Python for Data Analysis

Mashable Online News Popularity

Hugo DEBES & Ugo DEMY – DIA4

Is an online article likely to be popular ?

A **previous research paper** was published explaining how the dataset was built and the work that has already been done on it. Feel free to check !

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.

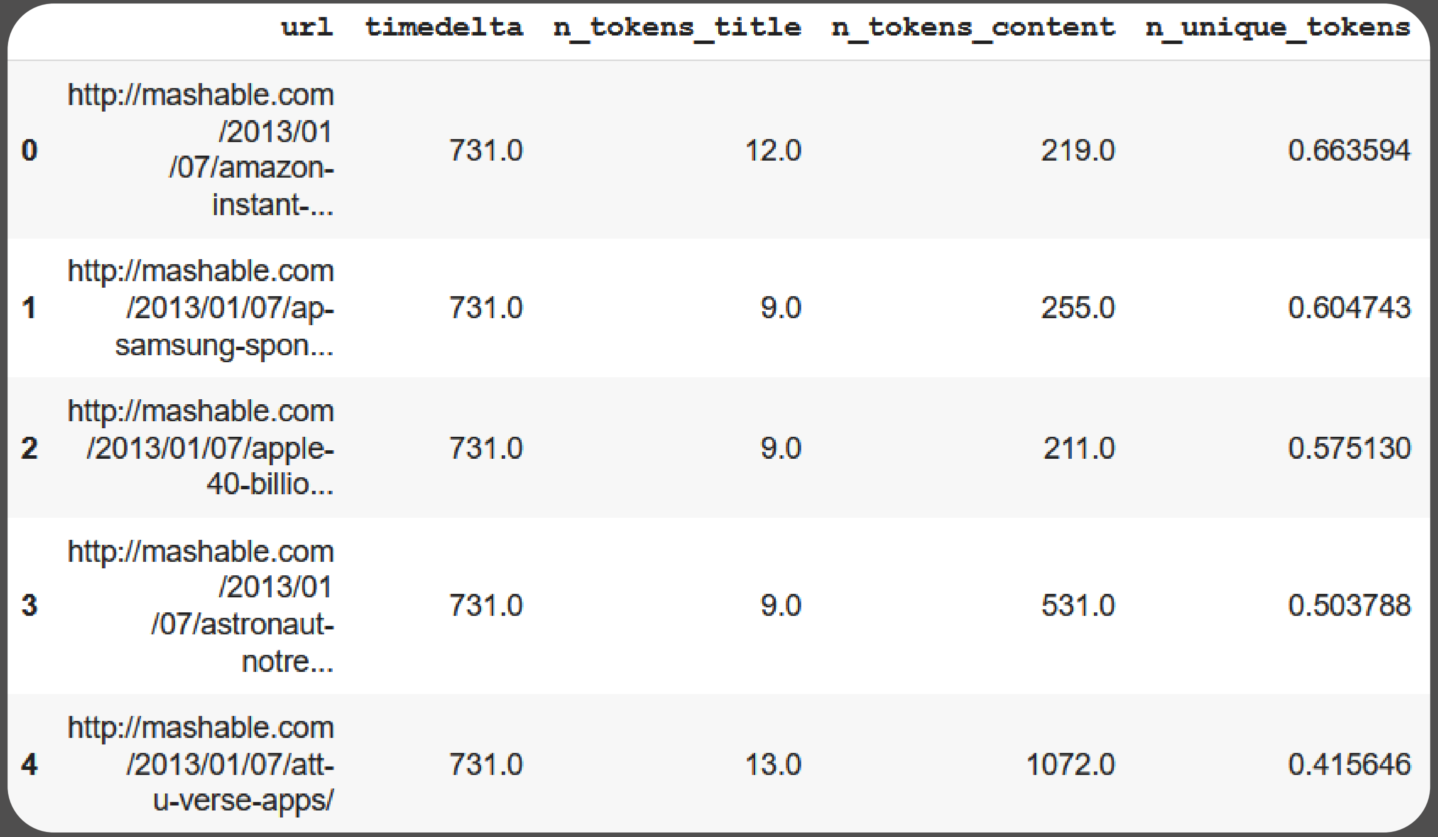
The *OnlineNews.csv* dataset

Quick Overview

We used the ***Online News Popularity*** dataset provided by the UCI Machine Learning Repository. This dataset is part of a research project aiming at **predicting the popularity of an article** on one hand and **optimizing the features** to improve it on the other hand. The interesting part is that we analyze article **prior** to their publication to help writers have a better understanding of their work before putting it out to the world. The articles were published on *Mashable* (*www.mashable.com*).

If you want to download it, follow this link : *https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity*

Our dataset here is pretty sizable. Indeed, with a number of **39797** instances and **61** attributes (58 predictive attributes, 2 non-predictive, and 1 goal field), we have data to feed our models. There aren't **any missing values**. Thus, our problems are very likely to be the treatment of outliers, disrupted statistics, and samples choice.



	url	timedelta	n_tokens_title	n_tokens_content	n_unique_tokens
0	http://mashable.com/2013/01/07/amazon-instant-...	731.0	12.0	219.0	0.663594
1	http://mashable.com/2013/01/07/ap-samsung-spon...	731.0	9.0	255.0	0.604743
2	http://mashable.com/2013/01/07/apple-40-billio...	731.0	9.0	211.0	0.575130
3	http://mashable.com/2013/01/07/astronaut-notre...	731.0	9.0	531.0	0.503788
4	http://mashable.com/2013/01/07/att-u-verse-apps/	731.0	13.0	1072.0	0.415646

The *OnlineNews.csv* dataset

*** Input type not mentioned ➔ float64**

- *url* : URL of the article - object
- *timedelta* : Days between the article publication and the dataset acquisition
- *n_tokens_title* : Number of words in the title
- *n_tokens_content* : Number of words in the content
- *n_unique_tokens* : Rate of unique words in the content
- *n_non_stop_words* : Rate of non-stop words in the content
- *n_non_stop_unique_tokens* : Rate of unique non-stop words in the content
- *num_hrefs* : Number of links
- *num_self_hrefs* : Number of links to other articles published by Mashable
- *num_imgs* : Number of images
- *num_videos* : Number of videos
- *average_token_length* : Average length of the words in the content
- *num_keywords* : Number of keywords in the metadata
- *data_channel_is_lifestyle* : Is data channel 'Lifestyle'?
- *data_channel_is_entertainment* : Is data channel 'Entertainment'?
- *data_channel_is_bus* : Is data channel 'Business'?
- *data_channel_is_socmed* : Is data channel 'Social Media'?
- *data_channel_is_tech* : Is data channel 'Tech'?
- *data_channel_is_world* : Is data channel 'World'?
- *kw_min_min* : Worst keyword (min. shares)
- *kw_max_min* : Worst keyword (max. shares)
- *kw_avg_min* : Worst keyword (avg. shares)
- *kw_min_max* : Best keyword (min. shares)
- *kw_max_max* : Best keyword (max. shares)
- *kw_avg_max* : Best keyword (avg. shares)
- *kw_min_avg* : Avg. keyword (min. shares)
- *kw_max_avg :* Avg. keyword (max. shares)
- *kw_avg_avg* : Avg. keyword (avg. shares)
- *self_reference_min_shares* : Min. shares of referenced articles in Mashable
- *self_reference_max_shares* : Max. shares of referenced articles in Mashable
- *self_reference_avg_shares* : Avg. shares of referenced articles in Mashable
- *weekday_is_monday* : Was the article published on a Monday?
- *weekday_is_tuesday* : Was the article published on a Tuesday?
- *weekday_is_wednesday* : Was the article published on a Wednesday?
- *weekday_is_thursday* : Was the article published on a Thursday?
- *weekday_is_friday* : Was the article published on a Friday?
- *weekday_is_saturday* : Was the article published on a Saturday?
- *weekday_is_sunday* : Was the article published on a Sunday?
- *is_weekend* : Was the article published on the weekend?
- *LDA_00* : Closeness to LDA topic 0
- *LDA_01* : Closeness to LDA topic 1
- *LDA_02* : Closeness to LDA topic 2
- *LDA_03* : Closeness to LDA topic 3
- *LDA_04* : Closeness to LDA topic 4
- *global_subjectivity* : Text subjectivity
- *global_sentiment_polarity* : Text sentiment polarity
- *global_rate_positive_words* : Rate of positive words in the content
- *global_rate_negative_words* : Rate of negative words in the content
- *rate_positive_words* : Rate of positive words among non-neutral tokens
- *rate_negative_words* : Rate of negative words among non-neutral tokens
- *avg_positive_polarity* : Avg. polarity of positive words
- *min_positive_polarity* : Min. polarity of positive words
- *max_positive_polarity* : Max. polarity of positive words
- *avg_negative_polarity* : Avg. polarity of negative words
- *min_negative_polarity* : Min. polarity of negative words
- *max_negative_polarity* : Max. polarity of negative words
- *title_subjectivity* : Title subjectivity
- *title_sentiment_polarity* : Title polarity
- *abs_title_subjectivity* : Absolute subjectivity level
- *abs_title_sentiment_polarity* : Absolute polarity level
- *shares (target)* : Number of shares – int64

5 Successful Article Ingredients

Thanks to those details, we were able to **emphasize specific features** such as the number of tokens in the content and the title, the number of images and links and, the average token length. Moreover, it gave us the idea to look for the number of articles made by each author.

Be brief

Get to the point quickly. Sweet spot between 1000 to 3000 words. Write irresistible headlines.

Be visual

Key to capture reader interest. At least one image in a post leads to ~twice as many shares (*Buzzsumo*)

Be simple

No big elaborated words. You are sending a message you hope lots of readers understand, so pare back on the jargon.

Play nice with others

Give credits where it’s due by linking to sources. Links send traffic to the source and increase visibility and popularity.

