ESILV

Online News Popularity Dataset Analysis & Prediction Models Building

PYTHON FOR DATA ANALYSIS 2020-2021 | ESILV

Student : TALEB Sammy

Teacher : Mr BERTIN Luc

Table Of Content

- Presentation of the dataset
- Main Objectives of this project
- Data Analysis
- Features Selection
- Regression Models
- Classification Models
- Conclusion

Presentation of the Dataset

- The Online News Popularity Dataset contains 39797 observations of 61 features.
- These features have been extracted by analyzing web articles
- Among the features we have 58 predictive features, 2 non predictive and the continuous target feature “shares”
- Some features concerns the article itself (number of medias in it, number of urls, release day of the article, …)
- Some other features concerns the document itself (NLP features : number of words, keywords, sentiment analysis, positive/negative words, …)
- The target variable in the number of shares of the article : the prediction task here is to predict this target variable using the predictive features.

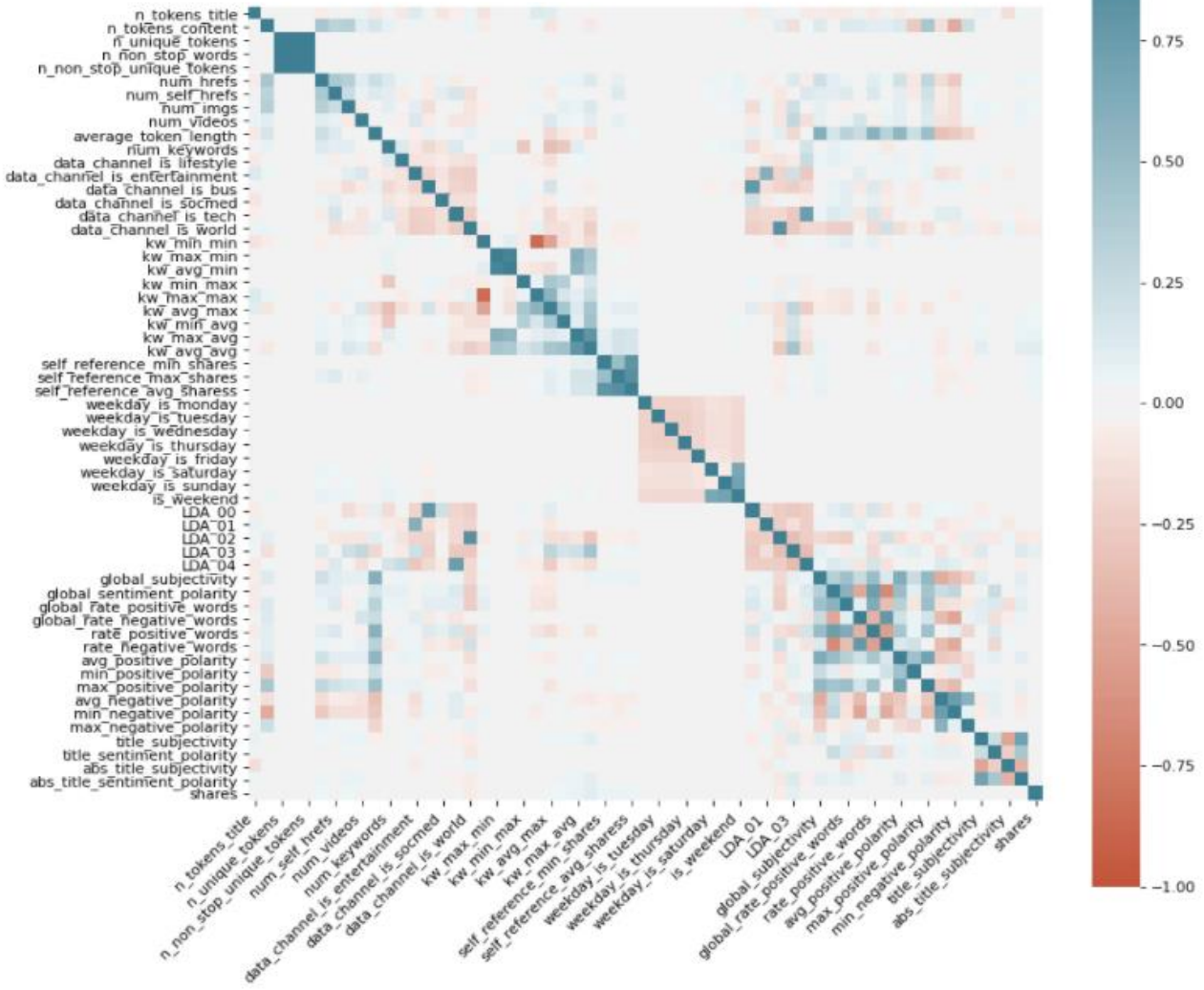
Main Objectives of this project

- As I have said before, the prediction task consist on using all or some of the predictive features to predict the number of shares of each article.
- I will begin by a dataset exploration and analysis , where I will try to identify some relations between some variables and the target one. Some data vizualisation will help to see it clearly and to gather important information.
