ONLINE NEWS POPULARITY

Python For Data Analysis Project

Barbier Lucas
Amenzou Brahim

DIA 2

INTRODUCTION

The Topic of this project was that we have to analyze the dataset by making data pre-processing, data visualization and modeling.

Our Dataset was Online News Popularity a dataset where we want to predict the numbers of shares of an article published on mashable

PROGRAM OF OUR WORK

To do that we will :

01
Explore our dataset

02
Data Cleaning

03
Data Visualization

04
Standardization

05
Modeling

06
Conclusion

FIRST PART

Data Exploration

Data exploration

- First step we just use the fonction shape() and columns() to see how many columns and row we have and theirs names :

```
df.columns
```

```
Index(['n_tokens_title', 'n_tokens_content', 'n_unique_tokens',
       'n_non_stop_words', 'n_non_stop_unique_tokens', 'num_hrefs',
```

```
df.shape
```

```
(39644, 61)
```

- After we used describe() to know everything about our dataset

```
df.describe()
```

	timedelta	n_tokens_title	n_tokens_content
count	39644.000000	39644.000000	39644.000000
mean	354.530471	10.398749	546.514731
std	214.163767	2.114037	471.107508
min	8.000000	2.000000	0.000000
25%	164.000000	9.000000	246.000000
50%	339.000000	10.000000	409.000000
75%	542.000000	12.000000	716.000000
max	731.000000	23.000000	8474.000000

SECOND PART

Data Cleaning

Data cleaning

- First step we just use the fonction info() to see what types are our data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39644 entries, 0 to 39643
Data columns (total 61 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   url                             39644 non-null  object
 1    timedelta                      39644 non-null  float64
 2    n tokens title                 39644 non-null  float64
```

- As we can that timedelta and url are useless for our dataset so wee can drop thems by using drop()

```
df.drop(columns=['timedelta','url'], inplace=True)
```

- So now our dataset have 59 columns and 39644 rows

```
df.shape
```

```
(39644, 59)
```

Data cleaning

- After using a fonction to know if there is some empty article we can see that there is 1181 article who are empty so let’s just erase them to not influced our futur modele

```
df[df['n_tokens_content']==0]
```

	n_tokens_title	n_tokens_content	n_unique_tokens	n
893	10.0	0.0	0.0	
917	8.0	0.0	0.0	
1062	12.0	0.0	0.0	
1121	10.0	0.0	0.0	
1312	14.0	0.0	0.0	
...	...	...	...	
39598	11.0	0.0	0.0	
39601	12.0	0.0	0.0	
39613	12.0	0.0	0.0	
39615	11.0	0.0	0.0	
39616	15.0	0.0	0.0	

1181 rows × 59 columns

```
df = df[df['n_tokens_content']!=0]
df = df.reset_index(drop=True)
```

- So now our dataset have 59 columns and 38463 rows

```
df.shape
```

```
(38463, 59)
```

Data cleaning

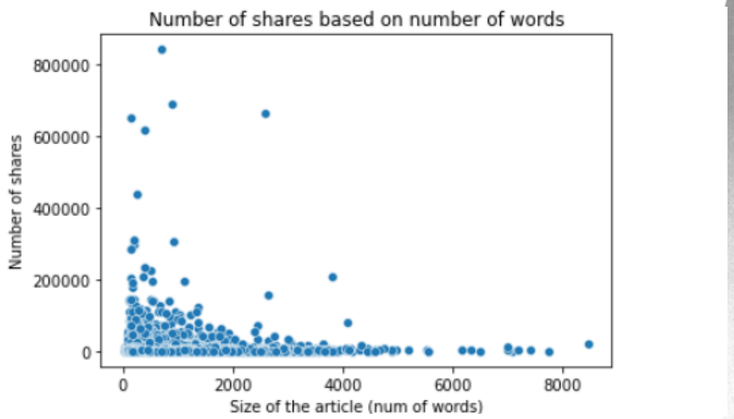
- All columns that are discribing day when article are published are erased in one columns named publish day

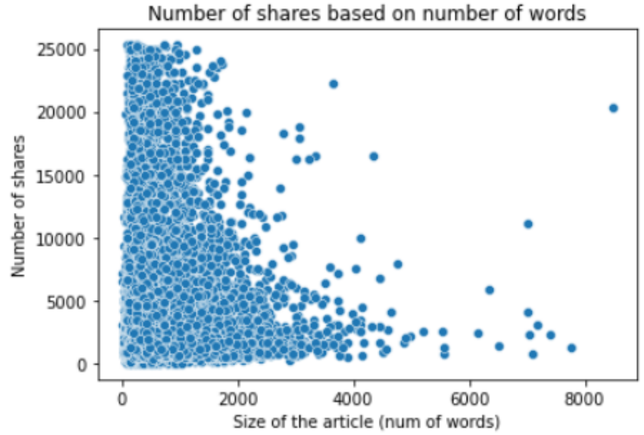
- All columns that are discribing the subject of the article are erased in one columns named Topic

Data cleaning

- We can find that there is a lot of outliers values in the dataset's target "shares". So we just take the heart of the values of the dateset with all the values between 25 and 90 percent of the toal amount of shares value

- Then we create a new target column named "categorize" by separating the level of success of the article.






```
quantiles = df.shares.quantile([0.3, 0.7]).tolist()
print(quantiles)
labels = ['flop', 'mid', 'top']

# Replace the values in the dataset
for i in range(len(labels)):
    if i == 0:
        df.loc[df.shares <= quantiles[i], 'categorize'] = labels[i]
    elif i == len(labels) - 1:
        df.loc[df.shares > quantiles[i-1], 'categorize'] = labels[i]
    else:
        df.loc[(df.shares > quantiles[i-1]) & (df.shares <= quantiles[i]), 'categorize'] = labels[i]