Play the numbers game

The more you post, the more it attracts people (some post 10 to 15 times). Also, when you try to make something for everyone, it usually serves no one

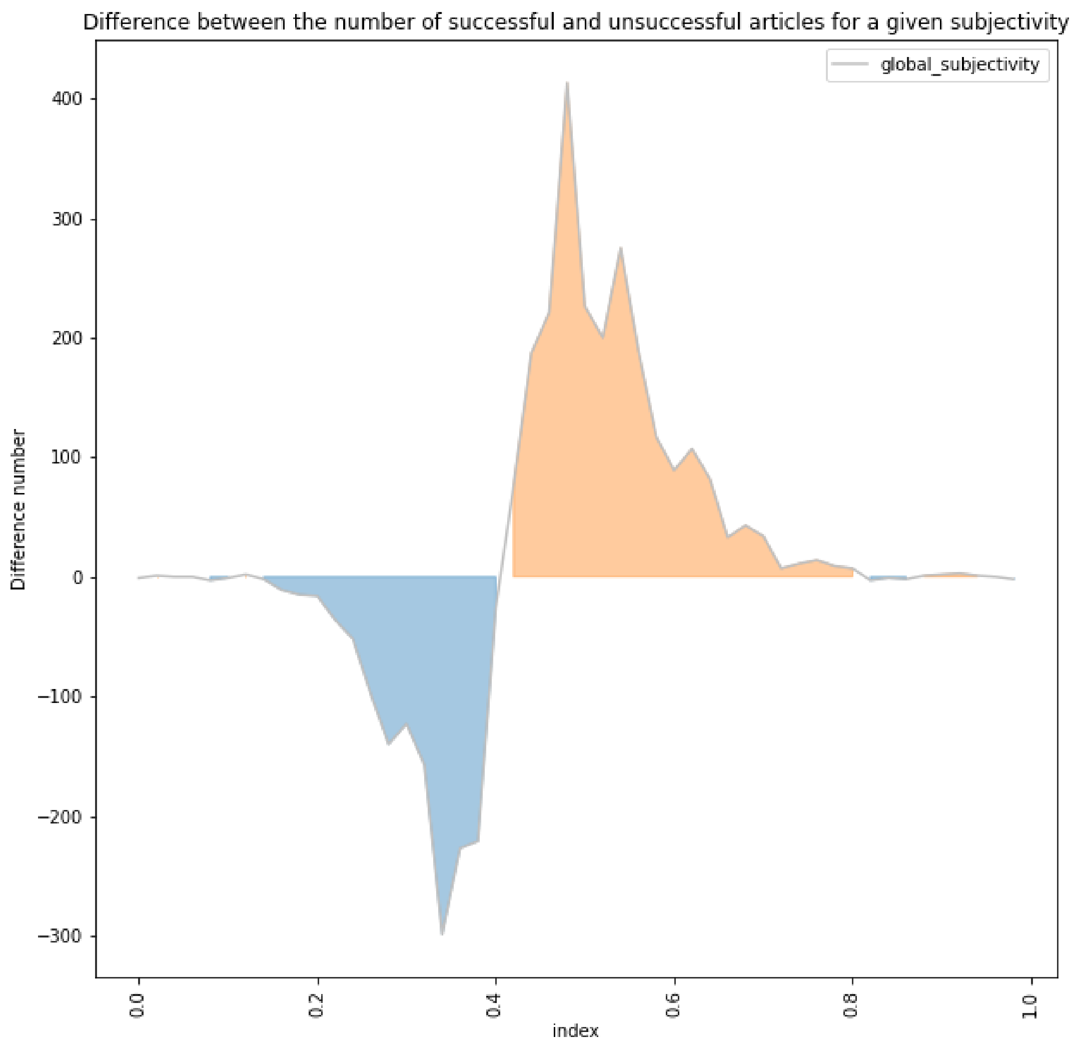
Express your Feelings !

Analysis of the Subjectivity

Subjectivity quantifies the emotional implication of a text. It is a float which lies in the range of [0,1] where 0 means a factual information whereas 1 an opinion.

In the graph, the more you talk about feelings the more you are likely to write a successful article.

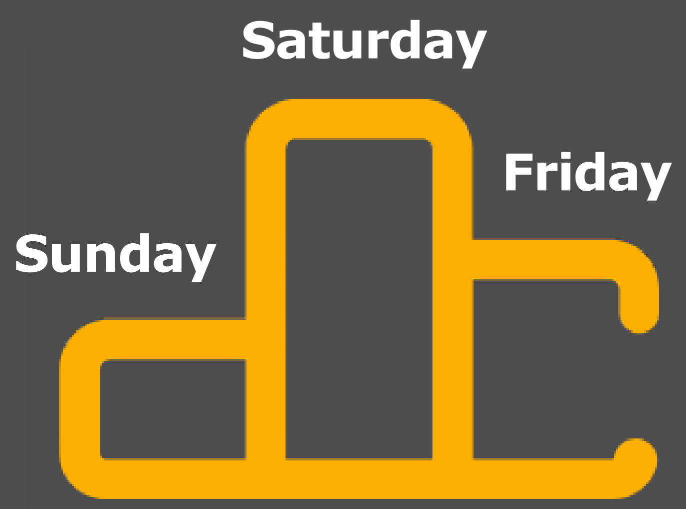
Stories that evoke intense emotions tend to drive popularity. The content that triggers "high-arousal" emotions performs better online whereas content that sparks "low-arousal" emotions is less viral.



Advice

Day of Publication

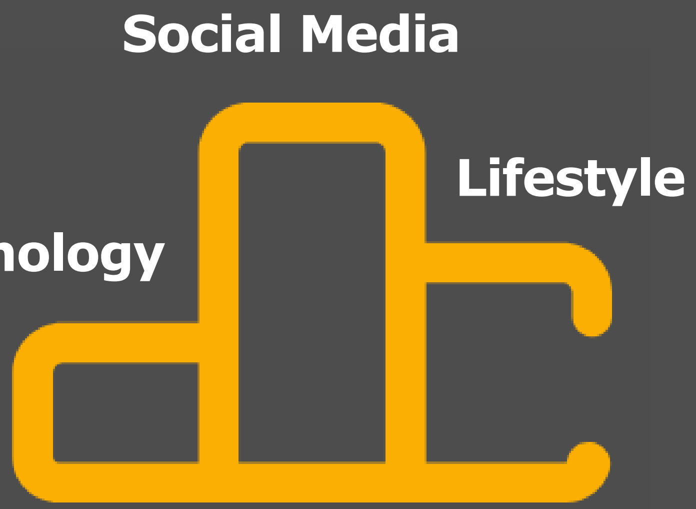
It's an advantage to post during the weekend in Mashable. It is not true for all platforms as we have seen that it is better to post on Tuesday and Thursday on LinkedIn for example.



Advice

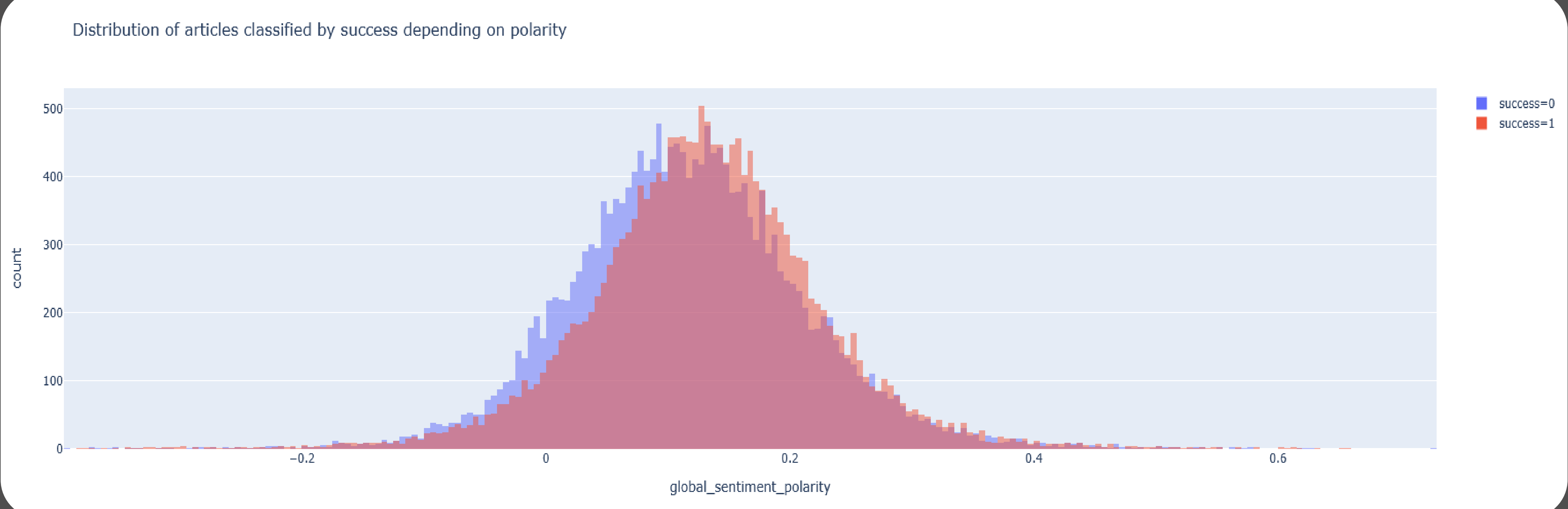
Best Topics

Finally, the **social media** channel is the grand winner in terms of popularity : the first quartile is equal to 1300 which means almost 75% of articles in this channel are successful.



Be Positive !

Analysis of the Polarity



Polarity is float which lies in the range of [-1,1] where 1 means positive statement and -1 means a negative statement.
In the graph, the more you are positive the more you are likely to write a successful article. Around 0.1, we see a switch regarding the highest number between the successful and unsuccessful articles.

Stories that evoke intense emotions tend to drive popularity. The content that triggers “high-arousal” emotions performs better online whereas content that sparks “low-arousal” emotions is less viral.