- After that, I will use all the gathered information to select which features would be relevant. Some Data processing/Feature Selection will be done.
- I will then build some Machine Learning regression models in order to see which performance can we achieve.
- Finally, I will transform the problem into a classification problem (by categorizing the target variable) and build some Machine Learning classifiers. This way, we will be able to identify which task (regression or classification) suits the most to the dataset and leads to the best performance.

Data Analysis & Vizualisation

Correlation Matrix

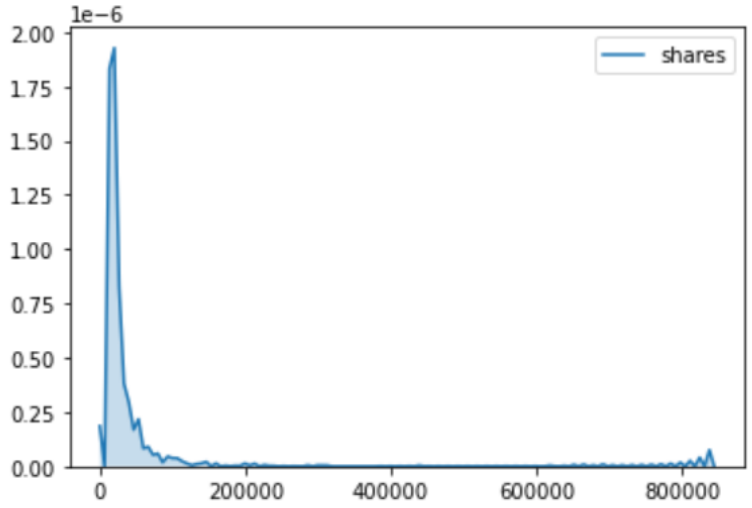
- The main objective of this section is to understand each variable, its impact on the target variable and any relation between them.
- The first plot I have made is a correlation matrix.
- The correlation matrix is very important : it gives interesting information on correlated variables.
- If the correlation coefficient between two variables is higher than 0.7 or lower than -0.7 we can assume that there is a strong relationship exists between them.
- Here, I have identified very strong relationships between the number of unique words, the number of non-stop words, the number of non-stop unique words and between the worst keywords (max min and avg min).
  - In fact their correlation coefficient is very high : very close to 1.
  - Plotting these variables together leaded to an important assumption : they are linearly dependent.



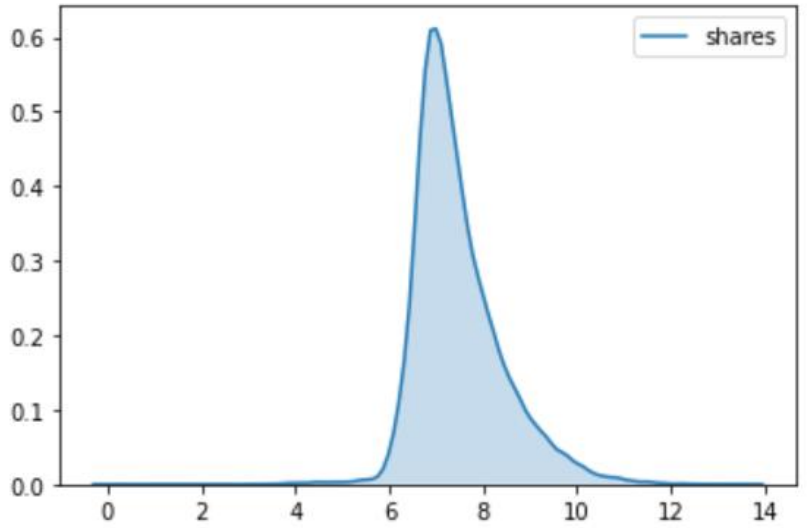
Pearson's correlation matrix of the features of the dataset

