REPORT - CHECKPOINT 3

Project: generate real time stocks prediction of a company based on Twitter posts

0. Workload distribution

Team member	**CP2**	**CP3**	**Final CP**
Clémentine	Exploration	Topic clustering	Presentation
Daniel	Modeling	Improving model	Deployment
Alexandre	Modeling	Scraping	Presentation
Gabriel	Modeling	Scraping	Presentation
Charlotte	Exploration	Topic clustering, scraping	Write-up

1. Overview of our project

1.1 Project name

Using sentiment analytics on Twitter to forecast Apple's stocks (AAPL)

1.2 Problem definition

Stock market prediction has attracted a lot of attention from academia as well as business lately. One of the questions still being raised is that : **can we really predict stocks market or is it random ?**
Behavioral economics indicate that emotions can deeply impact individual behavior and decision-making. By extension, we can wonder whether the public mood is correlated or even predictive of economic indicators?

Here, we make the hypothesis that emotions can indeed have an impact on investment decisions and therefore on stocks of a company. Our objective through this project is to use sentiment analytics on social media - and focus on Twitter posts - to forecast a company's stocks like Apple.

If this would work, we truly believe this could come up as a very useful tool to get real time trend predictions over stocks and to help make decisions on this stocks investments (like for example on the US Robinhood app).

1.3 Achievements

We decided to first focus on predict the sentiment analysis on tweets about Apple, which is a building block of the stocks prediction:

- EDA and preprocessing of dataset
- Train and evaluate sentiment analysis on our training set
- Improve accuracy for the sentiment analysis
    - Our first maximum accuracy was 68% by tweaking baseline models like SVM and Logistic Regression.
    - We reached 75% by using a pre-trained BERT model
    - We managed to reach 80% by tuning hyperparameters (changing the number of epochs)
- Predict on test set : dataset of Apple-related tweets
- Scrape live tweets on Apple using Twint library

1.4 Next steps

We now have a dataset historical tweets of Apple and the sentiment predicted on it which is becoming our training set.
We also have scraped live tweets on Apple which is becoming our test set. The next steps would be :

- To scrape stocks prediction on Apple from the same dates as our training dataset
- To join the stocks dataset with our sentiment dataset and get one final training dataset
- To train our dataset using VAR and LSTM time series
- To predict the stocks of Apple on the test set

2. Technical description

2.1 Dataset

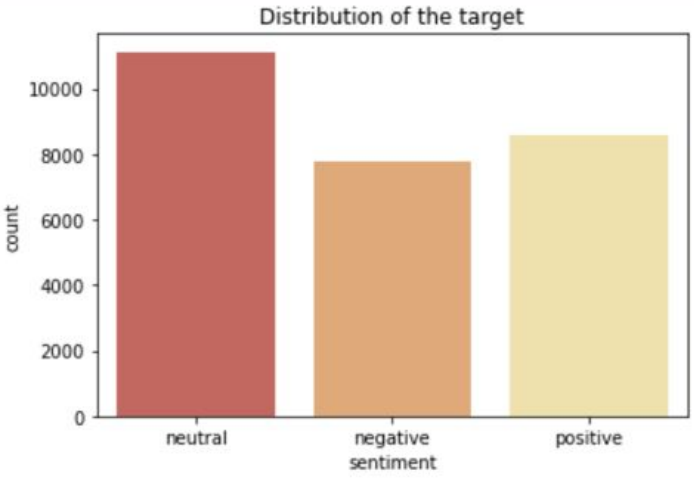
The training dataset we used was extracted from Kaggle :
https://www.kaggle.com/c/tweet-sentiment-extraction/data
It is a dataset with 27481 tweets, with a label sentiment of 3 classes : positive, negative and neutral.

A tweet is composed of a text of, at most, 280 characters (depending on the year that the dataset was created) and it can contain a lot of words that are not in the English dictionary. That is because tweets are written, most of the time, by users in an informal way. Besides that, a tweet could contain mentions, hashtags and retweets, which pollutes even more the text.