Machine Learning

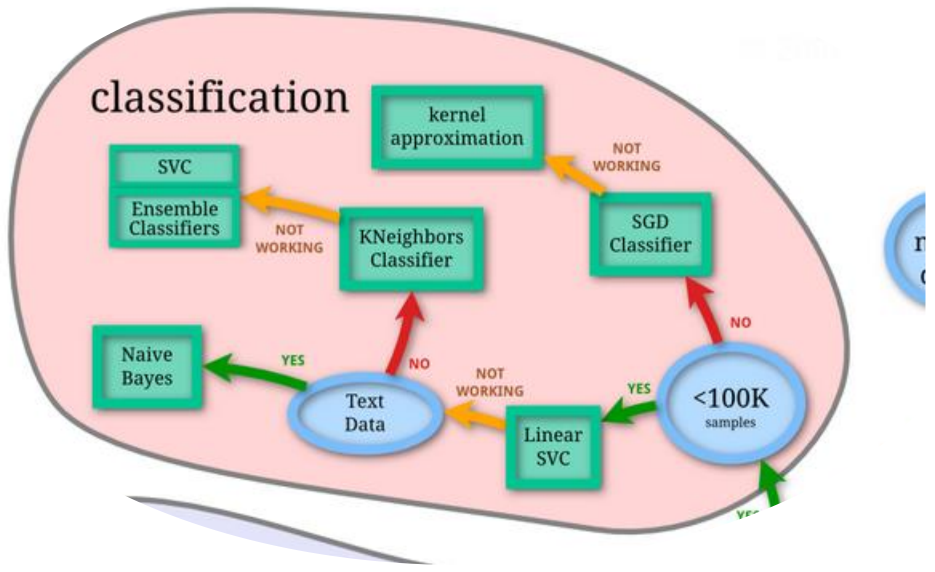
Classification / Regression ?

In the research paper, the authors wanted to predict whether an article was likely to be successful or not. In consequence, they set a **threshold to 1400** over which the article was considered popular. However, we thought that 1400 was quite subjective, even based on the number of Mashable users at that time, and we aimed for a model **as general as possible**.

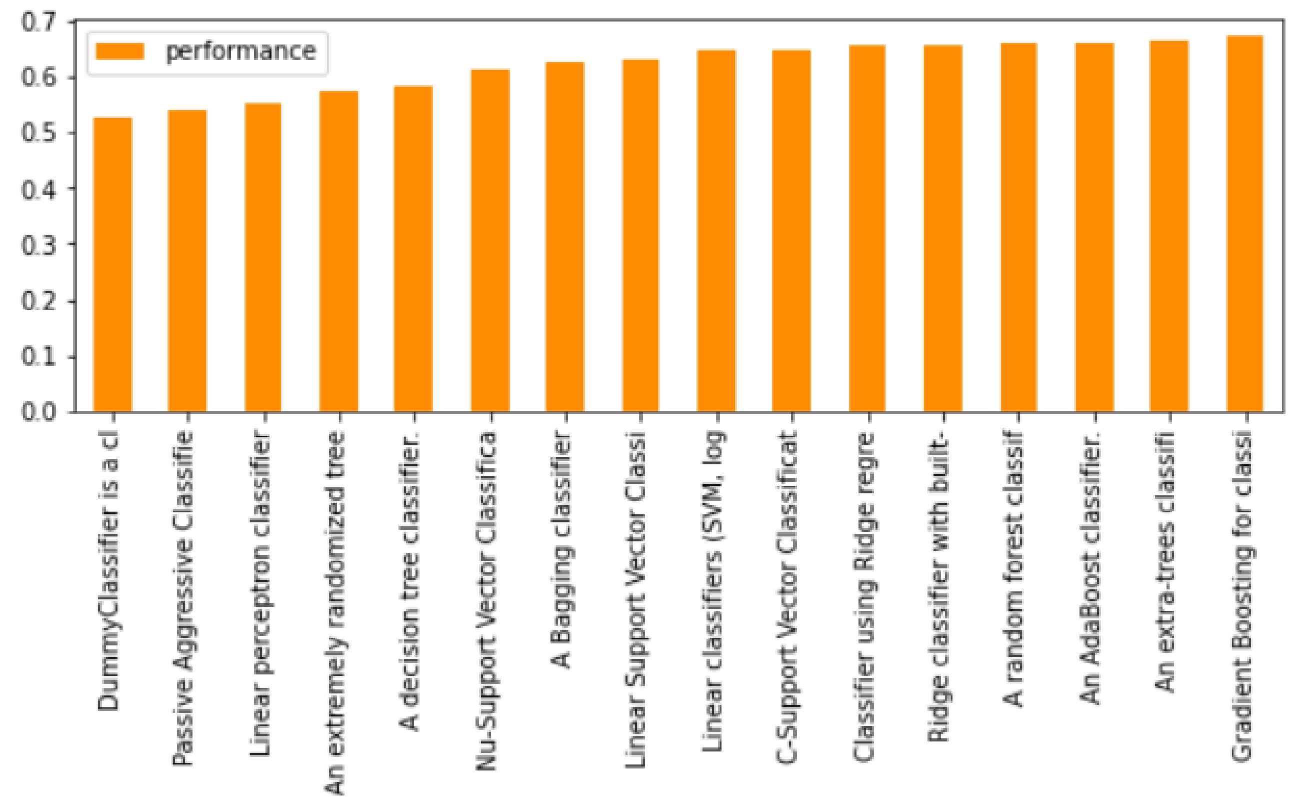
Thus, our first objective was to **predict the number of shares** an article will receive after its publication **nuanced by its *timedelta*** (days passed since the article publication).

But, after implementing some regression models, we faced an **important MSE score and a R² close to 0**. Cleary meaning that our models weren't representing the data. The problem was still there even after **treating the outliers** that we knew some models were sensible to, and **applying a logarithm** to normalize the shares distribution's curve (c.f *Regression Exploration* part).

So we decided to return to the classification problem, with the 1400 threshold, and **improve the previous scores** by implementing recent models and performing GridSearch.



After looking at the **Scikit Learn** library and all the algorithms it offers, we first decided to do a **brute force method** that would test **all the classification models** without doing any GridSearch. This allowed us to see which algorithm would get the **best performance score with the base hyperparameters**. It turned out to be *Gradient Boosting Classification*.



Some Web Scrapping

Origin & Reasons

During our research, we found that **playing the numbers game** was very efficient. This means that **the more articles you post, the more likely you are to have one of them popular**. Neetzan Zimmerman, who the Wall Street Journal called possibly “the most popular blogger working on the Web today”, shared in an interview with HubSpot.comthat, he **posts 10 to 15 times per day**. Not every post went viral, but the larger the volume of stories, the greater the chances of one taking off.

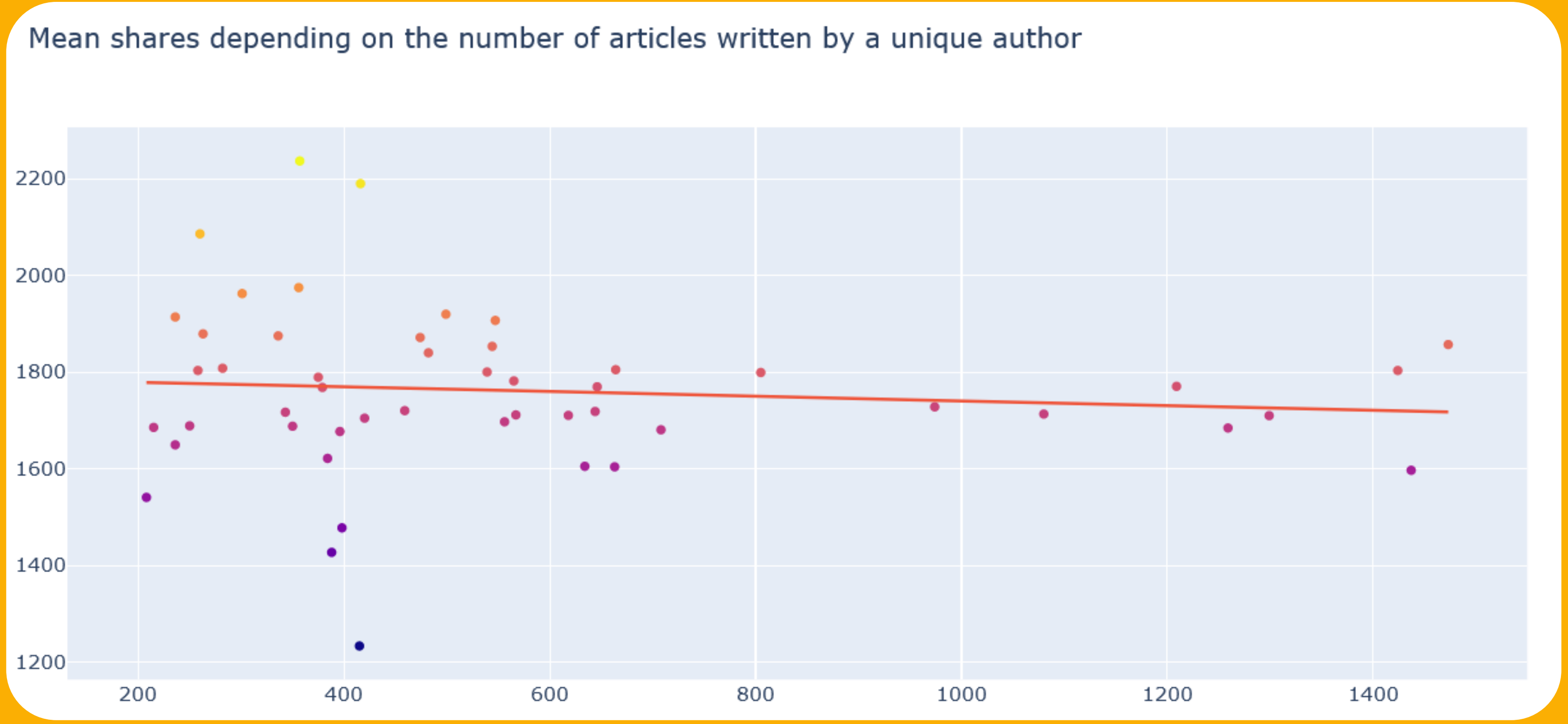
Thus, to study the impact of the number of articles written by an author of the shares, we decided to **scrap the author's names directly from Mashable.com** with *BeautifulSoup*, using *SoupStrainer* and *cchardet* librairy to fasten it. Reaching ~2 requests per second, it took almost 8h to get everything. From here, we created a **new variable** called *same_author* compiling the **number of articles done by the same author**.