Target Feature Distribution

- The second important plot is the distribution of the target variable "shares".
- This plot is mandatory as it gives an idea on how the target variable is distributed. This helps to answer to the questions :
  - *How many articles have a large number of shares ?*
  - *According to this dataset, from which number of shares can we assume that an article is popular ?*
- We can see that the variable is very right skewed : this can lead to lower performance on the models building section. We can apply a processing on the variable to reduce it.
- Applying log on the variable helped to reduce its skew. We will see later if it helps to improve models' performance.



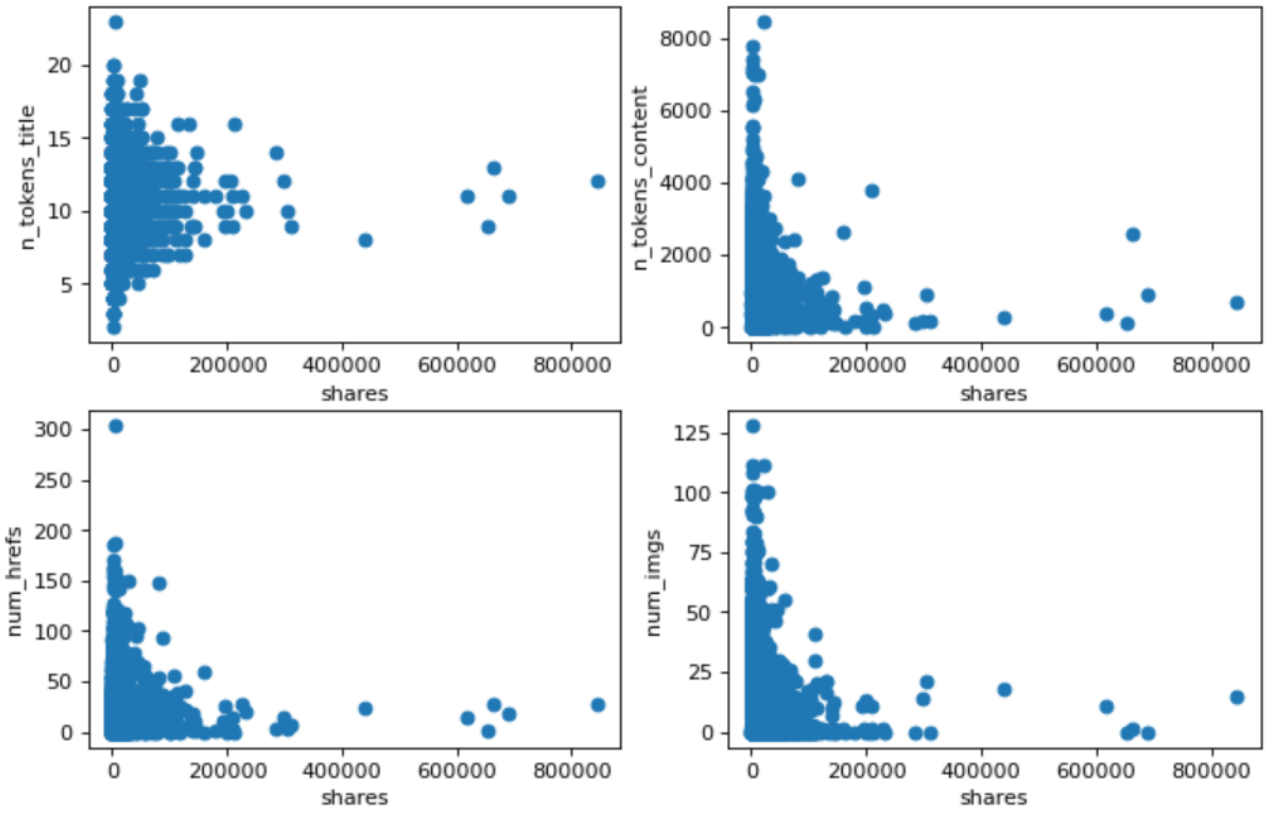
Distribution of the « shares » target variable