Based on that, in order for our tweets to be served as **input** to a model, a bunch of different *preprocessing* steps have to be made.

2.1 EDA and Preprocessing

2.1.1 Target




```
[ ]: 1 df[target].value_counts(normalize = True )
```

executed in 12ms, finished 16:44:18 2021-01-24

```
[ ]: neutral     0.404570
     positive    0.312288
     negative    0.283141
     Name: sentiment, dtype: float64
```

The labelled dataset we used for this step is quite balanced. The more represented class is neutral with 40% after we have 31% of positive tweets and 28% negative one

2.1.2 General statistics & Nan values

Shape: (27481, 4)
We have a dataset with 27481 tweets.

```
]:  1 df.info()
```

executed in 21ms, finished 16:47:57 2021-01-24

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27481 entries, 0 to 27480
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   textID         27481 non-null  object
 1   text           27480 non-null  object
 2   selected_text  27480 non-null  object
 3   sentiment      27481 non-null  object
dtypes: object(4)
memory usage: 858.9+ KB
```

We will drop the feature 'selected_text' since it is not useful on our prediction problem. Also, we can see that we have only one null value (row 314) that we will drop. Since it's only one row, it won't have an impact on our dataset.

2.1.3 Basic preprocessing to perform EDA

To perform our EDA, we needed to use some basic preprocessing steps:

- Keep only words with letters, numbers and basic punctuation (!, ?, :, @, ...)
- Lowercase every word
- Lemmatize every '****+' with the lemma -INSULT-

1.5 Total words

In the entire dataset, we can find the following repartition of words :

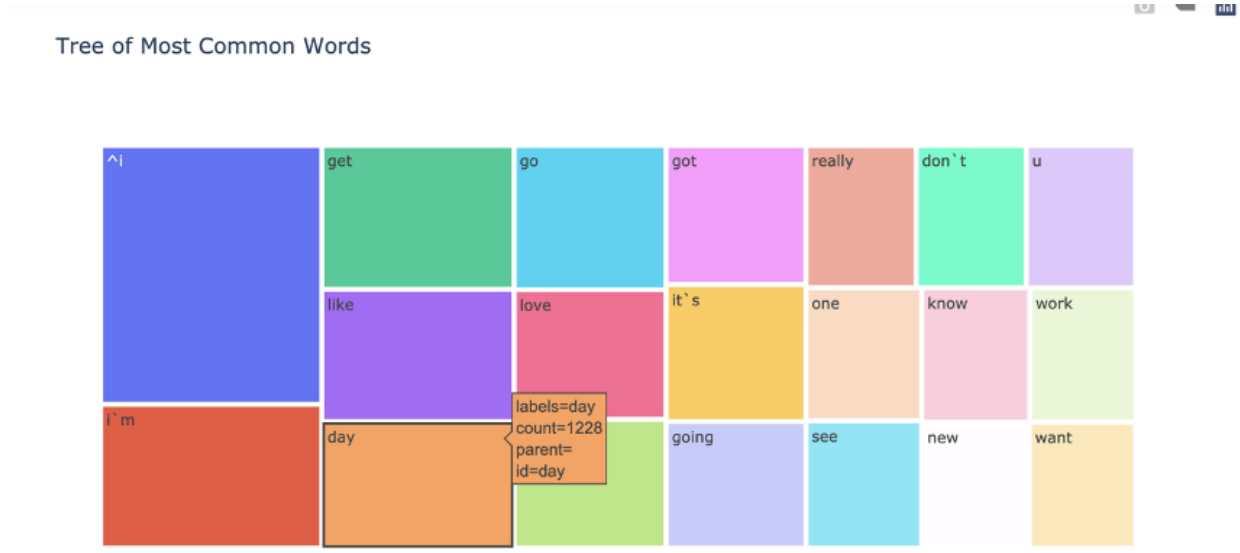
- Total words = 352037
- Stop words = 126708
- Other words = 225329
- Unique words = 52167

We can see that 35% of the words are stop words.

2.1.4 Most common words

To explore our data, we took a look at the most common words present in the tweet of our dataset.

First of all, we removed the English stop words using the nltk library. As we noticed before, the stop words in our dataset consist in 35% of the total of our words.



Among most common words, we can see there is the 'I' pronoun that starts a tweet.

Then, we observed the most common words in tweets labelled as positive to get a sense of the importance of features for the next steps:

Tree of Most Common Positive Words

We can see that there are words like happy, hope, mother, great, good, love which makes total sense. The tab below shows the occurrence of each of these words.