Issues encountered

- Almost **2sec per request** at the beginning - Solved with *SoupStrainer* and *cchardet* librairy.
- **Spyder crashing or Mashable.com aborting connections** due to too many requests - Solved by saving the author's list to a **pickle file every 50** new.

```python
def author_date(url):
  page = requests.get(url)
  strainer_author = SoupStrainer('a', attrs={"class": "underline-link"})
  soup = BeautifulSoup(page.content, 'lxml', parse_only=strainer_author)
  try:
    author = soup.get_text(strip=True)
  except:
    author='Unknown'
  return author
```

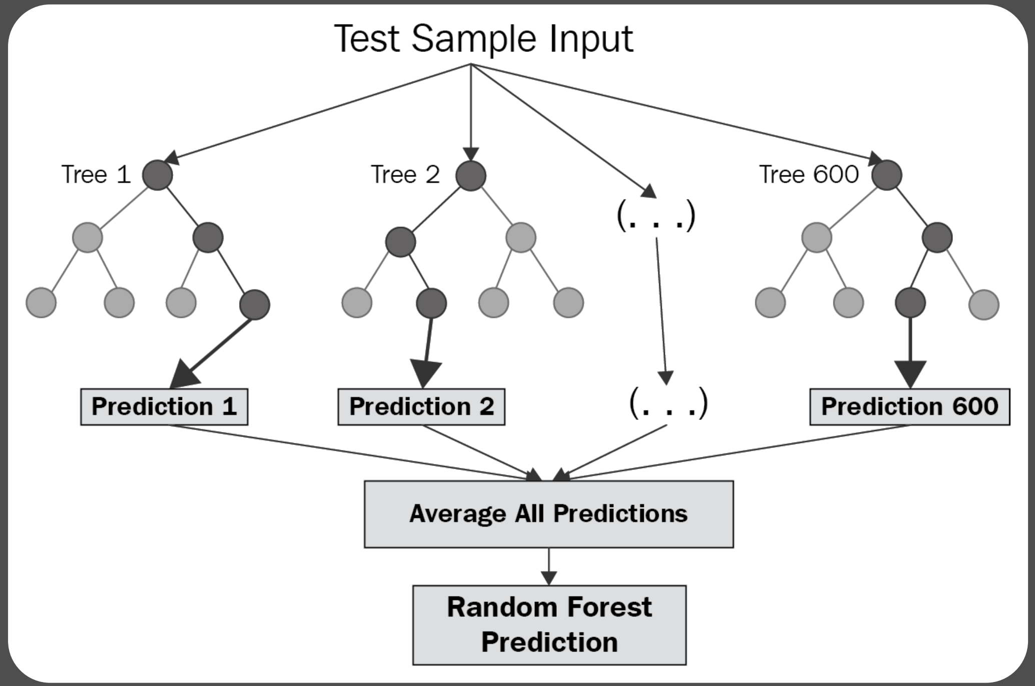
Once done, we used the *map()* function to **assign each article** a new variable indicating the **number of articles written by the same author**. However, the trendline for the author having at least 200 articles **didn't confirm our theory**. This can be partially explained by the fact that the people that post almost 2 articles a day for 2 years such as Sam Laird may **not focus on the quality** of their contents **rather than quantity**. Moreover, if they **don't actively promote** it on the social medias for example, the must not be really famous anyways.

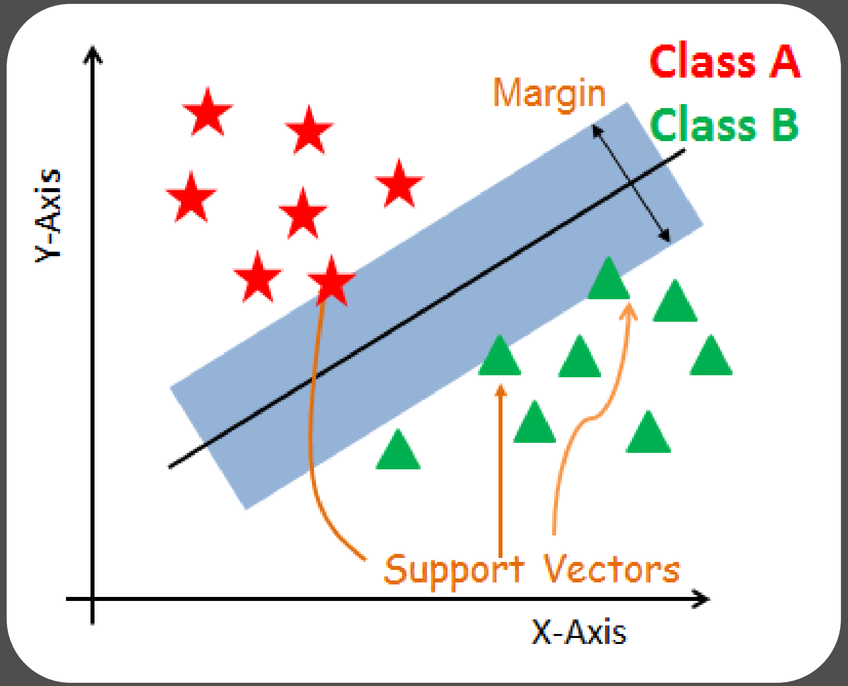


ML & Deep Learning Models

The Random Forest algorithm is an **ensemble method** for classification or regression based on **multiple learning decision trees** trained on **slightly different subsets** of data.

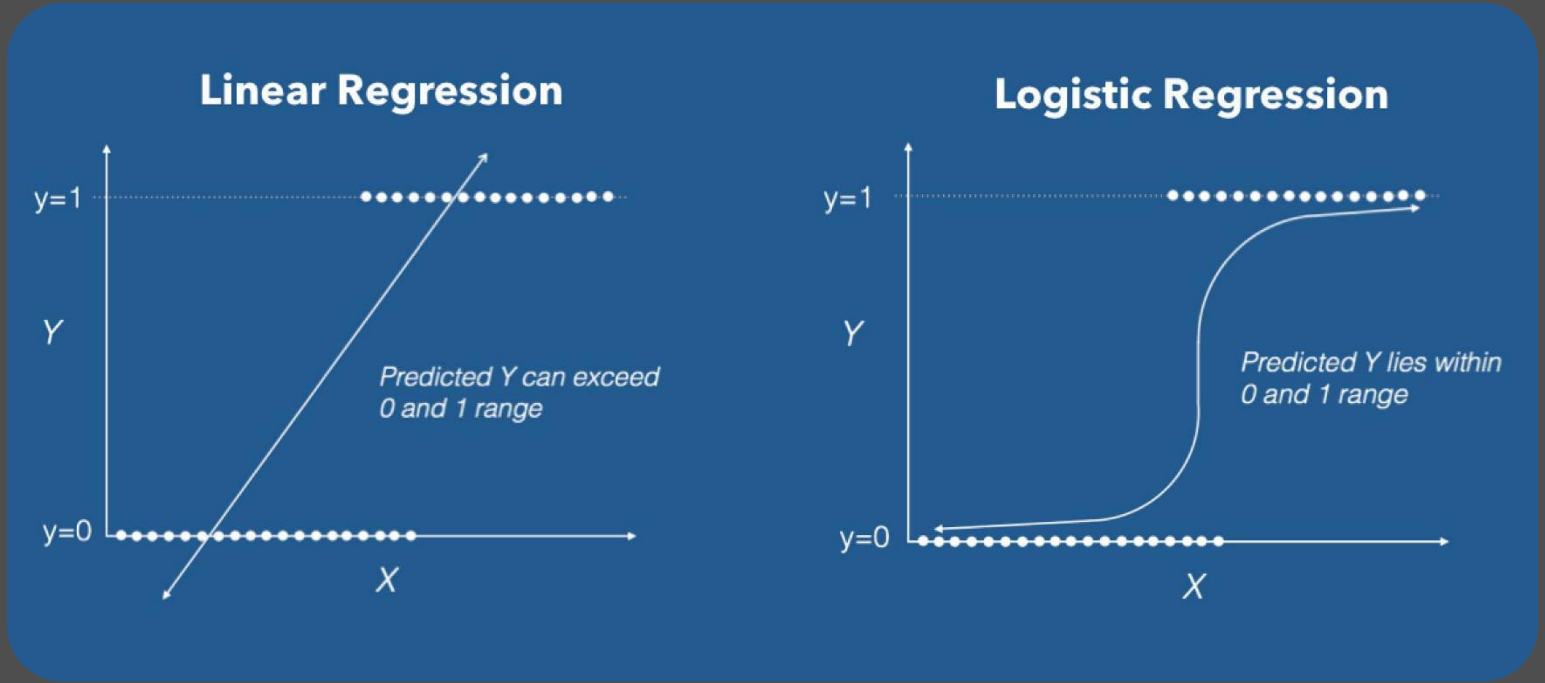
- An excellent **trade-off between interpretability and efficiency**. This allows, for example, to determine **which features were decisive** for obtaining a prediction while remaining accurate.
- Reduce the risk of error and **avoid overfitting** by considering different subsets (*low-correlated trees*).





SVC (or **Support Vector Classifier**) is a machine learning algorithm belonging to the Support Vector Machine (SVM) family. SVM algorithms use the **vector component of the dataset's samples** to determine a preferential orientation.