Distribution of the log « shares » target variable

Relation between features and target variable

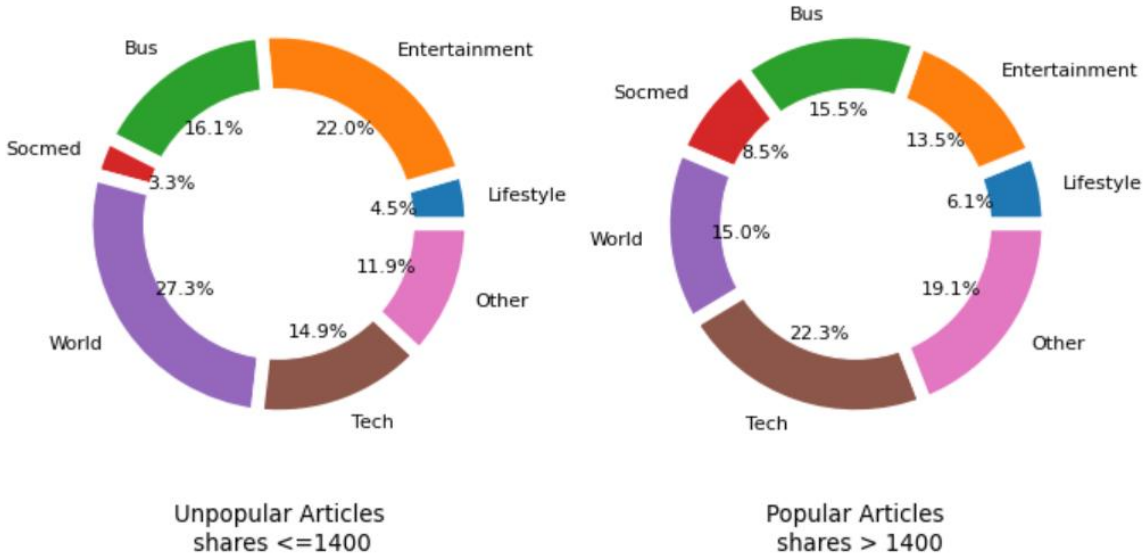
- In this part, the objective was to find any dependency between some predictive features and the target one.
- Here is an example of the most interesting features. I have plotted each feature with the number of shares.
- We can see from these plots that people generally prefer shorts articles as the number of shares increases as the number of words in the content is decreasing.
- The length of the title should not be too long or too short : 8 to 12 words seems to be a good tradeoff.
- Finally, the number of hyperlinks and the number of images should not be too high. The most shared articles have generally low number of links and images.
- These plots are very interesting as the give information on how these variables affects the target variable.



Relationship between features and target variable « shares »

