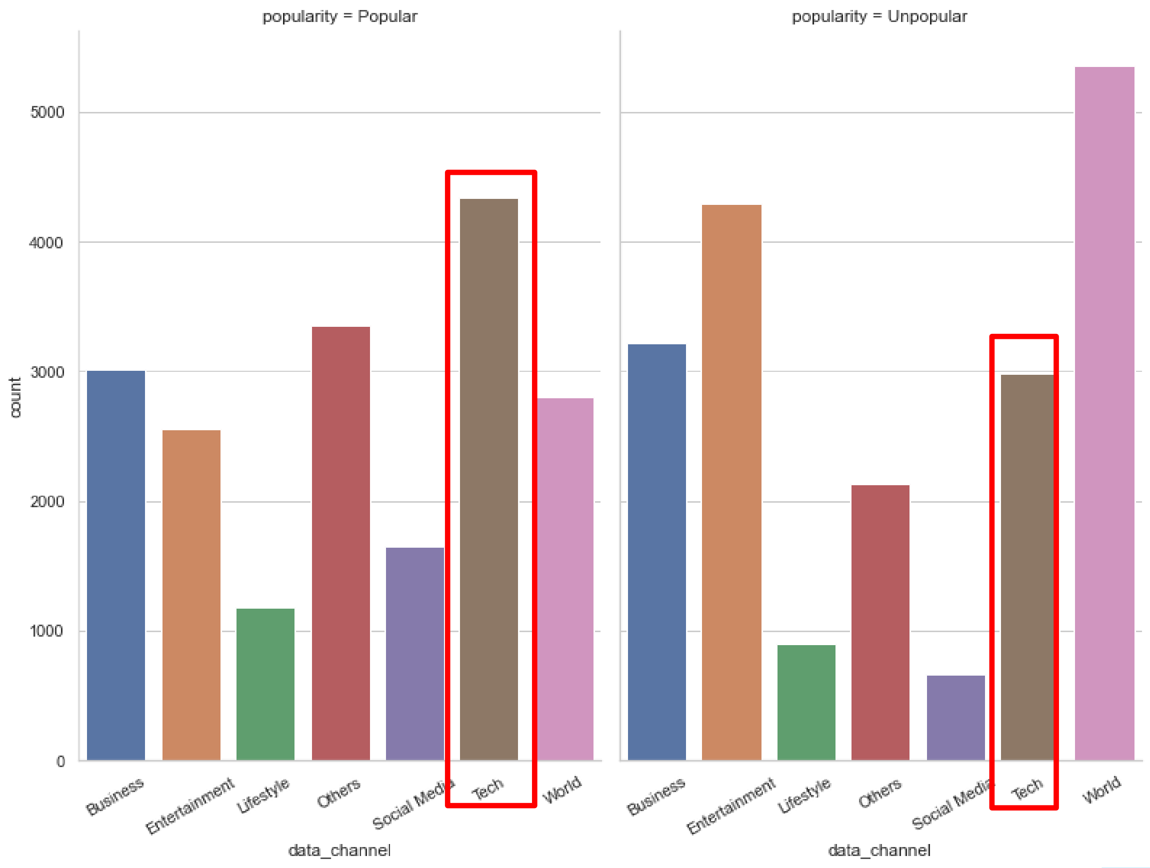
People are interested in articles about World, Tech, Entertainment and Business.

Data exploratory analysis

Global analysis



Popular articles are published more often on weekends than unpopular articles.



Popular articles most like the Technology topic, as opposed to the unpopular articles.