	Common_words	count
0	love	781
1	^i	730
2	day	704
3	good	644
4	i`m	434
5	happy	404
6	like	391
7	^happy	386
8	great	361
9	get	361
10	hope	346

Finally, let's take a look at tweets labelled as negative. As mentioned before, we replaced all the insults '****+' by the lemma 'insult'.

Tree of Most Common Negative Words

Here, there are words like miss, don't, can't, hate which makes sense in negative tweets. The following tab shows the occurrence of each of these words.

	Common_words	count
0	^i	1032
1	i`m	636
2	like	462
3	get	428
4	miss	409
5	go	366
6	insult	321
7	don`t	314
8	it`s	301
9	can`t	300
10	going	299

Finally, we observed the most common words for neutral tweets:

Tree of Most Common Neutral Words

]:

	Common_words	count
0	^i	1113
1	get	607
2	i`m	539
3	go	503
4	like	433
5	got	407
6	it`s	369
7	going	366
8	know	333
9	day	329
10	see	327

This concludes our EDA for this first step.

2.1.5 Basic Tweet Cleanup

The first step is to clean the tweet by removing all the hashtags, mentions, retweets and also URLs of the text. With this, we end up with the main information inside the tweet so the model can focus on it and won't get confused with text that is not relevant to transmit the sentiment of the tweet.

2.1.6 Insults Replacement

In this dataset any kind of insult in the tweets are represented by four stars or more (* * * *).

The goal of this step is to manually "lemmatize" them into -INSULT- , the same way the pronouns were replaced by -PRON- in the *Lemmatization step*.

INPUT: Sons of ****, why could not they put them on the release we already buy

OUTPUT: Sons of -INSULT-, why could not they put them on the release we already buy

2.1.7 Contractions Extraction

A contraction is a shortened version of the written forms of a word, syllable, or word group, created by omission of internal letters and sounds. For instance, a contraction of the group of words "I have" is "I've".
For our lemmatization step to be working as intended, we need to expand these contractions to their *original* words.

In order to accomplish this, we used a function that looks for such contractions and replaces them:

INPUT: *I've* been sick for the past few days and thus, my hair looks weird. if *I didn't* have a hat on it would look...

OUTPUT: *I have* been sick for the past few days and thus, my hair looks weird. if *I did not* have a hat on it would look...

2.1.8 Remove emojis

There are some tweets in the dataset that contain emojis. Emojis are ideograms and smileys used in electronic messages and web pages. For now, we decided to remove all emojis from the text, but we believe that they can be relevant to classify the sentiment since most of the emojis used represent emotions.