Topic of the articles regarding to their popularity

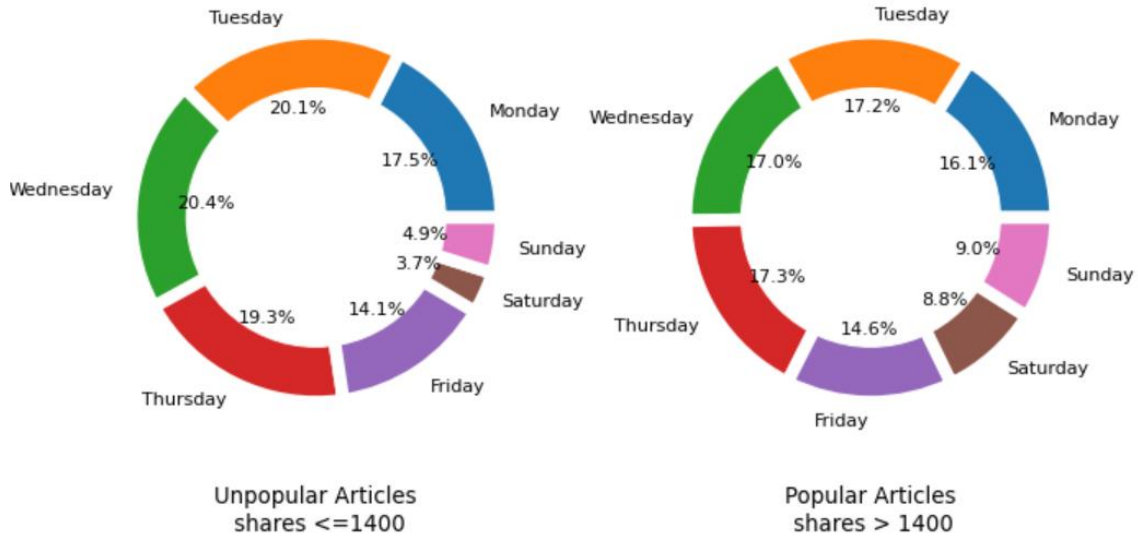
- We have now divided the articles into two groups with almost the same number of observation : The first group has a number of shares larger than 1400 : we call them Popular Articles. The second one has a number of shares less than 1400 : we call them Unpopular Articles.
- The objective here is to identify which topics can lead to a large number of shares.
- The topics are Bus, Socmed, Entertainment, Lifestyle, Technologies, World.
- We can see from these pie charts that Technologies and topics different from those cited above generally lead to a large number of shares.



Topics rate for Popular/Unpopular articles

Release day of the articles regarding to their popularity

- Using the same groups than those created before, the objective is to identify if releasing an article, a specific day of week, can lead to a higher number of shares
- We can see from these pie charts that the proportions are almost the same for both classes.
- A small difference can be highlighted : Weekend rates are higher for popular articles than for unpopular articles.
- The small changes could be a coincidence, but it also could be a relevant information : publishing an article the weekend may lead to a higher popularity



Release day of week for Popular/Unpopular articles

Features Selection

Features Selection

- Now we have explored the dataset, we have to select which features to use to build some machine learning models.
- To do so, we will remove some variables :
  - The two non-predictive variables ‘url’ and ‘timespan’
  - Two of the three highly correlated variables we have seen before : ‘n_non_stop_words’,’ n_unique_tokens’
  - One of the two highly correlated variables : ‘kw_avg_min’
- The highly correlated variables are removed to avoid overfitting on some machine learning models.
- We will also use the processing we made on the ‘shares’ target variable and compare models with both shares and log of shares to identify the differences
- Finally, as we are going to compute classifiers, we will use the two groups before to predict whether an article is popular or not :
  - We create a Boolean target feature ‘Y_bool’
- To build our models, we separate the dataset into a training_set and a test_set with a 0.8/0.2 ratio.