# Display unique values in the 'categorize' column for each category
for i, label in enumerate(labels):
    print(i, ': ', label)
    print(df.loc[df.categorize == label, 'categorize'].unique())

df.categorize.unique()
```

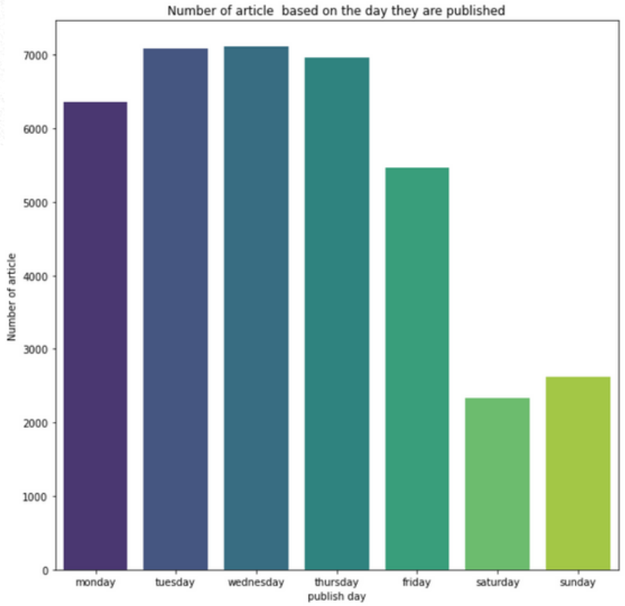
```
[1000.0, 2200.0]
0 :  flop
['flop']
1 :  mid
['mid']
2 :  top
['top']
```

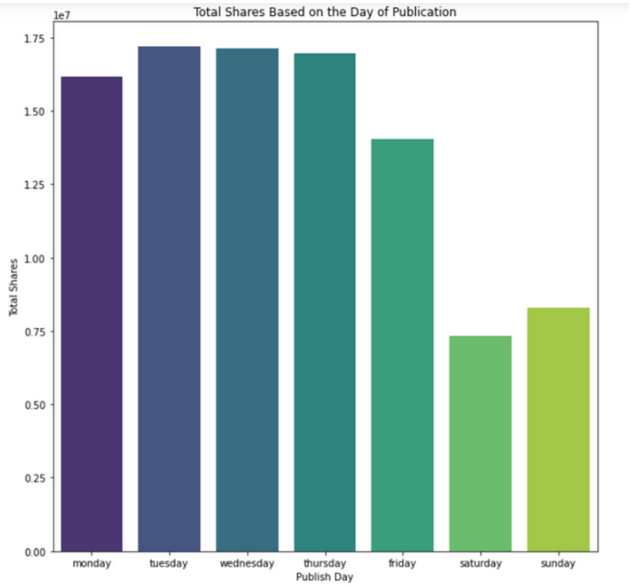
THIRD PART

Data visualization

Data visualization

First we wanted to see the number of articles published per day to see if this number changed from day to day but also the average number of shares in fnction of days

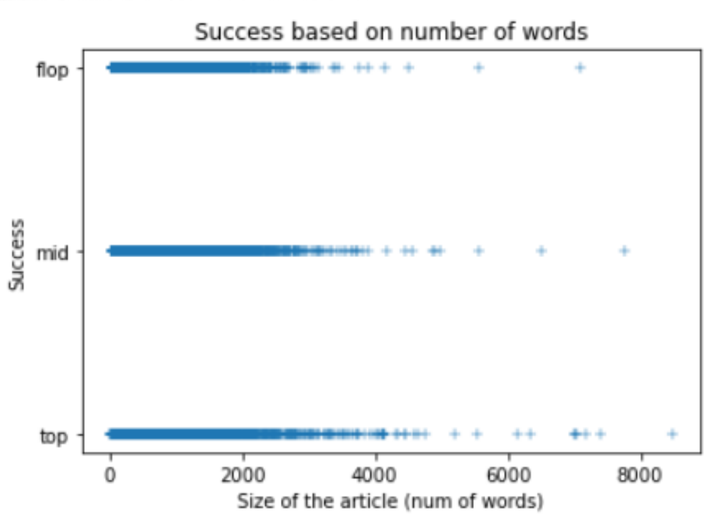


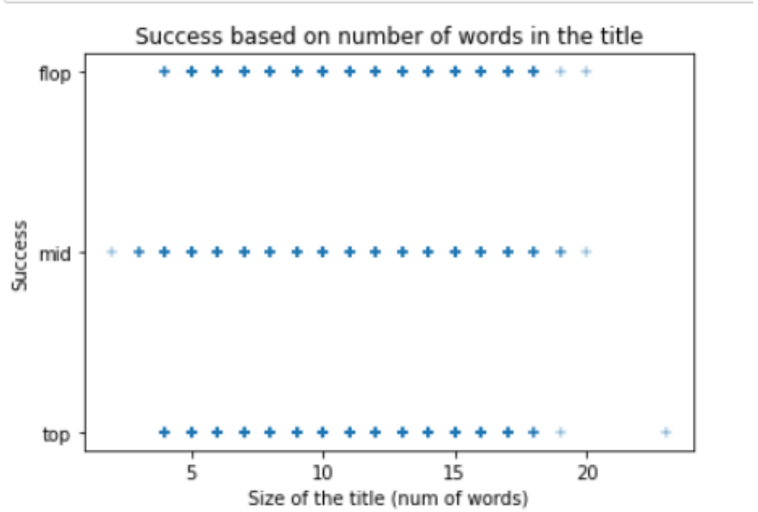


- As we can see it’s not representative enough because there is more articles published at the begining of the week so we can ponderate de number of article by day