2.1.9 Remove words with numbers

Since tweets are mostly informal written text, some words might contain numbers in between the letters, which could not be understandable by a model since it doesn't exist in the english dictionary. So we applied a regular expression to every tweet to remove every word that contains numbers in between the letters.

2.1.10 Lemmatisation

Lemmatization consists in determining the **lemma** of a word based on its intended meaning.

INPUT: I have been sick for the past few days and thus, my hair looks weird. if I did not have a hat on it would look...

OUTPUT: I have be sick for the past few day and thus , my hair look weird . if I do not have a hat on it would look ...

2.1.11 Remove punctuation

Punctuation is present in every text written in english. So words in the text can be followed by punctuation and therefore the model would identify a word followed by a punctuation as a different word. For instance "hi" would be treated differently as "hi.". We applied a regex to remove all the punctuations of every word in the tweets.

2.1.12 Tdidf Vectorizer

This step consists in creating a new dataset where each row is a tweet and each column is a unique word present in all of the documents of the dataset. For each tweet, values are assigned to each word present in the tweet. The value assigned to each word, takes into consideration the number of occurrences of the word in the tweet and in all of the documents of the dataset. After this step, we end up with a new dataset with more than 20000 columns, because every unique word in all of the documents is represented by a column.

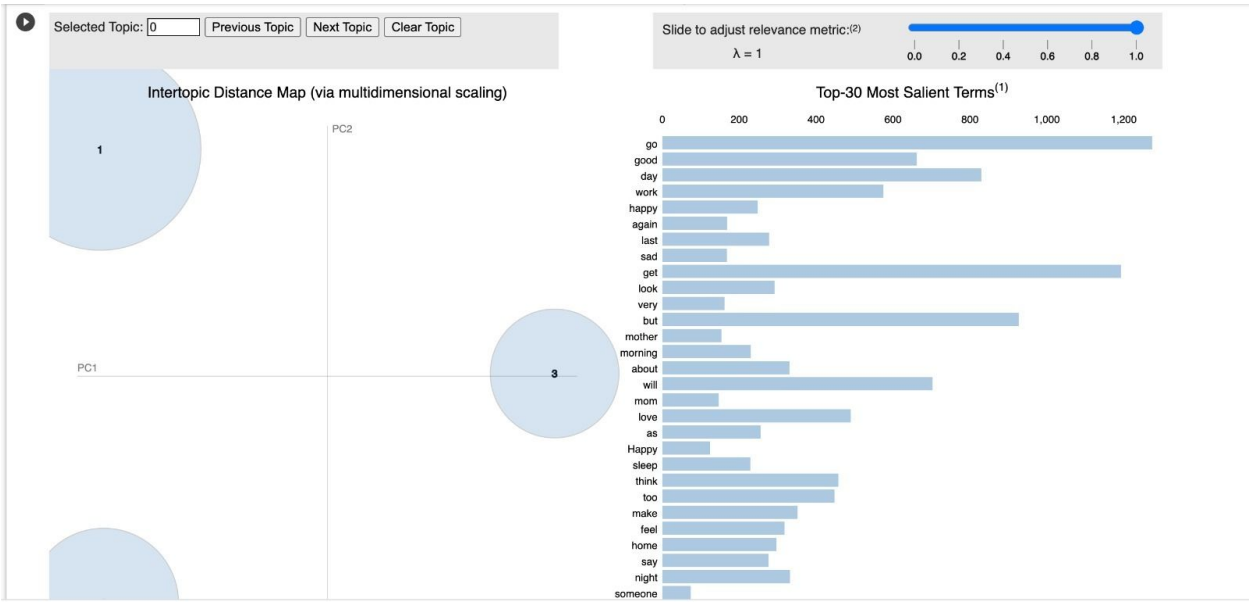
2.1.13 Topic modelling

First of all, we used Gensim's dictionary constructor to give each token in our preprocessed tweet dataset a unique integer identifier.
Then, we created a bag of words to count how many times each word token occurred in a tweet. We used gensim method doc2bow to output a vector for each tweet in the form of (word id, frequency of word occurrence in document).

Then we're ready to train our LDA model. LDA is an unsupervised model that takes documents as inputs and assumes that each document is a mix of a small number of topics and that each word is attributable to one of the documents' topics.

We fit our LDA model by choosing 3 topics. At the very first step, we tried 10 topics, then 5 topics, to realize that there were 3 clear clusters, probably related to the fact that this dataset is sentiment analysis related.