Regression Models

Regression Models

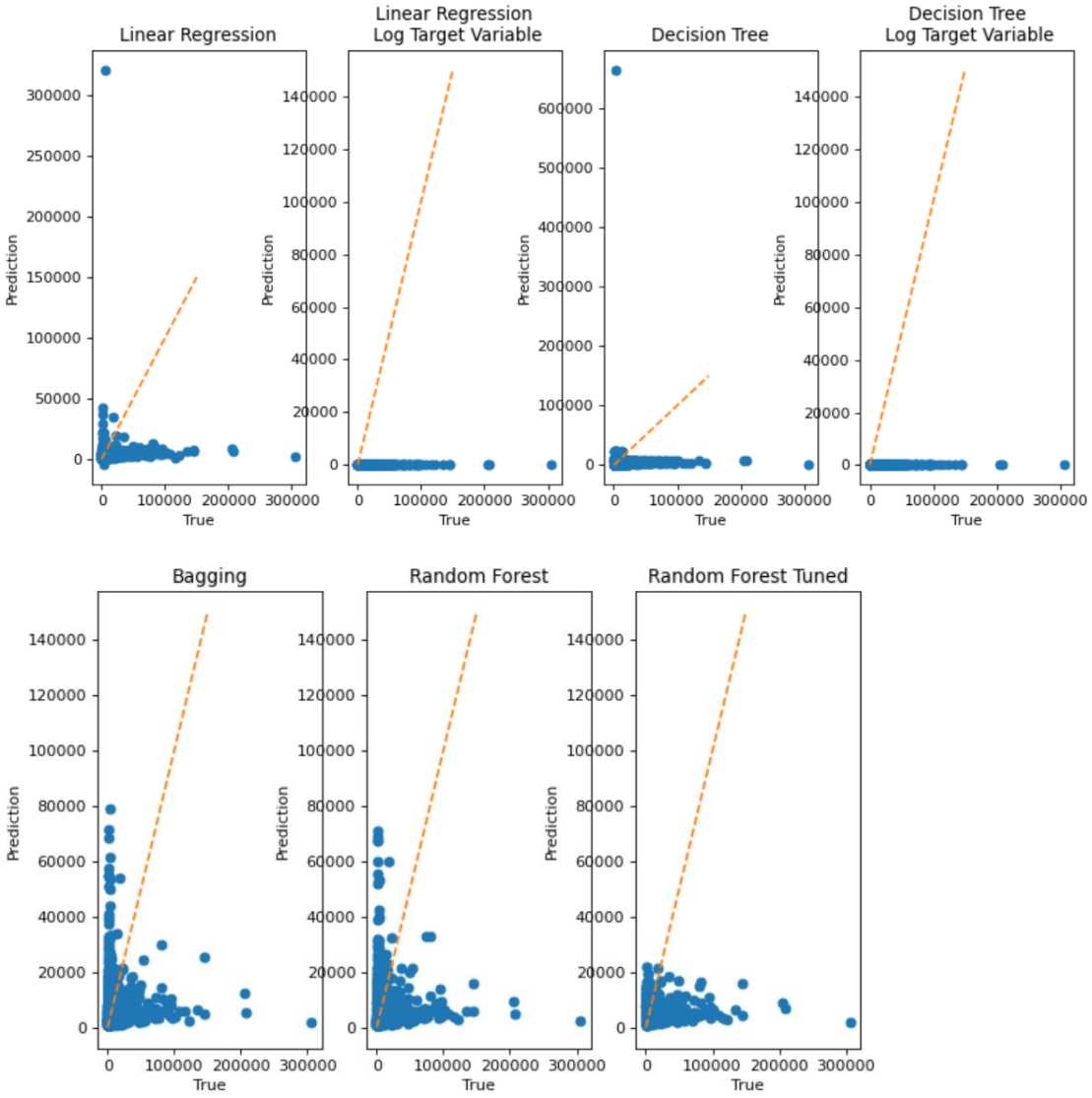
- In this part, the task was a regression one. To do so, I have built different regressor, starting from the simplest to the one which leads to the highest performance.
- The Metric that I have used to compare regression models' performance is the RMSE.
- Here are the first regressors I have built:
  - Linear Regression : RMSE of 9305 on the test set
  - Linear Regression Using log of shares : RMSE of 10205 on the test set
  - Decision Tree With a max_depth of 4 : RMSE of 11407 on the test set
  - Decision Tree With a max_depth of 4 Using log of shares : RMSE of 8782 on the test set
- The results are not very good, knowing that the target values range is from 1 to 800k for the target variable and from 1 to 14 for the log of the target variable. We can notice that using the log of shares helped the decision tree to improve its performance. We could imagine that tree based techniques may perform better using this processing. Nevertheless, the RMSE was always higher using this processing on other tree based techniques. I decided then to stop using this processing

Regression Models

- As the RMSE were not good at all, I thought that scaling would help to improve the performance. Nevertheless, knowing that tree based techniques uses relative distances to fit to the data, scaling would had not helped to improve the RMSE.
- I have then build some stronger regressors, which are :
  - Bagging Regressor : RMSE of 9145 on the test set
  - Random Forest Regressor : RMSE of 9061 on the test set
- As the random forest had performed the best RMSE among these two models, I'd decided to tune its parameters to see if it can achieve a better RMSE.
  - Random Forest Regressor (Tuned parameters) : RMSE of 8553 on the test set
- Even if the improvement was notable, the RMSE is still very high. From now, I've started to ask myself if regression was a good task to perform on this dataset

Comparing Regression Models

- To understand how bad this RMSE is, I tried to compute the R^2 of this last model (Random Forest Regressor). The R^2 was terribly low, 0.02 which shows that 2% of the model explains the variability of the data
- A plot would help to understand it. These plots represent the true values regarding to the predicted values using each model.
- We can see that the predictions are not good at all.
- We would had tried some other regressors such as Gradient Boosting or Adaptative Boosting, but I was not sure we would improve this score a lot. We can conclude than regression tasks are not efficient for this dataset.