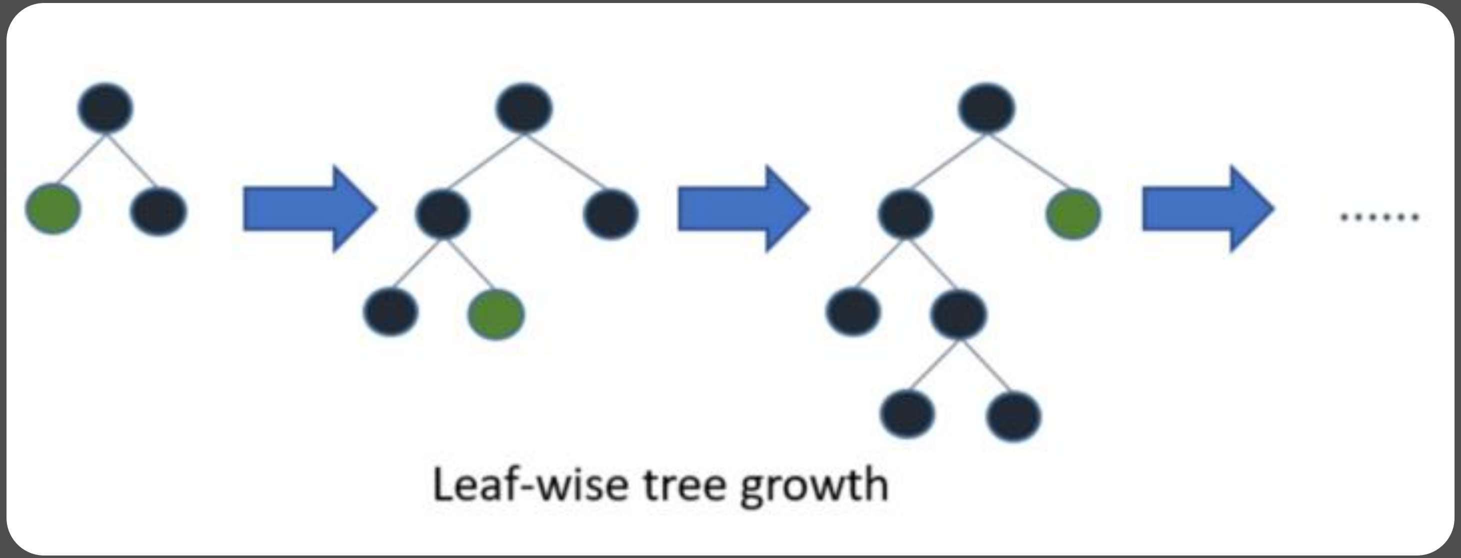
- **Effective in high dimensional spaces** and **very versatile** thanks to the different kernels available.
- **Memory efficient**, by using a **subset** of training points in the decision function (called support vectors).


Linear Regression
y=1
Y
y=0
X
Predicted Y can exceed 0 and 1 range
Logistic Regression
y=1
Y
y=0
X
Predicted Y lies within 0 and 1 range

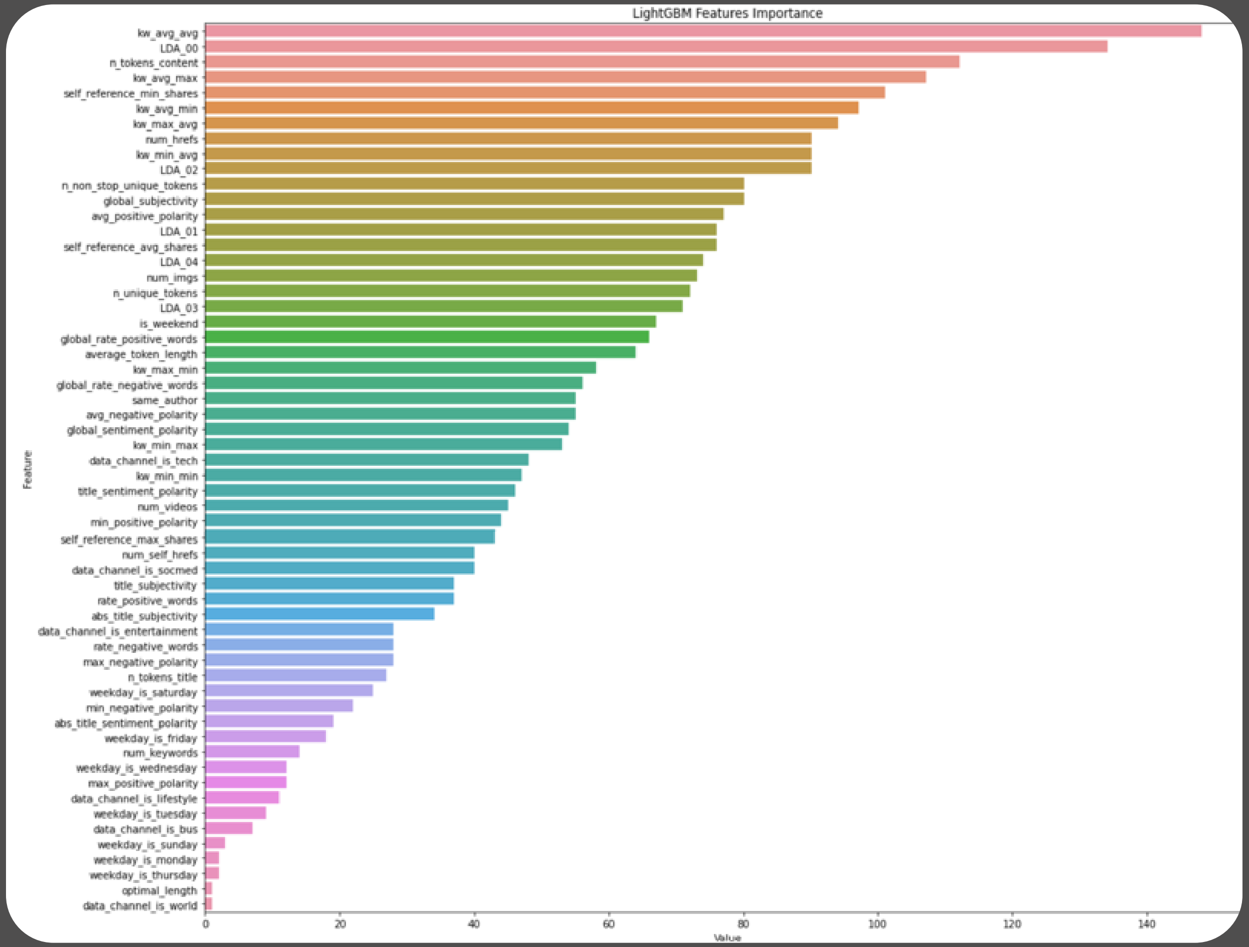
The Logistic Regression is a classification algorithm from *sklearn.linear_model* that computes the **probability** that an input X belongs to a class Y. This probability is found using the logistic function which only depends on two parameters. They are estimated via the **Maximum Likelihood Estimation Method** (MLE).

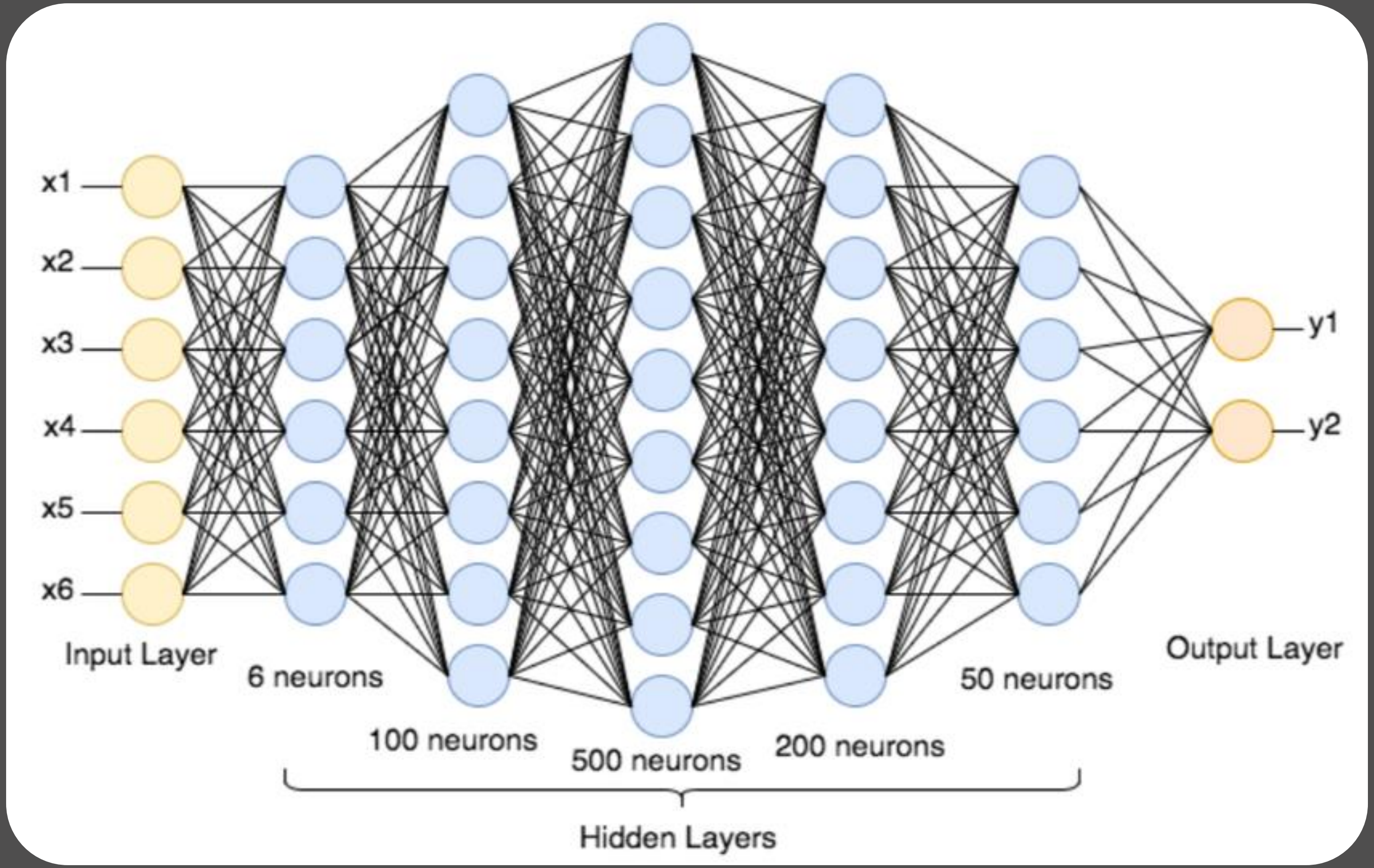
- **Really easy** to implement, interpret, and efficient to train.
- **Less inclined to over-fitting** even in high dimensional datasets thanks to the **Regularization** (L1 and L2) techniques.