Data visualization

Then we wanted to find a correlation between the success of the article and the number of words in the article :

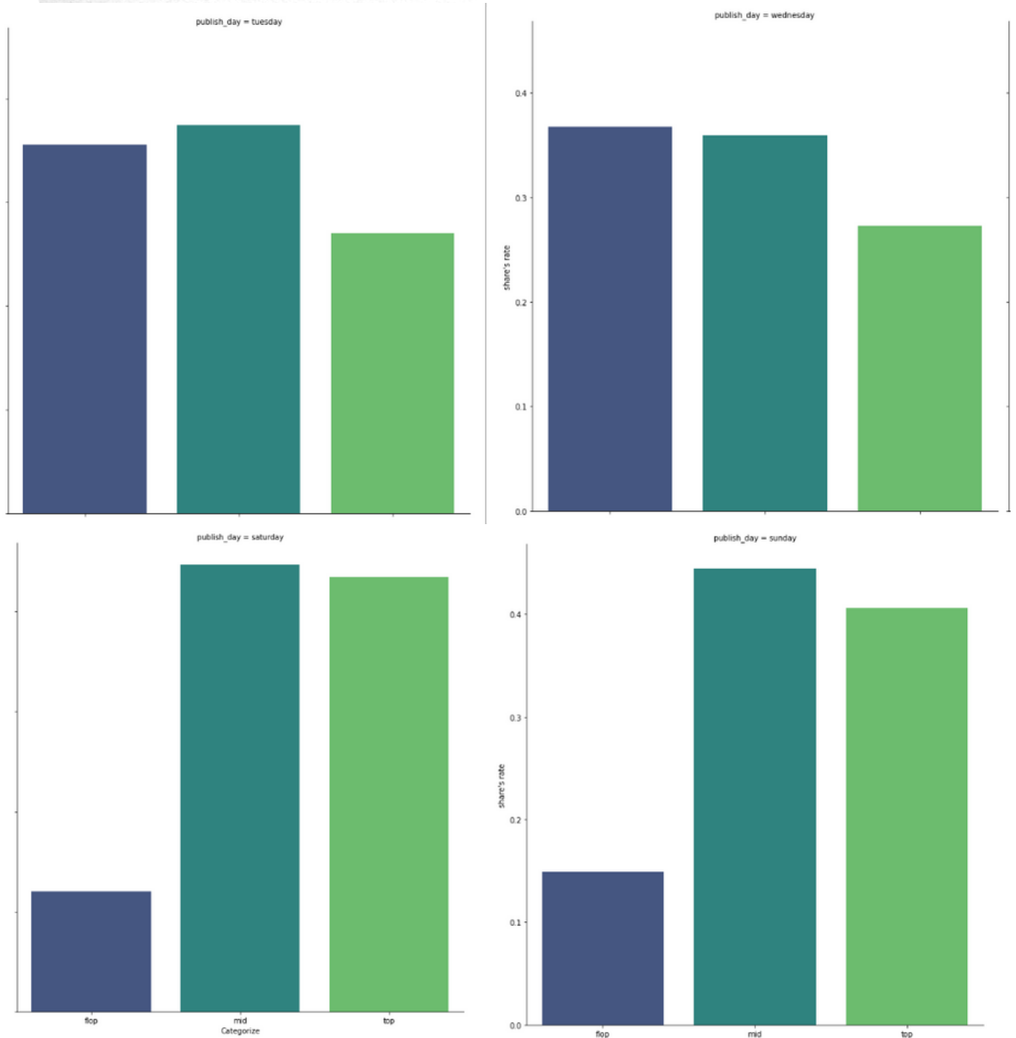




- We cannot predict the success based on the number of words in the article or in the title

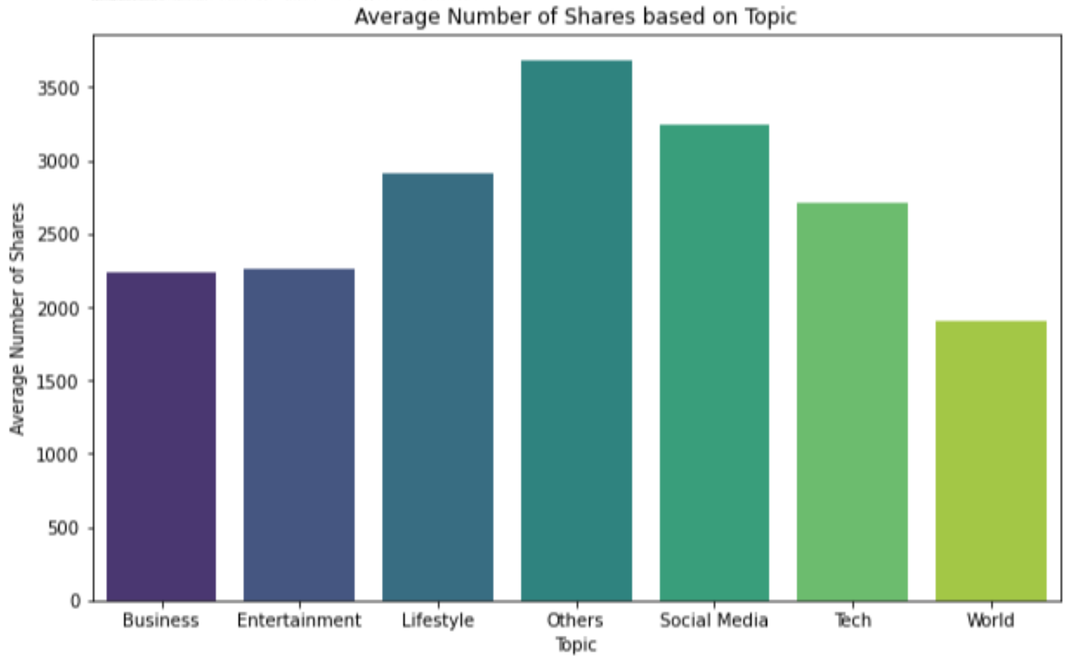
Data visualization

we have ponderate de number of article by day so now let’s see what we have :



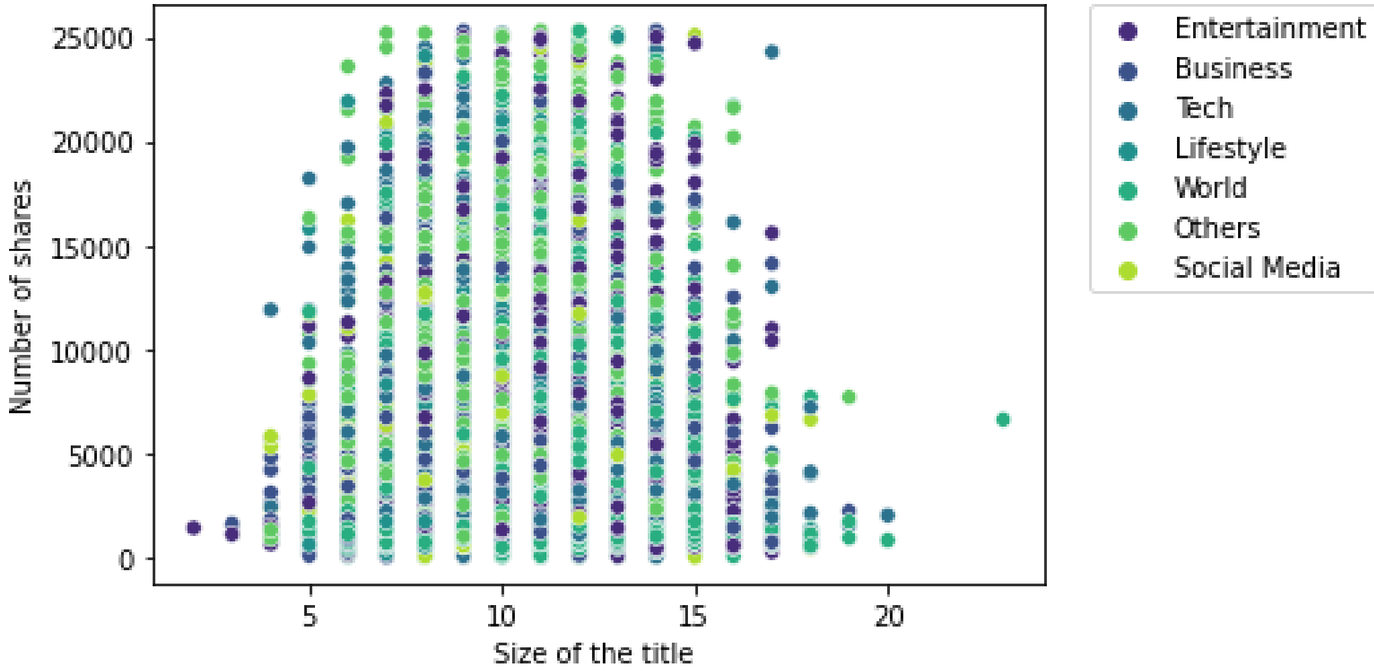
- The article published during the weekend a more likely to not be flop
- The worst day to publish an article is wednesday and the best one is saturday
- The most often category that appears the highest rate is mid

TOPIC OF THE ARTICLE

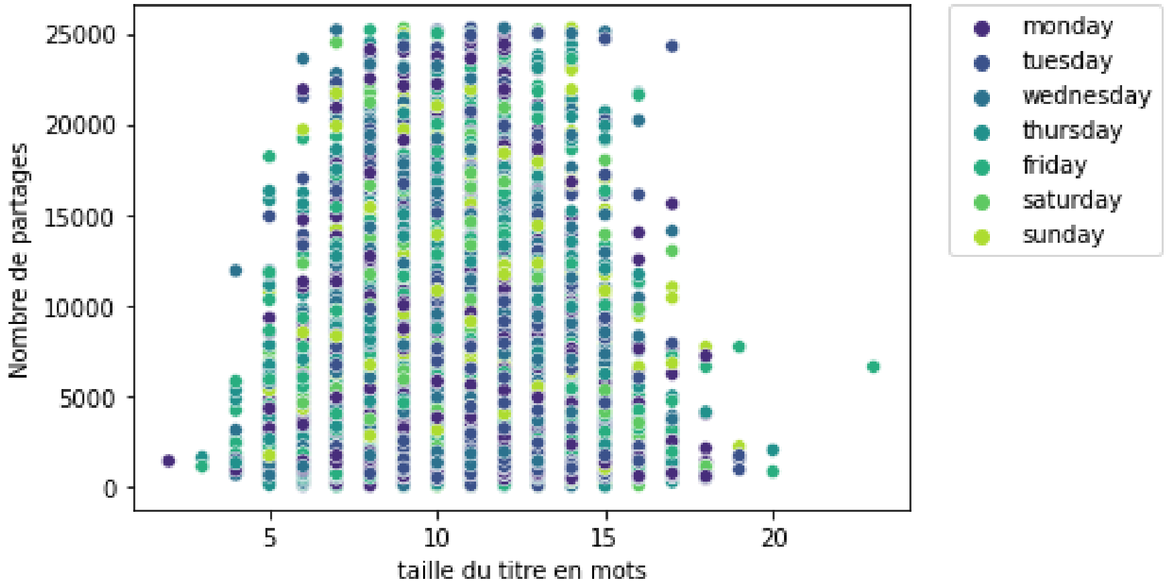


- In this graph we cannot identify a majority define by the size of the title or the TOPIC

- The topic with the highest mean of shares is “Others” followed by “Social Media”



Data Visualization



- We also cannot conclude on the day they were published combine with the size of the article .

Data Visualization Conclusion

So according with all plot we can conclude that if you want to make an article with a lot of share you have to :

- Publish it during the weekend
- It must contain less than 3000 words
- Title must contain between 8 and 15 words
- It must talk about Social Media or Various subject

FOURTH PART

Standardization

Standardization

Now we have to prepare our data for all model :

First step we just put “shares” column and “categorize” columns to our target .