Classification Models

Classification Models

- In this part, the task was a classification one. To do so, I have built different binary classifiers, starting from the simplest to the one which leads to the highest performance.
- The Metric that I have used to compare classifiers' performance is the accuracy.
- Here are the first classifiers I have built:
  - Logistic Regression : Accuracy of 60.9% on the test set
  - Logistic Regression With Scaled Data : Accuracy of 65.4% on the test set
  - LDA : Accuracy of 64.9% on the test set
  - QDA : Accuracy of 59.5% on the test set
- We can see that the results are better than for the regression task. We can note that scaling the data before performing the logistic regression helped to improve its accuracy by a 5% rate. Moreover, the low accuracy of the QDA can be explained by the fact that there were multicollinearity among the features. Removing this multicollinearity would help to improve QDA classifier's accuracy but is not mandatory as I had built tree-based models which performed better.

Classification Models

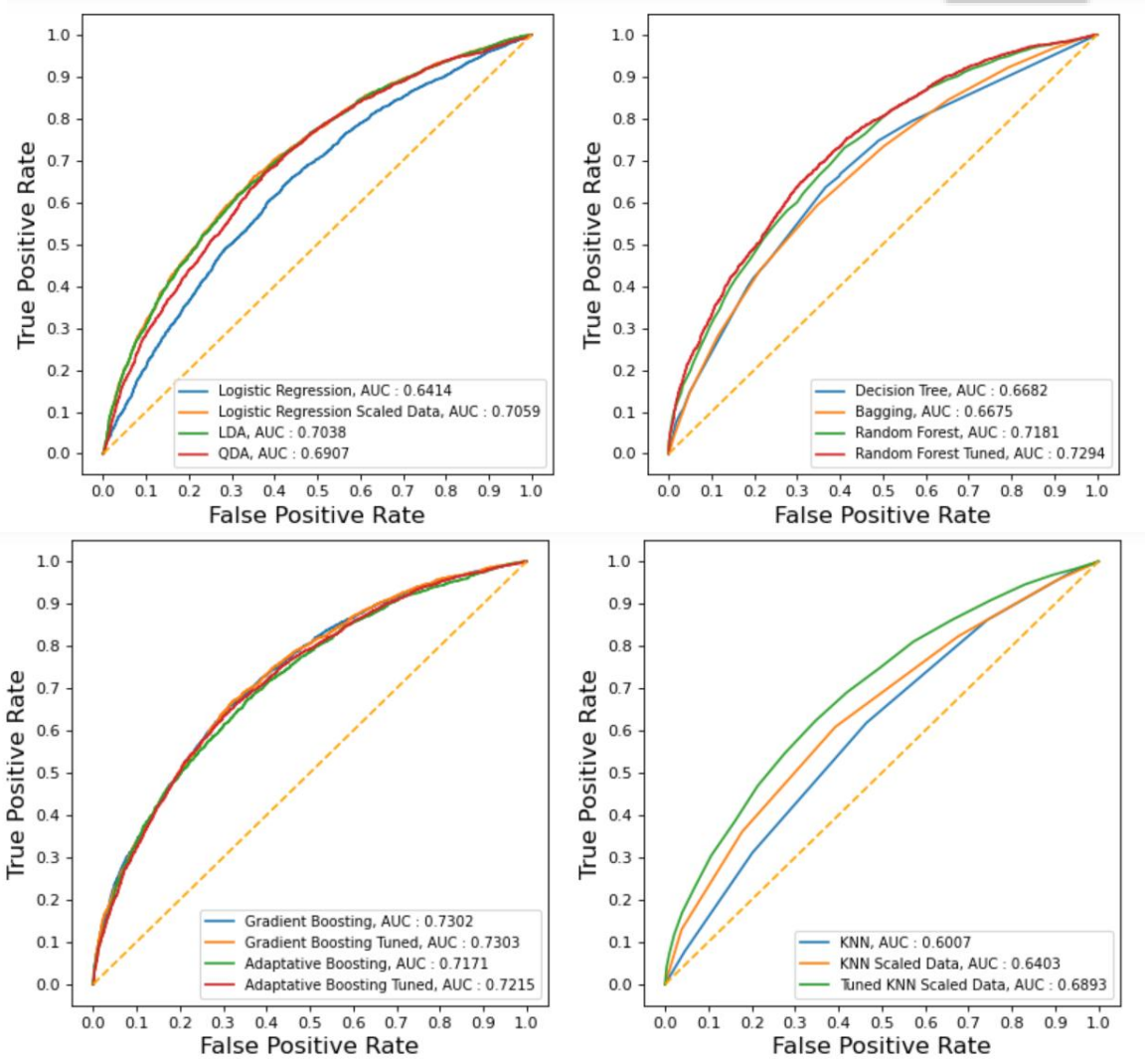
- Here are the tree-based models I have built :
  - Decision Tree With a max depth of 4 : Accuracy of 63.5% on the test set
  - Bagging : Accuracy of 62.1% on the test set
  - Random Forest : Accuracy of 66.1% on the test set
- We can see that the results are better than for the previous classifiers. We achieved a 66.1% accuracy for the random forest classifier. As the author of the dataset performed a 67% accuracy, it was time to tune parameters for each model from now.
- Other Models I have built :
  - Tuned Random Forest : Accuracy of 67.1% on the test set
    - Tuning the random forest helped to increase its accuracy by a 1% rate which is a good improvement as we are reaching the target accuracy of 67%
  - Gradient Boosting : Accuracy of 67.03% on the test set
  - Tuned Gradient Boosting : Accuracy of 66.9% on the test set
- We can see that tuning the gradient boosting did not helped to increase the accuracy. Nevertheless, as there is a strong random factor for these tree-based classifiers, we cannot say that this last model has a lower performance than the gradient boosting