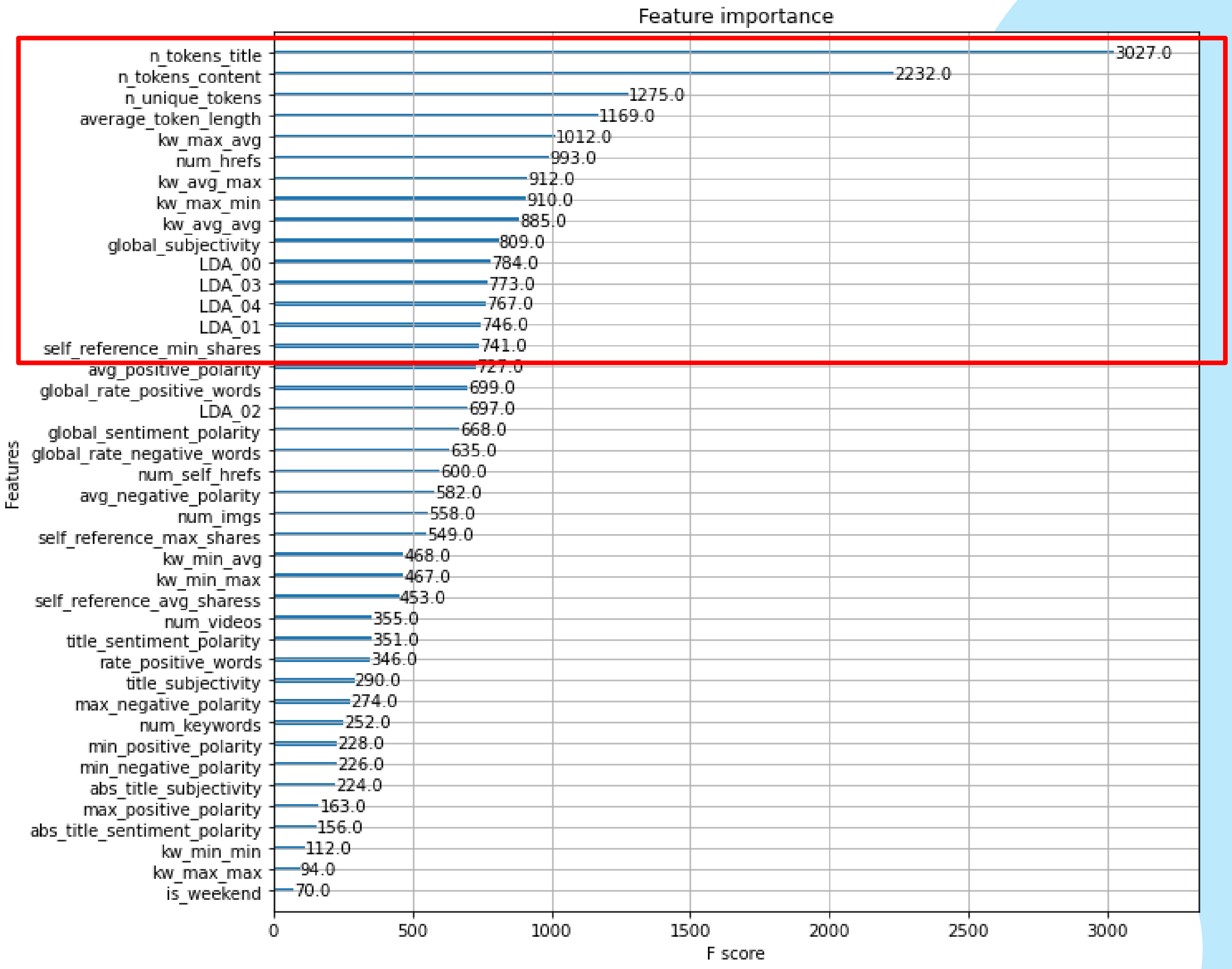
Data exploratory analysis

Rank of important variables

To optimize the design of our model and reduce the possibilities of overfitting, we have started a ranking of the most important variables in the influence of the target share prediction.

Main analysis :

- Selection of the attributes that will compose the sample for predicting the popularity of an article.