Below is our visualization of clusters with the 30 most salient terms :



2.2 Baseline Model

Now that we have well preprocessed data, we are allowed to input it into a *Machine Learning* model.

Since we have three classes (positive, neutral and negative) we use the method OneVsRest to classify the tweets.

We tried 3 different models from the sklearn library:

1. Logistic Regression

Logistic regression applies a sigmoid (or in this case softmax) function to a linear regression to predict the output. It will return a value from 0 to 1, generally with a threshold of 0.5 and classify probabilities above the threshold as 1 and below as 0.

2. SVC

The SVC algorithm implemented by sklearn for multiclass problem uses a one-vs-one scheme. It basically tries to find the best hyperplane that could separate our data between the 3 classes..

3. GaussianNB

Gaussian Naive Bayes classifiers are a subfamily of Naive Bayes classifiers that deal with continuous data. The assumption is that the continuous values associated with each class are distributed according to a normal distribution.

Even after hypertuning each model and performing a model selection, we couldn't find an accuracy above 70% (the maximum was actually 68%).

2.3 Final Model

Because of our limited accuracy, we decided to move on to deep learning models and especially transformers, that are state-of-the art right now in NLP. We tried to use transfer learning from a pre-trained model of Bidirectional Encoder Representations from Transformers (BERT). BERT was created and published in 2018 by Jacob Devlin and his colleagues from Google.

We used BERT to tokenize the tweets and then transform them into a special vector format so we can pass them to the transformer model.

Doing that we were able to achieve a validation accuracy of 75.5%.

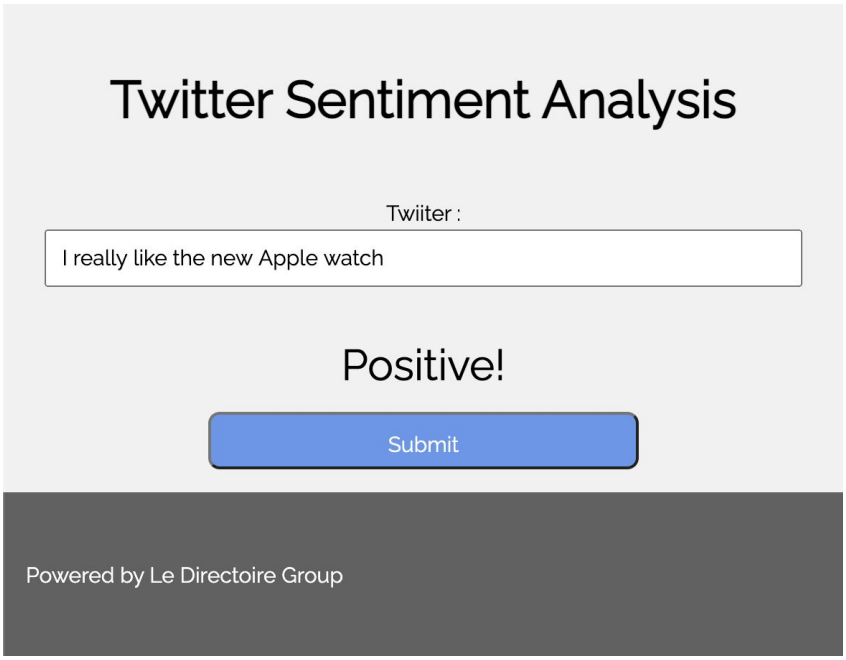
Then we tried to tune our model, by changing the number of epochs to 30 with EarlyStopping with patience = 5 and restore best weights.
Our final accuracy achieved **80%.**

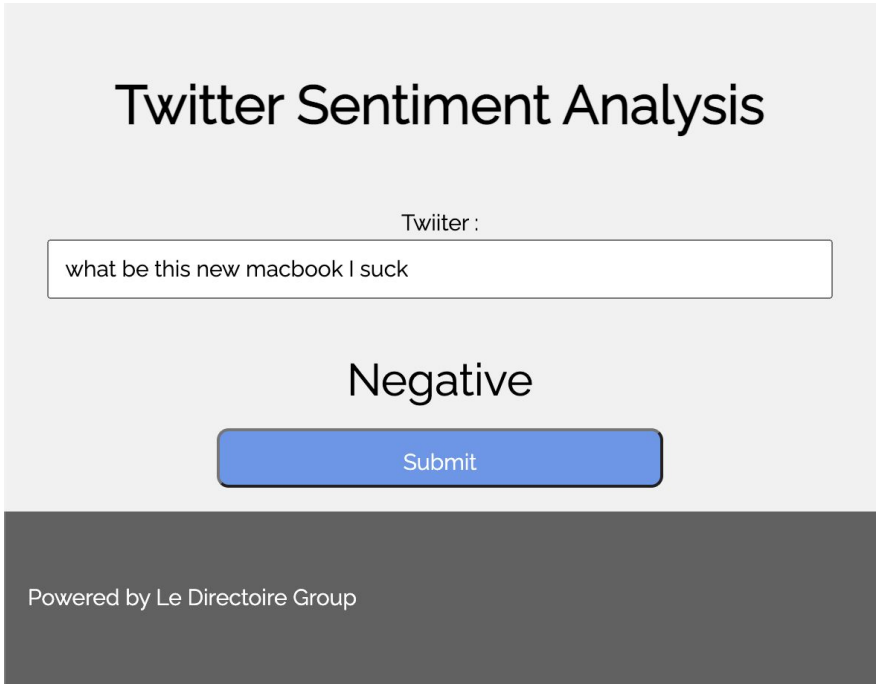
2.4 Example of predictions

The test dataset we used was extracted from and consist in 118,350 Apple tweets from the year of 2016. Examples of preprocessed tweets we had and that we predicted :
"Official Apple watch magnetic Charging Dock use once alexjone"
-> Predictor : Neutral
"the Apple employee find dead at the company be headquarters shoot -PRON- authority say ..."
-> Predictor : Negative
"Apple ’s plan to turn the iPhone into a medical wonder start now"