Classification Models

- Final Models :
  - AdaBoost : Accuracy of 65.7% on the test set
  - Tuned AdaBoost : Accuracy of 66.2% on the test set
  - KNN Classifier : Accuracy of 57.9% on the test set
  - KNN Classifier, scaled data : Accuracy of 60.8% on the test set
  - Tuned KNN Classifier, scaled data : Accuracy of 63.7% on the test set
- These final models achieved a good accuracy on the test set. It is remarkable that scaling the data, then tuning the model helped the KNN classifier to improve its accuracy by a 6% rate which is a good improvement. We can assume that the KNN classifier's performance is impacted by scaling. Nevertheless, none of these models had beaten the Random Forest and the Gradient Boosting which have the two highest accuracies.

Comparing Classification Models

- Now we have built all models, we can now compare them. The first plot we can make to compare them, is to plot their ROC curves. Recall that the highest the AUC, the better the model.
- We can see from these plots that the best model is the Gradient Boosting Classifier with tuned parameters, which achieved an AUC of 0.73.



Comparing Classification Models

- In this report, we used the accuracy as the metric to compare the classifiers, but there are some other metrics that can be very interesting depending on the situation
- The highest precision has been achieved by the QDA : 72.6% which means that if an article is predicted as a popular one, it will be true in 72% of the cases.
- The highest recall has been performed by the Decision Tree : 74.3% which means that the model finds 74% of the popular articles.
- The highest Accuracy has been performed by the Tuned Random Forest : 67.1%
- The highest F1 score has been performed by the Tuned Random Forest : 70.3% The F1 score is a better metric than the accuracy if the classes are not equally distributed and if the cost of FP and FN is not the same. As this is not our case, we can compare our models using the accuracy

Conclusion

- Working on this dataset, we have seen that classification tasks performed better than regression tasks. The best classifiers (Random Forest and Gradient Boosting, both with tuned parameters) performed enough to use them for prediction tasks on articles. Nevertheless, it could be interesting to try more powerful predictive methods such as fitting a Deep Learning network to this dataset.
- Moreover, we have seen that some features came from NLP analysis of these articles. Using other NLP indicators may help to improve models' efficiency.
- Note finally that as they were the best models for prediction, both gradient boosting and random forest binary classifiers will be deployed in the API.