Data exploratory analysis

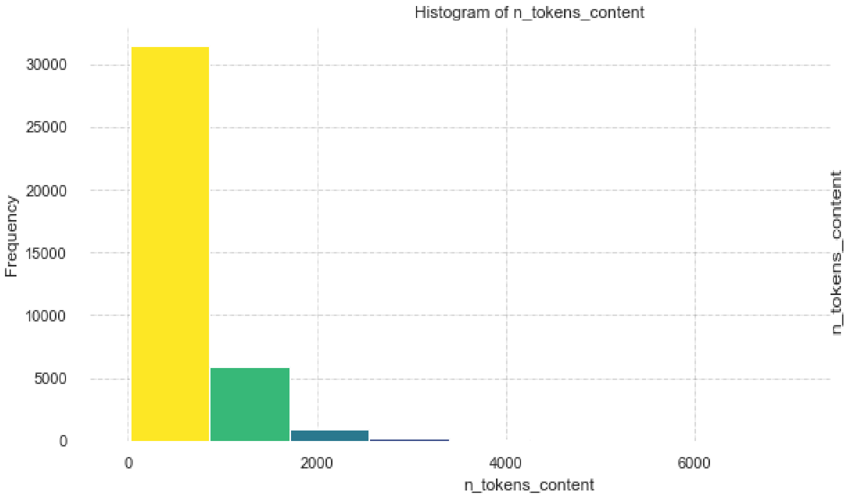
Analysis of important attributes

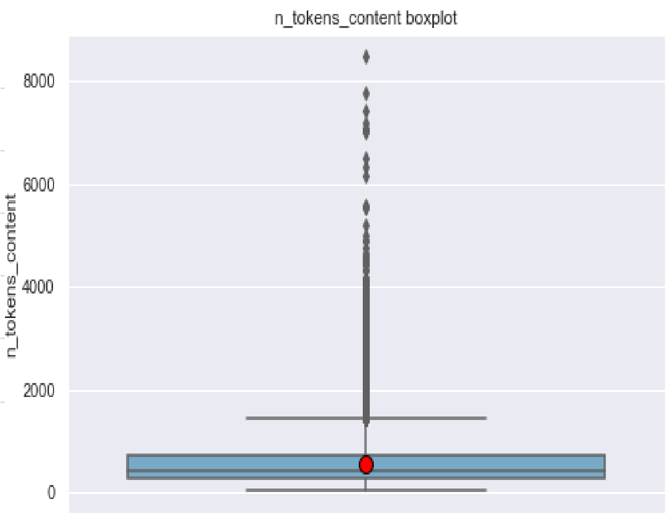
After having identified the important attributes, we proceed to their analysis in a univariate way in order to obtain all the knowledge necessary to build our predictive model.

Main analysis :

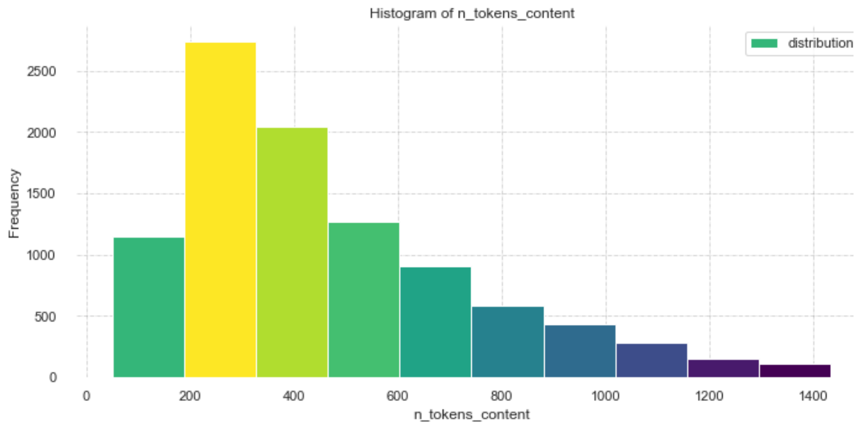
- Observation of the difference between the important attributes with and without outliers.

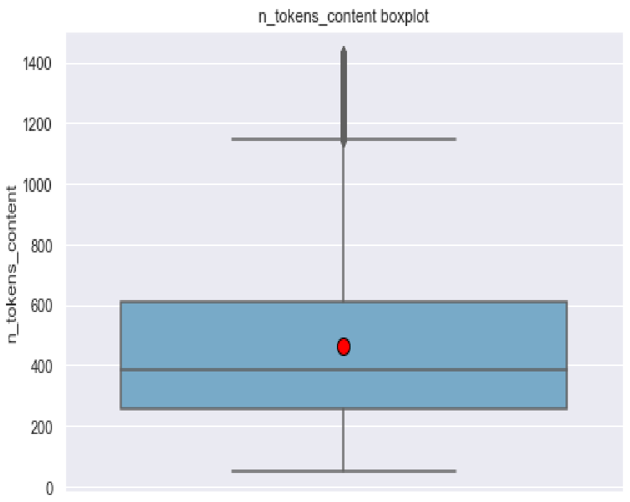
With outliers :





Without outliers :





Data modeling

Scaling dataset :

Before scaling the data, we extract the dataset with the 16 most important features.