-> Predictor : Positive

2.5 Deployment

Using Flask we deployed our model :





3. Discussion

3.1 Challenges

We had to face different challenges through the past weeks, the two biggest being :

- Sentiment analysis algorithm :

Improving the accuracy of our sentiment analysis: we realized while reading papers that baseline models would never reach a very high accuracy and that actually sentiment analysis accuracy is quite low on average. To solve this, we decided to use state-of-the-art models . We found out that BERT, which is a transformer NLP model, was able indeed to achieve better results in this task. But we still could not get higher than 80%. That might be because of our dataset that is not big enough and also there are many things to try in order to improve it such as preprocessing the tweets in a different way that we have.

- Dataset preparation for the stocks prediction

We found it difficult to scrape tweets related to Apple : there was a limitation on Twitter public API on the number of tweets per company. To solve this, we used Twint library to scrape as many tweets related to Apple from this week. However, it was tricky to distinguish tweets about the fruit apple from tweets related to the company. There were not enough specific parameters on the library to filter out irrelevant twitter. This would need some preprocessing for the further steps.

3.2 Further steps

In conclusion, at this step of our project, we managed to get the first building block of a more long-term project that would be to predict the stocks of Apple in the future based on sentiment analytics on Twitter. We have a supervised model trained on a sentiment analysis dataset that could predict the sentiment of tweets related to Apple with 80% accuracy.

There are further steps that we would complete in order to achieve our final objective :

- To improve our sentiment analysis we could:
  - Take a test set that we label ourselves to test our model
  - Add more samples (larger dataset)
- Perform the join of our datasets of sentiment analysis and stocks predictions
- Predict stocks using LSTM and VAR : this combination would be interesting to try because we're facing the problem of multivariate time series : multiple features are used to predict the stocks. We would try to use this VAR, which is a multivariate extension of ARIMA and a simple LSTM structure. This would not be considered as an ensemble model : we use the ability of VAR to filter and study historical data and provide benefit to our neural network to forecast the stocks