ML & Deep Learning Models



LGBM is relying on the method of turning **weak learners into strong ones**. It is a **recent model** that is **performing in Kaggle competitions**, with even better results than *NN* or *XGBoost* algorithms.





A **neural network** is an algorithmic model composed of **several layers of computational units**, called neurons. Each neuron receives as input the outputs of the previous neuron while **continuously adjusting the weights** given to the values thanks to an error **back-propagation system**. Our NN here **isn't too complex** and will of course obtain increased performance with framework like *GPT-3*.

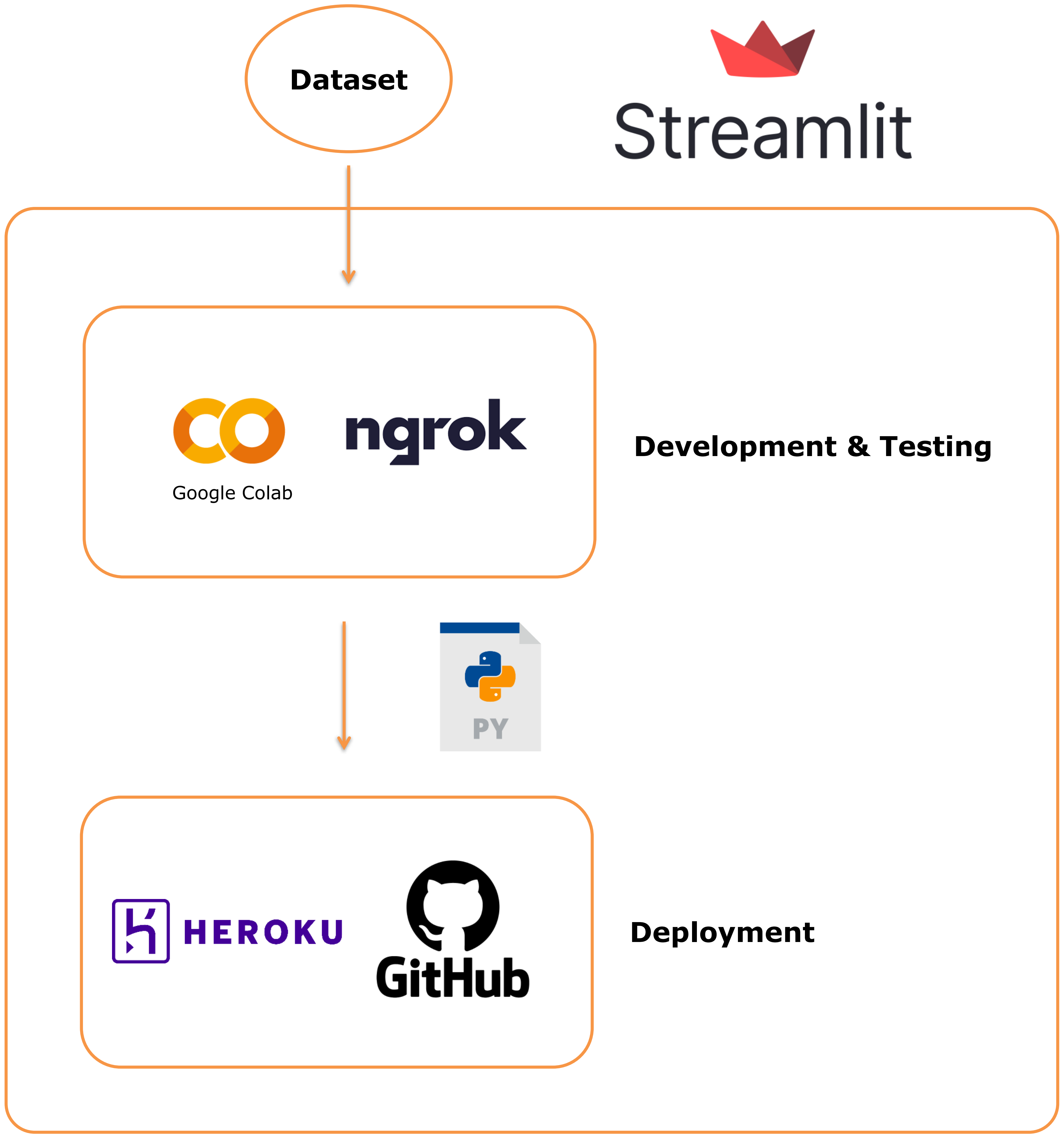
After performing LGMB, we can visualize the **relative importance of each features** and get a representation of how much it contributes to our model. Here the first **5 features** contains not surprisingly either the **keywords**, the **number of tokens** or the number of **self references** withing the article. We can also observe that the feature we created via web scrapping concerning the number of articles from the **same author is quite influent** too (#25) !

Building Streamlit App

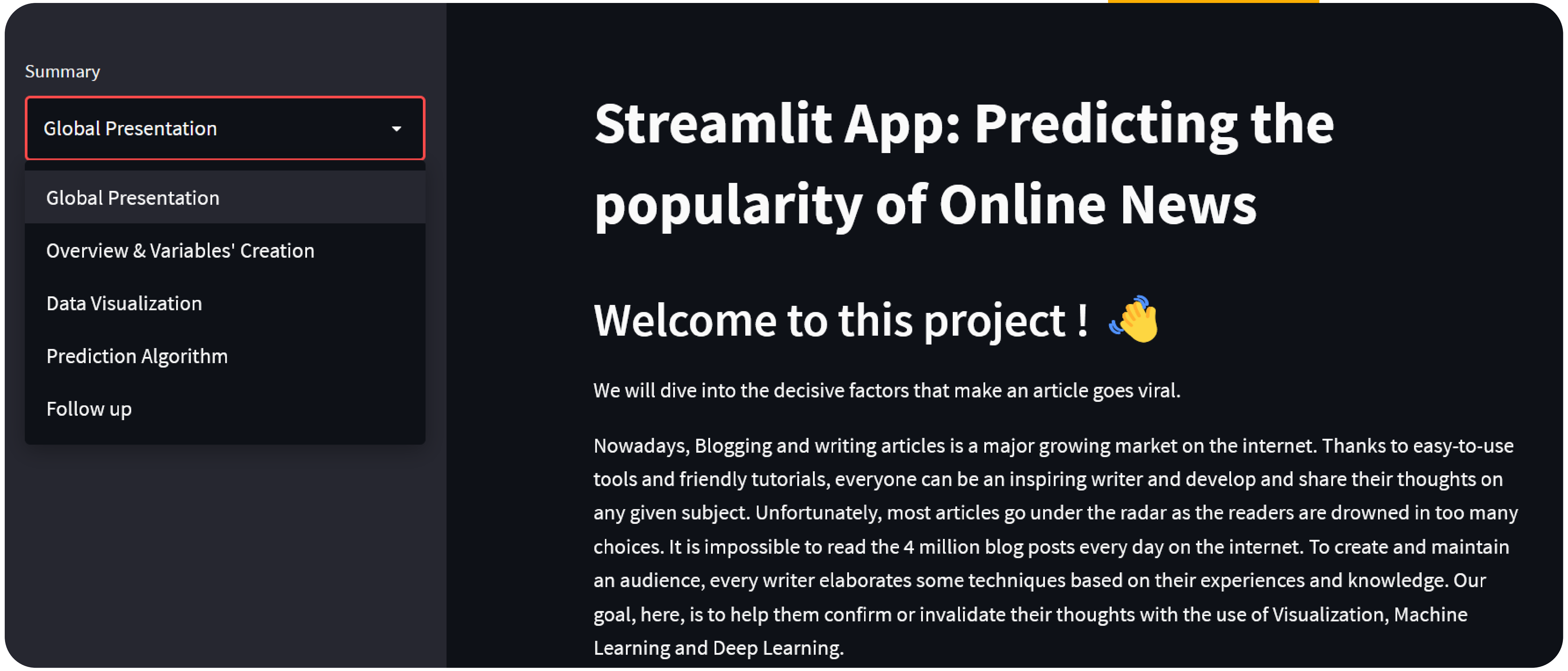
To showcase our work and develop an interactive API, we have decided to create a Streamlit App.

The **development phase** was made on a *Google Colab* notebook by installing the *Streamlit* and *Ngrok* libraries. *Ngrok* enables us to allocate a special tunnel to see and test the different functionalities implemented by *Streamlit* on our local machines. One of the main advantages of *Streamlit* besides its easy-to-use functions and documentation is the simplicity of the output containing only one Python file.

Then, when we were satisfied with the application, we needed to **deploy it online**. We used *Heroku* which is a cloud platform as a service (PaaS) to run and operate it. To do so, we wrote multiple files to link a *GitHub* repository containing all the data to the Heroku platform. The only task left was to make sure that no error occurs during the process and our code responds well.