In the analysis of the data done previously, we noticed that the distribution of the data was not normal. For this, we had to normalize our data set.
We used the RobustScaler method because it is a method that is not vulnerable to outliers.

```
# Shapiro-Wilk Test
from scipy.stats import shapiro

#seed(1)
# normality test
stat, p = shapiro(df["shares"]) # shapiro(df_target)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Sample looks Gaussian (fail to reject H0)')
else:
    print('Sample does not look Gaussian (reject H0)')
```

```
Statistics=0.154, p=0.000
Sample does not look Gaussian (reject H0)
```

```
# #applying log transformation
for col in X.iloc[:,:-1].columns:
    temp = X[X[col] == 0]
    # only apply to non-zero features
    if temp.shape[0] == 0:
        X[col] = np.log(X[col])
        print (col)
```

```
from sklearn.preprocessing import StandardScaler, RobustScaler
scaler = RobustScaler()

X = scaler.fit_transform(X)
```

Data modeling

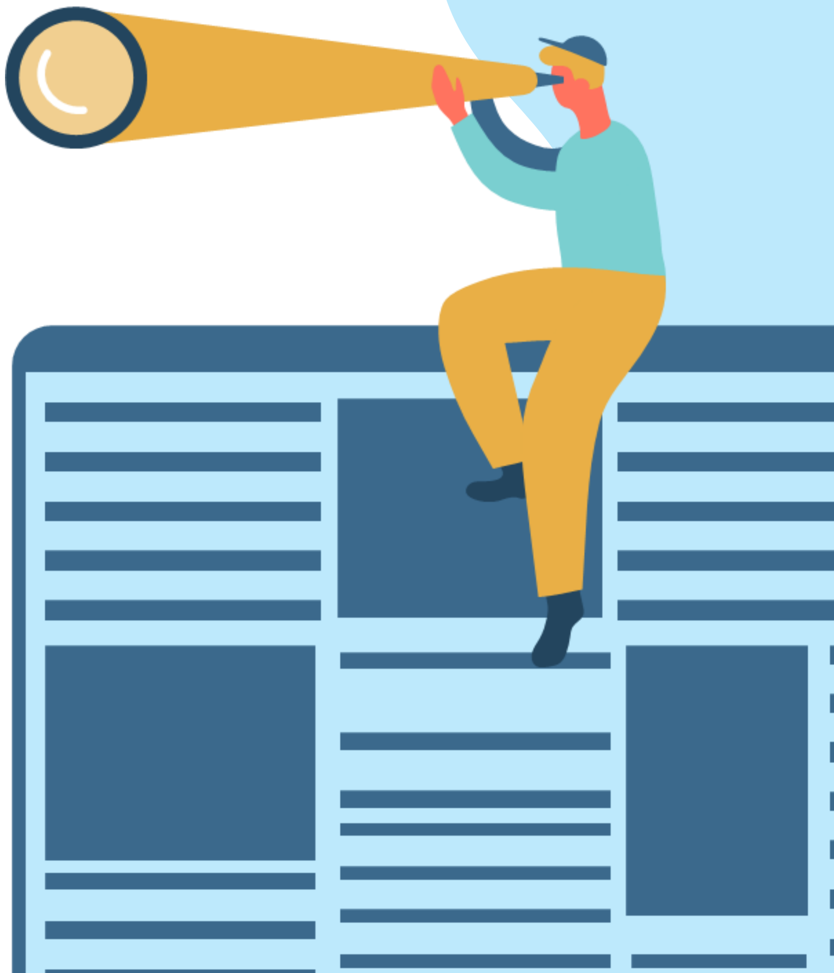
Implementation of models

We started by creating our data sets for training and testing.

```python
X = data_used.drop("popularity", axis=1)
y = data_used["popularity"]
```

```python
from sklearn.model_selection import train_test_split, GridSearchCV

X_train, X_test, y_train, y_test = train_test_split(X, y_encode, test_size=0.3)
```