```
x = df.drop(['shares', 'categorize'], axis=1)
y = df[['shares', 'categorize']]
```

We separate our set in train and test set but we have to know wich columns we can’t stadardize and we choosed publish_day, channel and lda because they are not numerical caracteristic.

```
# Identification des colonnes à ne pas standardiser
publish_day = x.columns.values[list(x.columns).index('weekday_is_monday'):list(x.columns).index('weekday_is_sunday')+1].tolist()
channel = x.columns.values[list(x.columns).index('data_channel_is_lifestyle'):list(x.columns).index('data_channel_is_world')+1].tolist()
lda = x.columns.values[list(x.columns).index('LDA_00'):list(x.columns).index('LDA_04')+1].tolist()
columns_ns = publish_day + channel + lda + ['is_weekend']
```

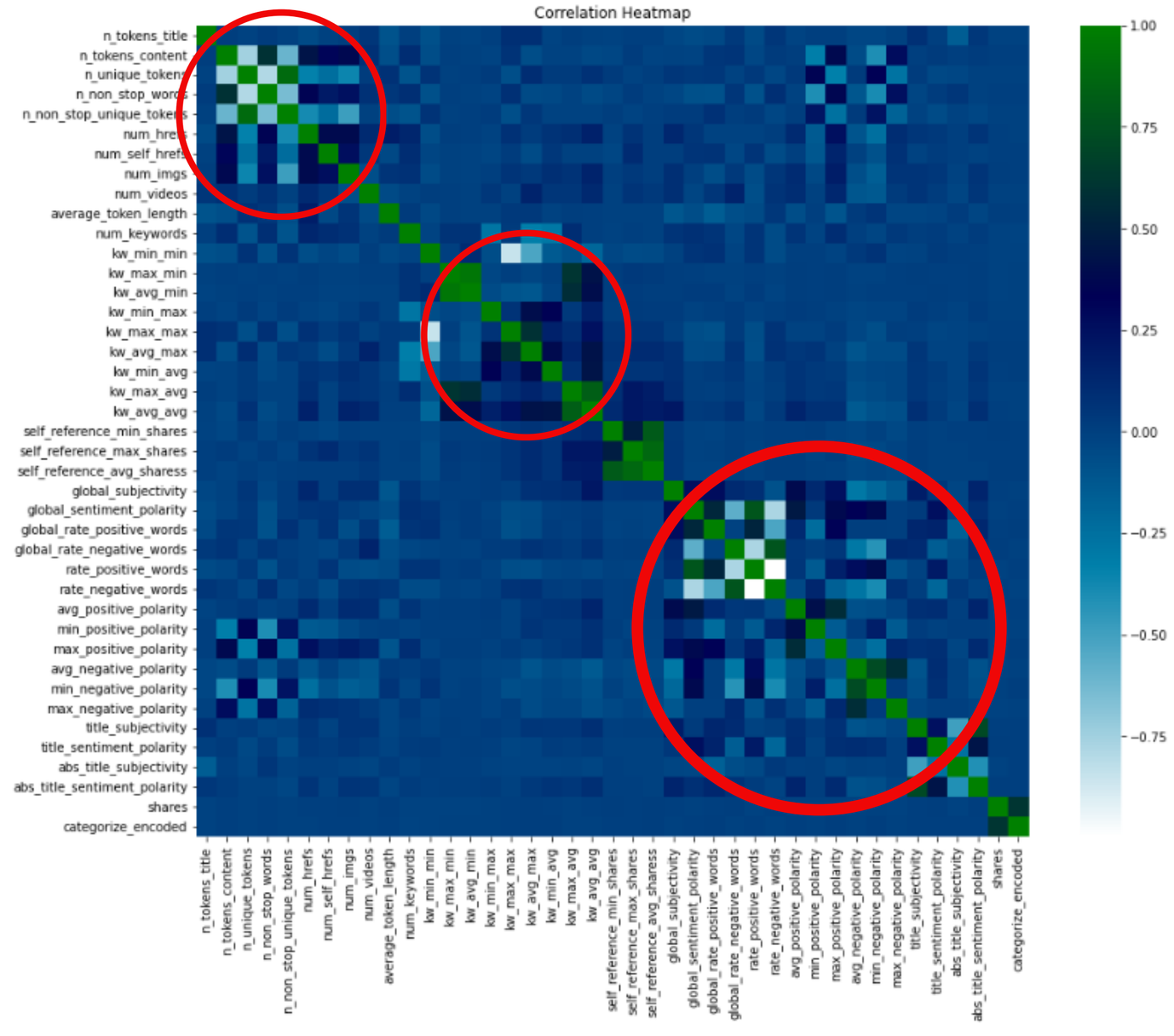
We standardized our data using StandarScaler() and created a dataframe with all columns

```
# Standardisation des caractéristiques numériques
scaler = StandardScaler()
feature_column = x.drop(columns=columns_ns).columns
x_train_scaled = scaler.fit_transform(X_train[feature_column])
x_test_scaled = scaler.transform(X_test[feature_column])

# Création de DataFrame avec des colonnes correspondantes
x_train_df = pd.DataFrame(x_train_scaled, columns=feature_column, index=X_train.index)
x_test_df = pd.DataFrame(x_test_scaled, columns=feature_column, index=X_test.index)
```

finally we encode the columns categorize and concatenate the training dataframes

```
# Encodage de la colonne 'categorize'
label_encoder = LabelEncoder()
y_train['categorize_encoded'] = label_encoder.fit_transform(y_train['categorize'])
y_test['categorize_encoded'] = label_encoder.transform(y_test['categorize'])