Streamlit App : Predicting the popularity of Online News



Streamlit App: Predicting the popularity of Online News

Welcome to this project ! 👋

We will dive into the decisive factors that make an article goes viral.

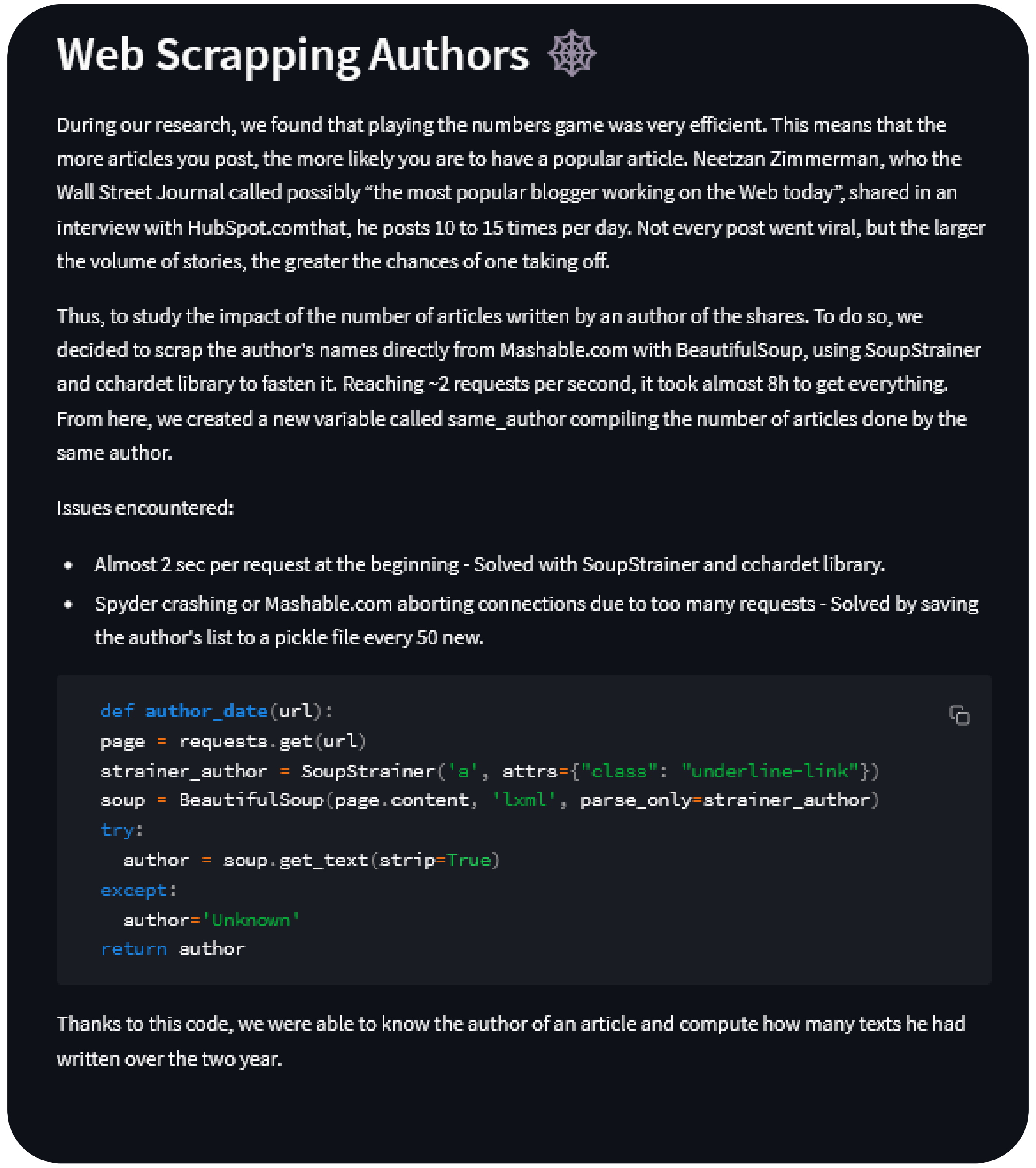
Nowadays, Blogging and writing articles is a major growing market on the internet. Thanks to easy-to-use tools and friendly tutorials, everyone can be an inspiring writer and develop and share their thoughts on any given subject. Unfortunately, most articles go under the radar as the readers are drowned in too many choices. It is impossible to read the 4 million blog posts every day on the internet. To create and maintain an audience, every writer elaborates some techniques based on their experiences and knowledge. Our goal, here, is to help them confirm or invalidate their thoughts with the use of Visualization, Machine Learning and Deep Learning.

Our app consists of **5 main** pages that you can access easily through a Summary.

Each page represents a part of the project and shows **how we adapt** to our data and **experiment** with different solutions. We have written all of our interpretations and hypothesis to enhance our understanding of the subject.

Overview Page

This page is the starting point of our project from our first encounter with the dataset to our tries to improve it with the creation of new variables thanks to web scrapping.



Web Scrapping Authors 🕸️

During our research, we found that playing the numbers game was very efficient. This means that the more articles you post, the more likely you are to have a popular article. Neetzan Zimmerman, who the Wall Street Journal called possibly "the most popular blogger working on the Web today", shared in an interview with HubSpot.comthat, he posts 10 to 15 times per day. Not every post went viral, but the larger the volume of stories, the greater the chances of one taking off.

Thus, to study the impact of the number of articles written by an author of the shares. To do so, we decided to scrap the author's names directly from Mashable.com with BeautifulSoup, using SoupStrainer and cchardet library to fasten it. Reaching ~2 requests per second, it took almost 8h to get everything. From here, we created a new variable called same_author compiling the number of articles done by the same author.

Issues encountered:

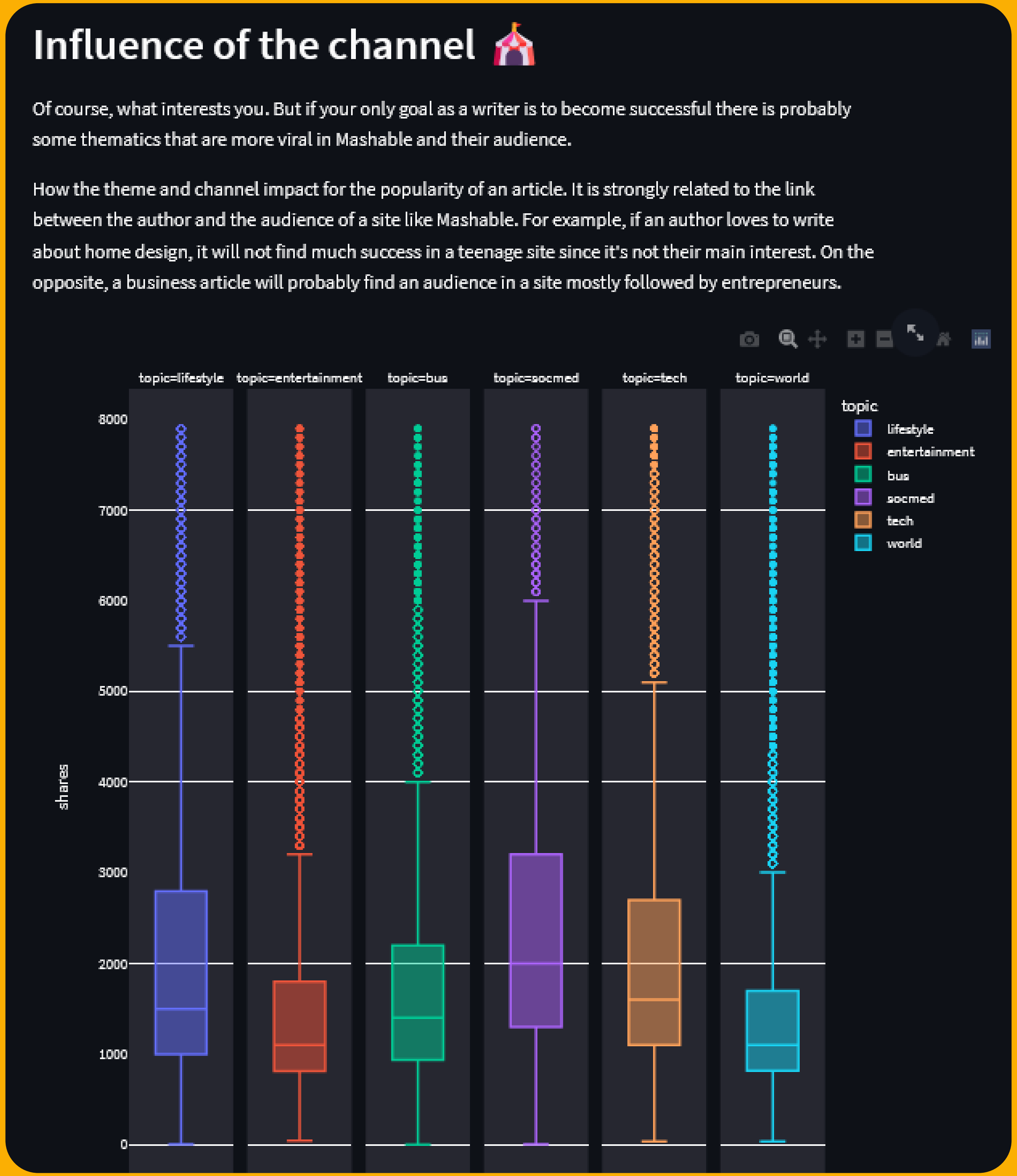
- Almost 2 sec per request at the beginning - Solved with SoupStrainer and cchardet library.
- Spyder crashing or Mashable.com aborting connections due to too many requests - Solved by saving the author's list to a pickle file every 50 new.

```
def author_date(url):
page = requests.get(url)
strainer_author = SoupStrainer('a', attrs={"class": "underline-link"})
soup = BeautifulSoup(page.content, 'lxml', parse_only=strainer_author)
try:
  author = soup.get_text(strip=True)
except:
  author='Unknown'
return author
```

Thanks to this code, we were able to know the author of an article and compute how many texts he had written over the two year.