Data modeling

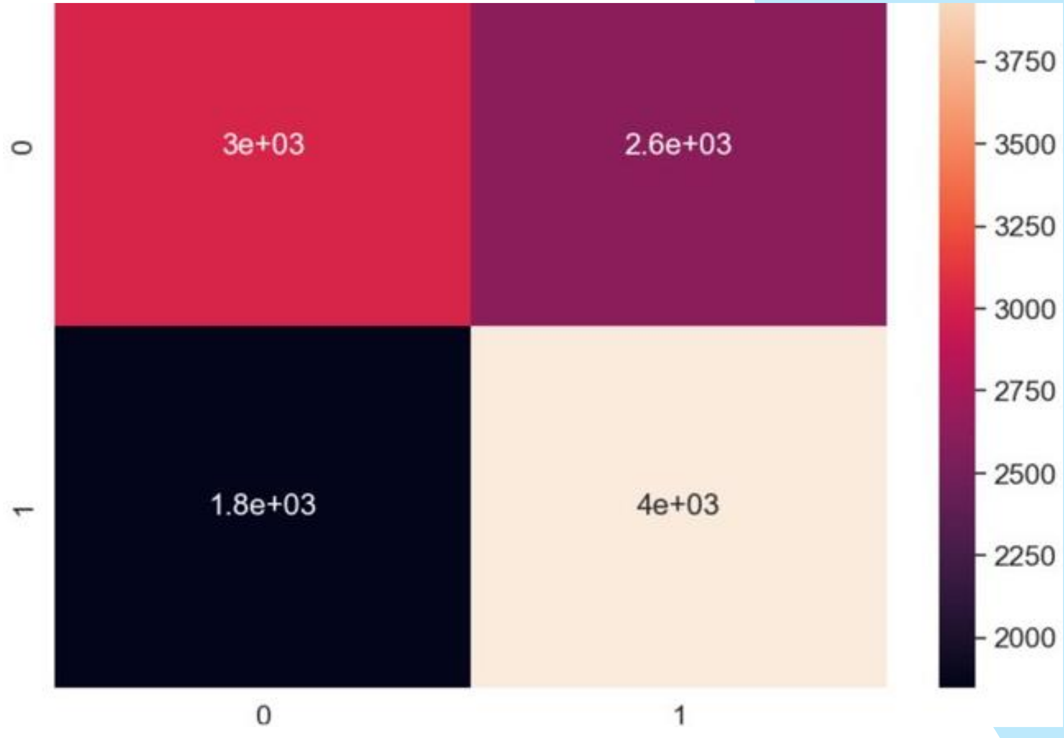
Choose the best model et make predictions

Random Forest Classifier

This model is typically for the classification or regression task. It is robust to ouliers and non-linear data. We first imported our model and then trained it on our data and finally made predictions.

Accuracy score: 61.27%

F1_score equal to 64.4%.



Data modeling

Choose the best model et make predictions

Random Forest Classifier

In order to overcome bad predictions, we changed the hyper parameters of our model using GridSearchCV to find the best possible combination of hyper-parameters at which the model achieved the highest accuracy. The different parameters we have considered here are :

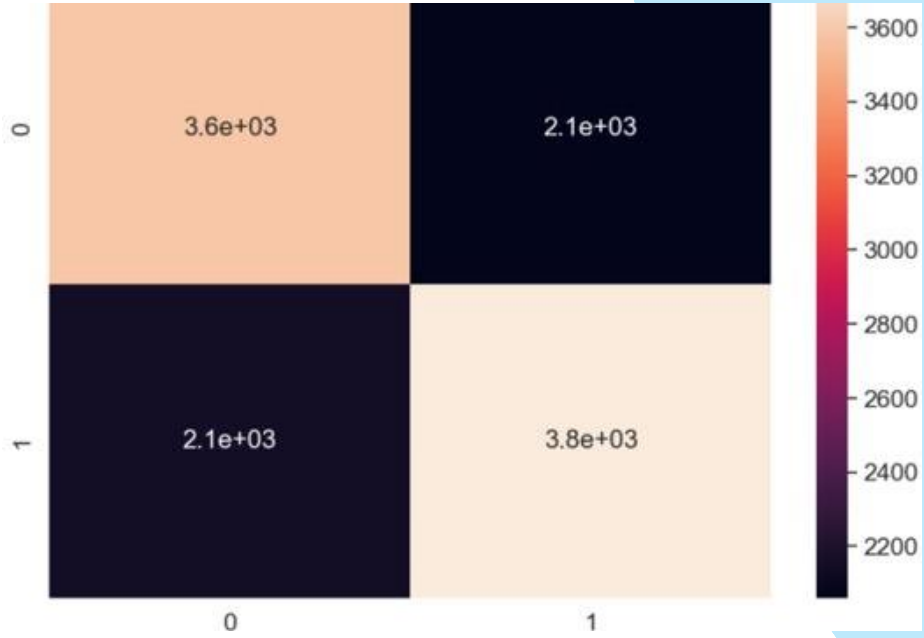
N_estimators: to control the number of trees inside the classifier

Max_depth: the depth of the trees which is an important parameter to increase the accuracy of a model

Accuracy : 63.58%

F1_score is 64.1%.