# Concaténation des DataFrames d'entraînement
train_data = pd.concat([x_train_df, y_train.reset_index(drop=True)], axis=1)
```

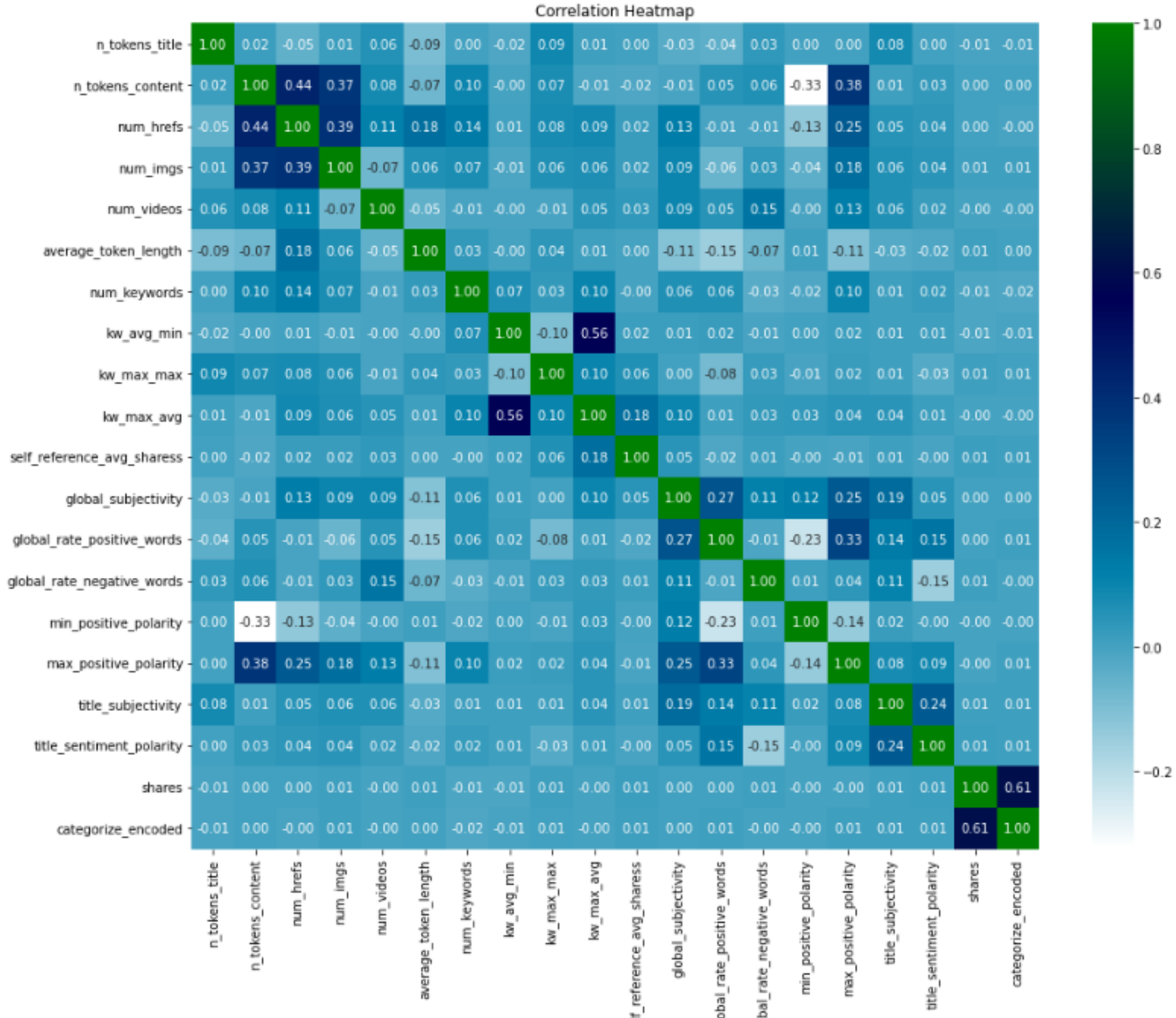
Standardization



- **After generating the correlation matrix we can notice that there are several correlations between certain columns that will allow us to drop them**

- **we can drop 21 columns after studying our matrix**

Standardization



- **Now after the removal of the columns 'useless' we have only 20 columns of which our 2 columns target which will allow us now to start our model**

FIFTH PART

Modeling

Modeling

For this classification problem, we chose to set up many models ML with different approaches:

- **Decision Tree: A very practical simple model for classification problems, easy to visualize and understand practical for feature engineering.**
- **Random Forest: Lets see if we can improve the performance of Decision Tree, and limit over-fitting**
- **XGBoost: To see if boosting gives us better performance than Decision Tree**
- **K-Nearest Neighbor: We wanted to see if a clustering approach would be more Decision Tree**
- **We will also use LinearRegression but we will see after why it will not be interesting to make it**

Modeling

Linear Regression:

- So, First we try the most simple model, the Linear Regression On the target “shares”

Metrics\Modele	Linear Regression	RIDGE Linear regression
MSE	9403313.145463472	9403313.11654438
R^2	0.03225522054648655	0.03225522352270338

MSE is very high and the R^2 score is very poor

Linear Regression:

- Then we tried linear regression but on the “categorize_encoded” columns and here are the results

Metrics\Modele	LASSO Linear Regression
MSE	0.5853919442582628
R^2	0.041474283129691 7
Accuracy	0.3892988929889299

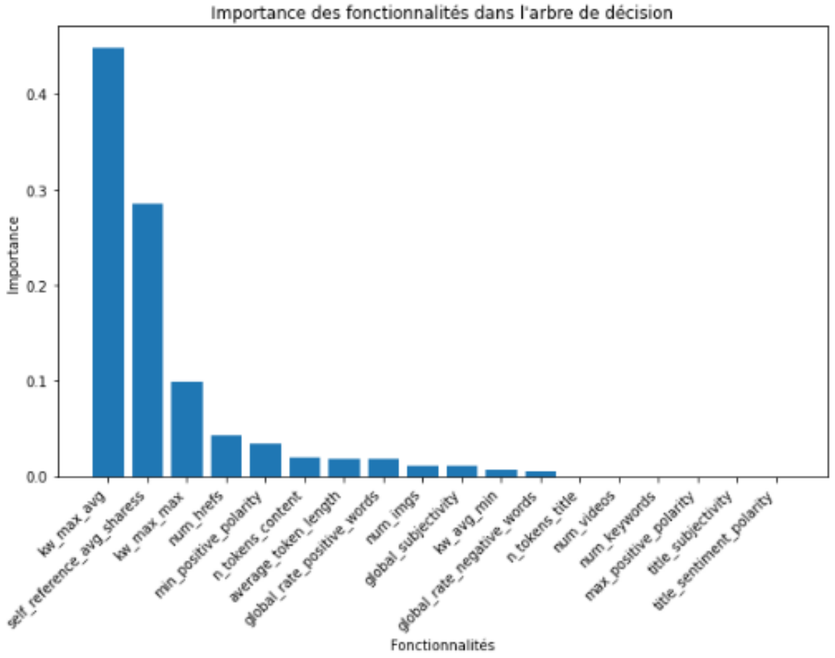
Here we have got a decent accuracy even if the R^2 is still very low

Modeling

Decision Tree Classifier:

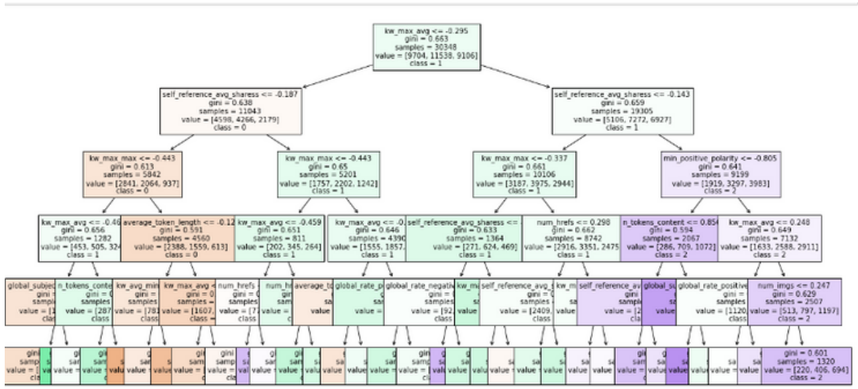
- Now we want to use the Decision Tree Classifier model with all its best parameters so we apply a grid search on the different parameters of the modele on the “categorize_encoded” column