Visualization Page

On this page, we grouped together most of our visualization regarding the impact of some variables on the number of shares. The different plots made with the Plotly library are interactive which is pretty convenient.



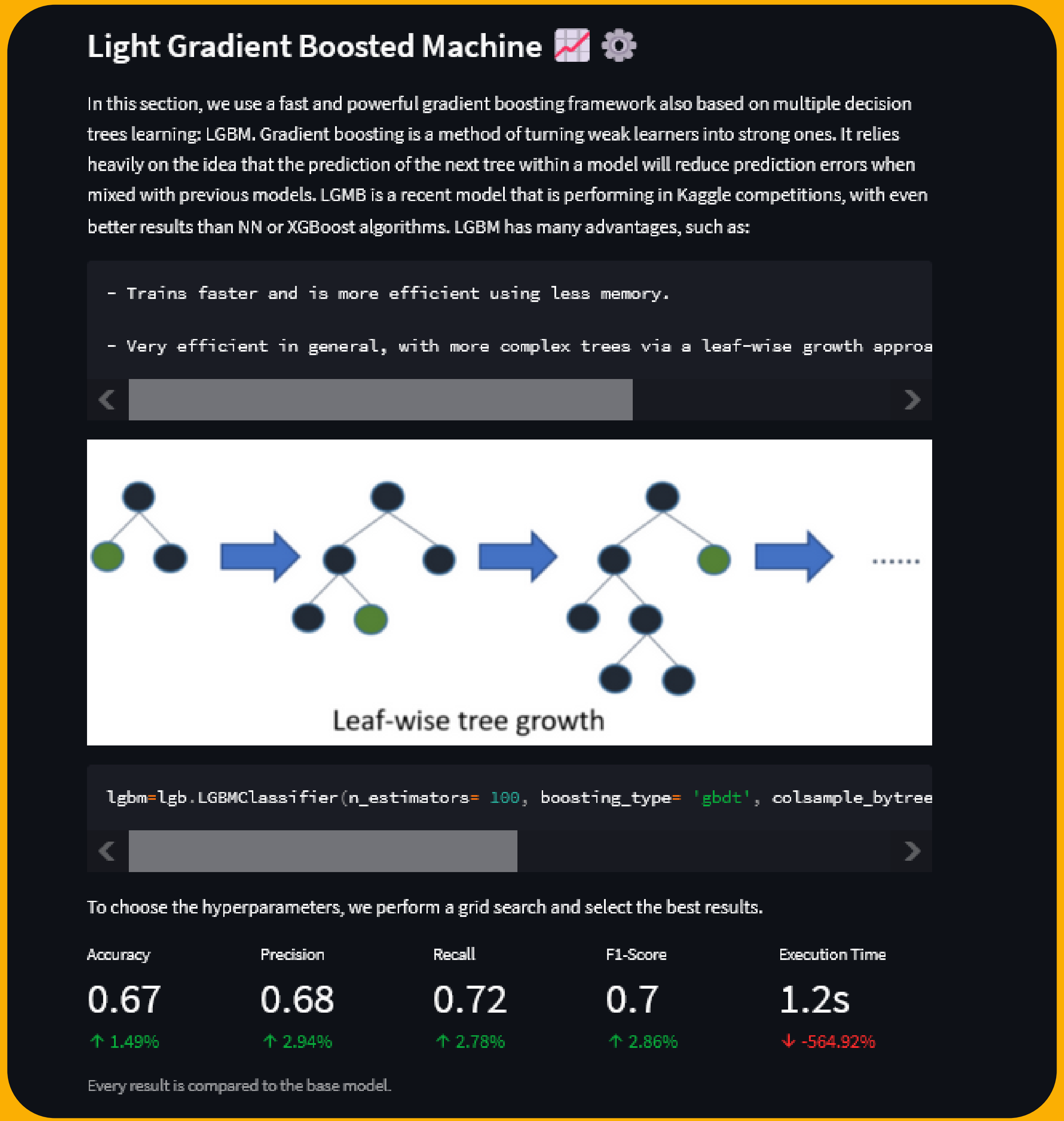
Influence of the channel 🎪

Of course, what interests you. But if your only goal as a writer is to become successful there is probably some thematics that are more viral in Mashable and their audience.

How the theme and channel impact for the popularity of an article. It is strongly related to the link between the author and the audience of a site like Mashable. For example, if an author loves to write about home design, it will not find much success in a teenage site since it's not their main interest. On the opposite, a business article will probably find an audience in a site mostly followed by entrepreneurs.

Prediction Algorithm Page

On this page, you will find the **results of the predicting models** used in the *Google Colab Notebook*. Each model is described with a text regarding its functioning, pros and cons along with an explicative image. We select a total of **4 Machine Learning** models (Logistic Regression, LGBM, Random Forest and SVC) and **1 Dense Neural Network**.



Light Gradient Boosted Machine 📈 ⚙️

In this section, we use a fast and powerful gradient boosting framework also based on multiple decision trees learning: LGBM. Gradient boosting is a method of turning weak learners into strong ones. It relies heavily on the idea that the prediction of the next tree within a model will reduce prediction errors when mixed with previous models. LGMB is a recent model that is performing in Kaggle competitions, with even better results than NN or XGBoost algorithms. LGBM has many advantages, such as:

```
- Trains faster and is more efficient using less memory.

- Very efficient in general, with more complex trees via a leaf-wise growth approa
```

```
lgbm=lgb.LGBMClassifier(n_estimators= 100, boosting_type= 'gbdt', colsample_bytree
```

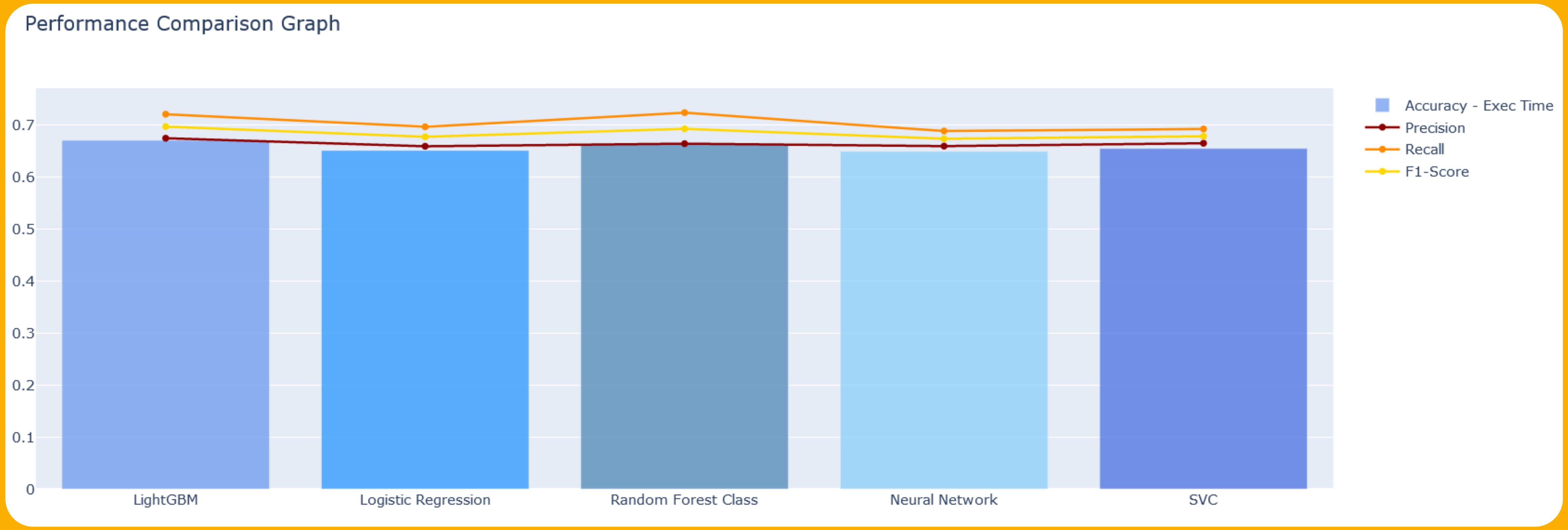
To choose the hyperparameters, we perform a grid search and select the best results.

Accuracy	Precision	Recall	F1-Score	Execution Time
0.67	0.68	0.72	0.7	1.2s
↑ 1.49%	↑ 2.94%	↑ 2.78%	↑ 2.86%	↓ -564.92%

Every result is compared to the base model.

We wrote the code lines to use those models and **display their performance**. We made a comparison with our base model to see the improvement or downgrade for our use case.

Finally, we made a final plot to compare the different performances between the models and as you can see **LGBM is clearly our best one**. To conclude, we explored the **different improvement possibilities** and why we need to be careful of those results due to notably being **the only source** of our dataset on a very limited period.



Conclusions

Nowadays, Blogging and writing articles is a **major growing market** on the Internet. Thanks to easy-to-use tools and friendly tutorials, everyone can be an inspiring writer and develop and share their thoughts on any given subject. Unfortunately, **most articles goes under the radar** as the readers are drown in too many choices. It is impossible to read the 4 millions blog posts every day on the internet. To **create and maintain an audience**, every writer elaborates some techniques based on their experiences and knowledge. Our goal, here, was to **help them confirm or invalidate their thoughts** with the use of Visualization, Machine Learning and Deep Learning.

Lessons learnt

We learnt a lot from this project. Not only in the **Datascience field** with learning how to use Streamlit, deploy an app on Heroku, manage to webscrap a sizable dataset and manage outliers. But also in **Team-working skills** and how to listen and trust each other on such a project.

What's next ?

- **Improvement of the dataset** in terms of period and sources to avoid audience bias.
- **Interactive and real-time prediction** directly from a written article since our goal is to help writers who from most of them have no background in Machine Learning and NLP techniques.