The best values of the hyperparameters are : max_depth : 16 and n_estimators : 256



Data modeling

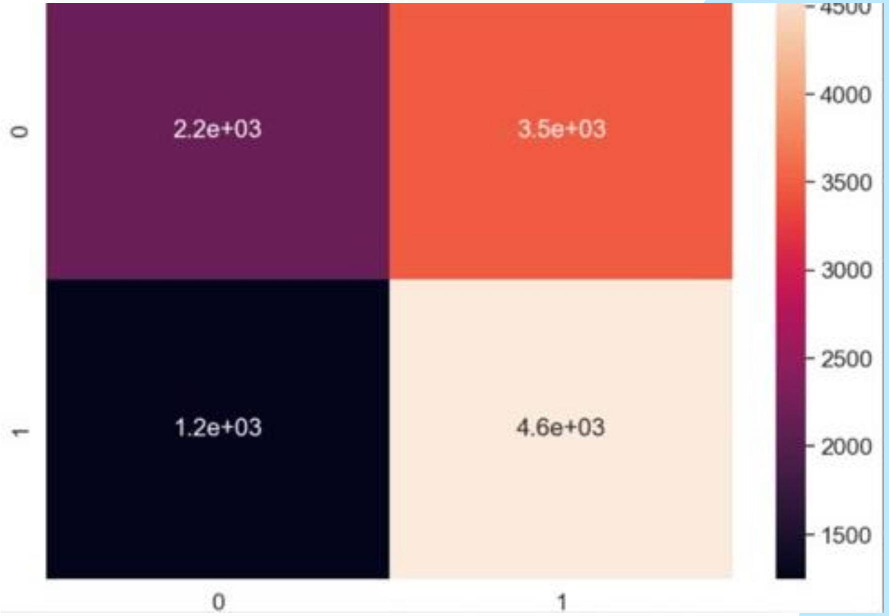
Choose the best model et make predictions

AdaBoost Classifier

Ada-boost or Adaptive Boosting is one of ensemble boosting classifier. It combines multiple classifiers to increase the accuracy of classifiers. .

Accuracy score: 59%

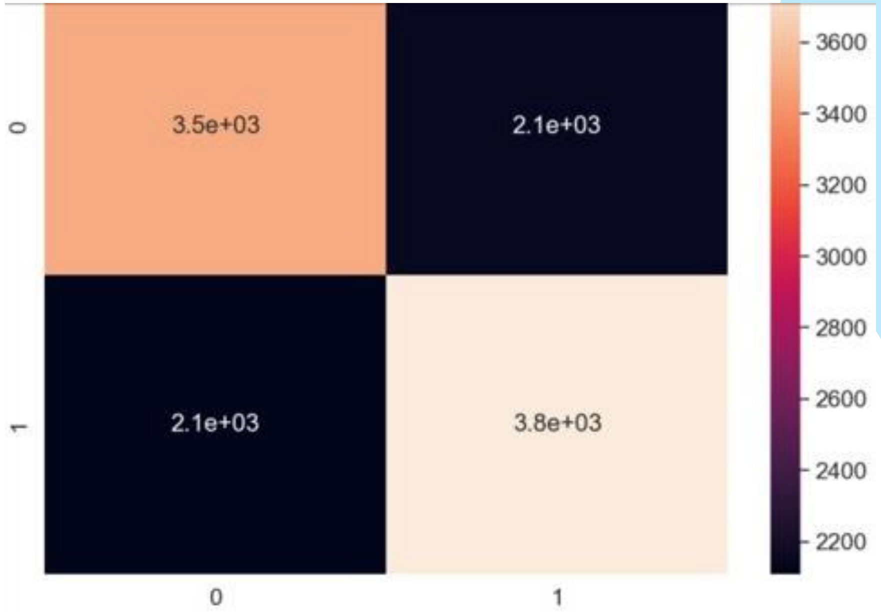
F1_score equal to 66.3%.



After changing hyper-parameters :

Accuracy score: 63.15%

F1_score equal to 64%.