```
Fitting 3 folds for each of 330 candidates, totalling 990 fits
Meilleurs paramètres trouvés :  {'criterion': 'gini', 'max_depth': 5, 'max_features': None, 'min_impurity_decrease': 0.0001}
Accuracy du meilleur modèle : 0.4368740115972588
R² du meilleur modèle : 0.4368740115972588
meamn squared error du meilleur modèle : 0.8473906167633105
```



We have plotted the Importance of each colonne in the modele and all the metrics we wanted:

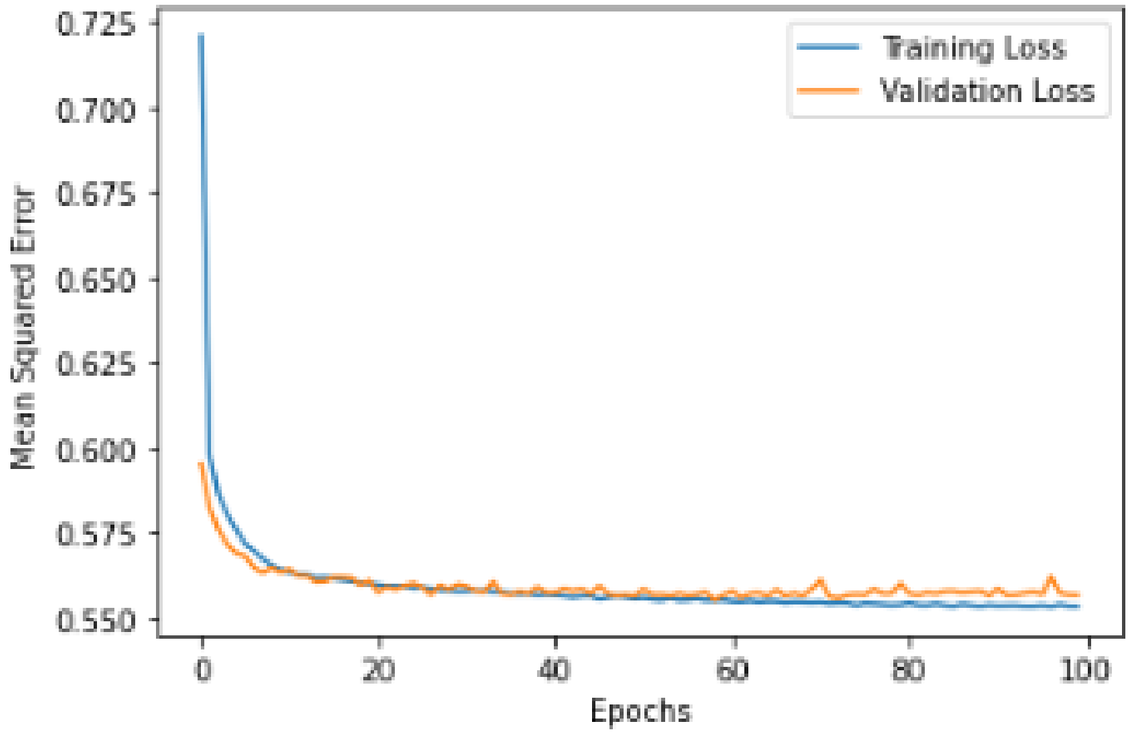
We also plotted a piece of the decision tree to visualise the complexity of the model



Modeling

Sequential Neural Network:

```
Epoch 100 - R²: 0.0888
949/949 [==============================] - 1s 1ms/step - loss: 0.5533 - val_loss: 0.5565
```



We have plotted the Evolution of the training MSE based on epoch compared to the validation MSE

FINAL RESULTS

Choice of the model

Metrics\Modele	Linear Regression	RIDGE Linear regression	GradientBoostingRegressor	RandomForrestRegressor	Sequential (neural network)
MSE	9403313.145463472	9403313.11654438	9065799.520282324	9104061.946441049	9148235.0
R^2	0.03225522054648655	0.03225522352270338	0.06699053603697003	0.06305274702670038	0.0889

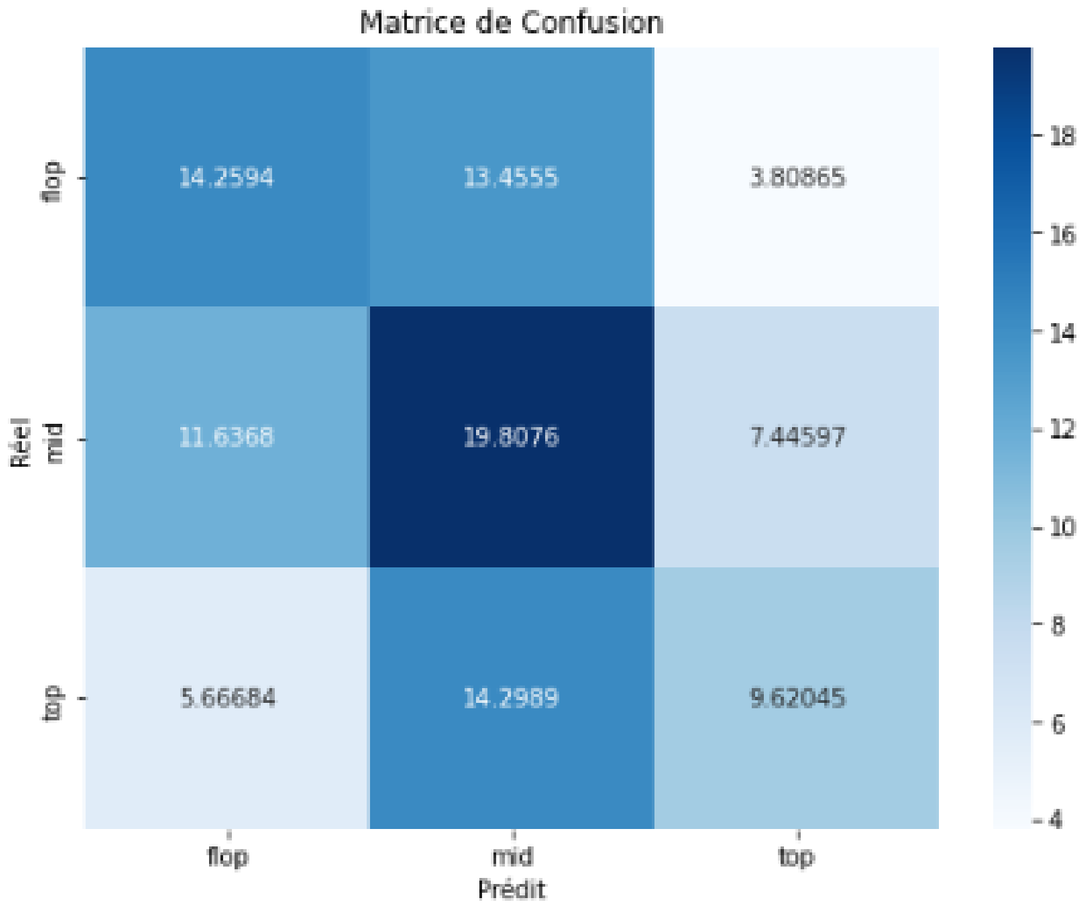
For the target ‘Shares’ we have not efficient results so we will not focus on these modeles but :

Metrics\Modele	LASSO Linear Regression	DecisionTreeClassifier	Sequential (Neural Networ)	Logistic regression
MSE	0.5853919442582628	0.8473906167633105	0.5533	0.7829467580390089
R^2	0.0414742831296917	0.4368740115972588	0.0888	-0.2820036385562745
Accuracy	0.3892988929889299	0.4368740115972588	NaN	0.41552451238798105

For the Categorize column we got DecisionTreeClassifier that seems to be the best model.

FINAL RESULTS

Choice of the model

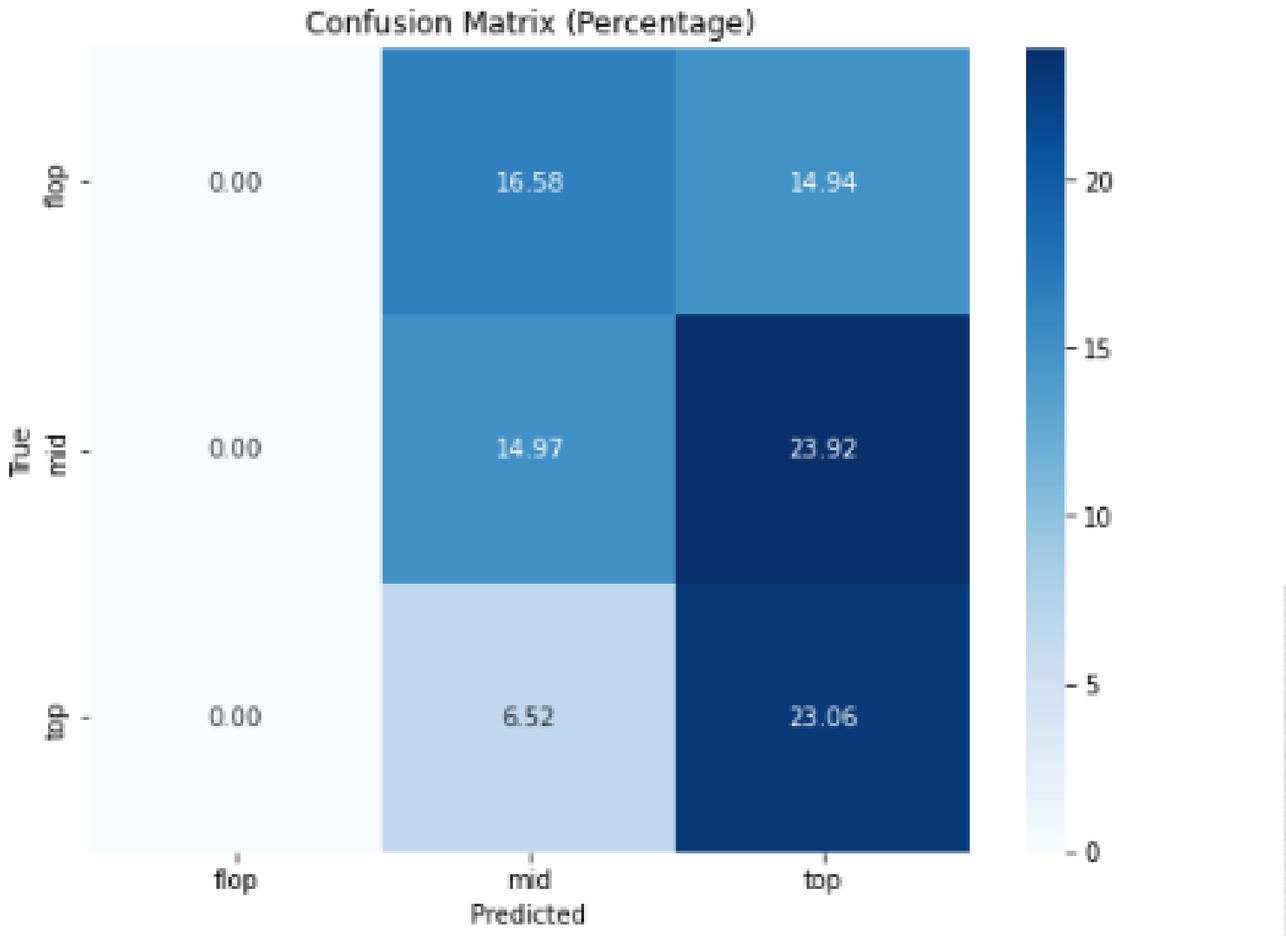


Here is the confusion Matrix for the DecisionTreeClassifier

We can conclude that the modele is the best when he predicts a mid and he is the not performing when he predicts a top or a flop.

FINAL RESULTS

Choice of the model



Here is the confusion Matrix for Random Forrest Regressor

There is a problem because the model never predict a flop., but is a acceptable rate for the top prediction.

Conclusion

The goal was to predict the number of shares of an article. To reach our goal we:

- Cleaned the dataset
- Deleted some uninterresting columns
- Used several modeles on differents target

The results we get :

- We first manage to improve our model by re-cleaning and redefining our dataset.
- We have a model that is 40% accurate on its prediction.

What more could be done :

- Maybe we could have contact the dataset creator to understand a little bit more the signification of all the variable.
- If we had enough time we could have tested other ML Models or other parameters.

THANKS FOR YOUR ATTENTION

BARBIER Lucas

AMENZOU Brahim