N_estimators : 49

Data modeling

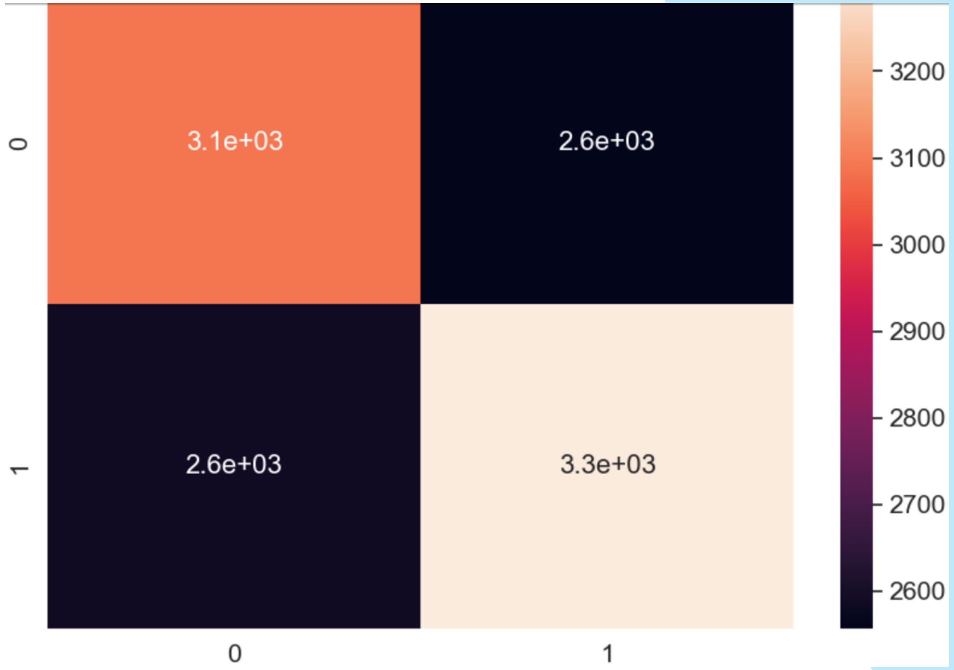
Choose the best model et make predictions

Decision Trees Classifier

DTs are a non-parametric supervised learning method used for classification and regression. Same to the previous models, we first imported our model and then trained it on our data and finally made predictions.

Accuracy : 55.42%

F1_score is 56.2%.



Data modeling

Choose the best model et make predictions

Decision Trees Classifier

For the change of hyperparameters, we chose:

Max_depth: best tree depth

Min_sample_split : minimum number of

samples required to split an internal code

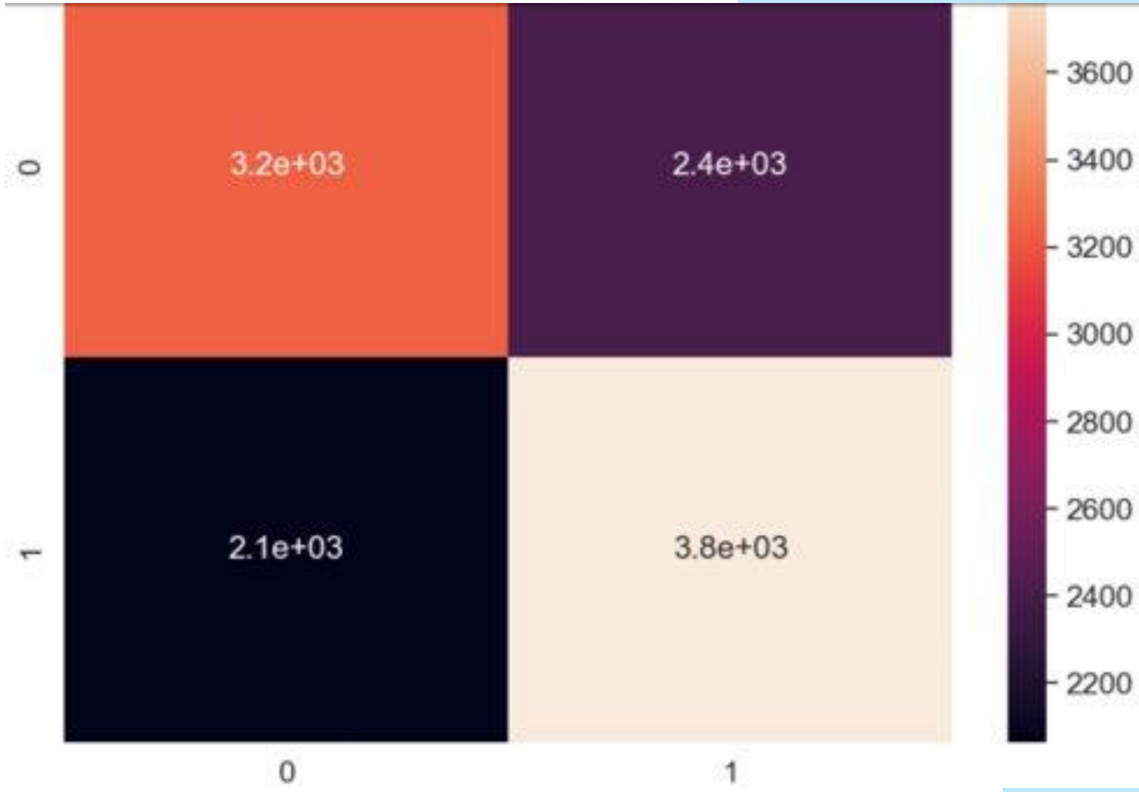
Min_sample_split : minimum number of leaf samples

Criterion : best criterion used to split particular nodes to make decisions

Accuracy : 61.26%

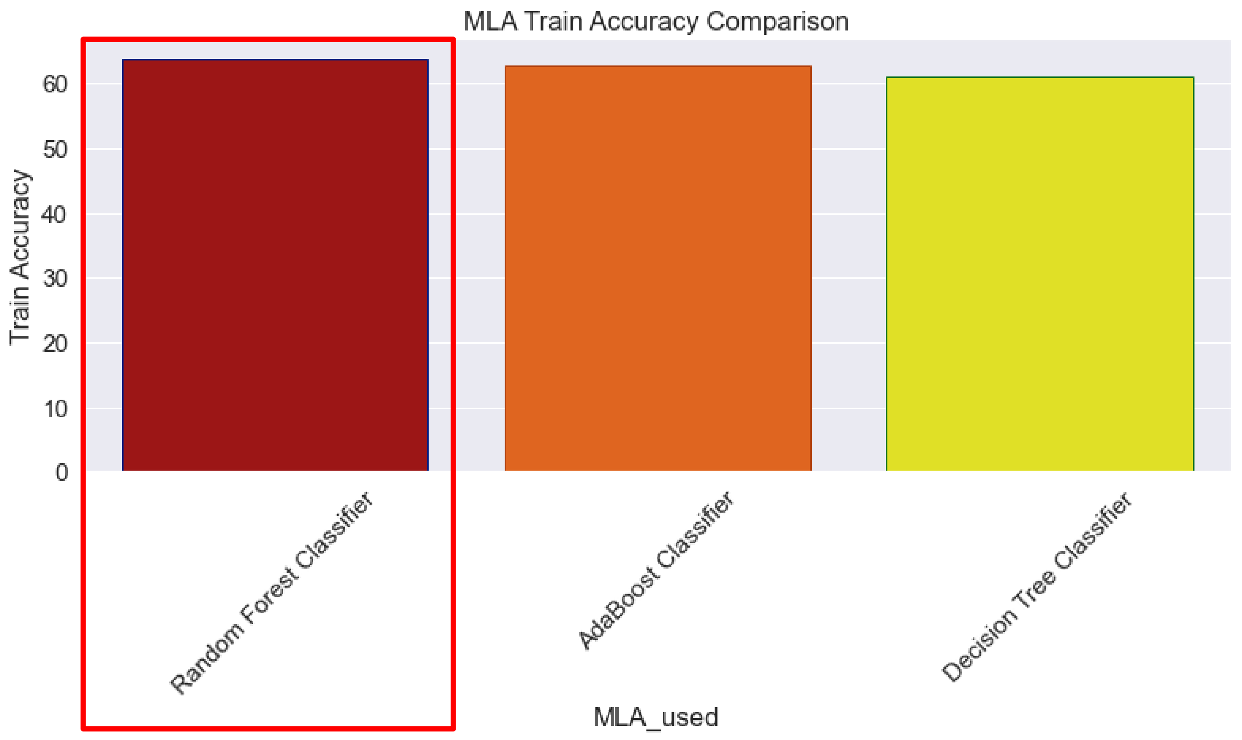
F1 score : 63.1%

The best fitting hyperparameters are : **criterion: 'entropy', max_depth: 5, min_samples_leaf: 1, min_samples_split: 2**



Data modeling

Comparaison of models :



To compare the accuracy of these different models, we have made a plot to show their difference. All three of them have similar accuracy rates, as can be seen in the graphic. But Random Forest is relatively better, so we have chosen to use Random Forest Classifier to make predictions.

Transformation of the model into an API

Process :

- Registration of the best model and the prediction sample
- Creation of the input of the API request
- Creation of the output of the API request
- Realization of the display process of and operation of API