**CCT College Dublin**

**Assessment Cover Page**

*To be provided separately as a word doc for students to include with every submission*

|  |  |
| --- | --- |
| **Module Titles:** | *Advanced Data Analytics*  *Big Data Storage and Processing* |
| **Assessment Title:** | *MSC\_DA\_CA2v4* |
| **Lecturer Name:** | *David McQuaid*  *Muhammad Iqbal* |
| **Student Full Name:** | Gustavo Henrique Nascimento |
| **Student Number:** | 2022313 |
| **Assessment Due Date:** | *26/05/2023* |
| **Date of Submission:** | *31/05/2023* |

**Declaration**

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

# About the project

This project set out to do sentiment analysis, forecasting and a dashboard with the 1-year data of Twitter, between the dates of 01/11/2021 and 31/10/2022, using the search of tweets with the words "vaccine", "vaccination", and "vaccines", more than 1.4 billion tweets were analyzed, and resulted in a data frame with 763,267 tweets. In this way, this report was written, explaining the steps made in the project. However, due there are restrictions on the number of words, the report can’t go into detail, though, the Jupyter notebooks used have more details and were made available along with this report and its GitHub.

For the purposes of managing activities and saving time in executing the codes, this project was divided into 5 parts, namely:

- Environment settings and resource analysis.

- Data collection.

- Sentiment Analysis.

- Forecast.

- Dashboard.

# Data

This project uses data made available by the Internet Archive Project. The internet Archive is a non-profit project that is building a digital library of the internet, with websites, books, newspapers, videos, images, and various other artifices around the world, with the aim of preserving the history of democratizing access to information on the Internet. all. With an impressive 99+ Petabytes of server space per copy, they store at least two copies of everything. The data from the Internet Archive is under the Creative Commons license, which allows public use of the data for non-commercial purposes, and provided that the copyrights of the original source of the data are respected. (*Internet Archive Terms of Use*, no date). As Internet Archive data is a copy of Twitter data, and Twitter data is considered public, therefore, there are no restrictions on the use of this data for the research proposed purposes.(*Twitter - Rules and policies*, no date)

# Virtual environment settings and resource analysis

To execute this project, a new environment was created, as the initial one, which has been configured in the classroom, couldn’t process all data gathered to the research, and was constantly failing and/or crashing. So, a physical computer had to be prepared with the following software:

- Ubuntu (System Operation)

- Anaconda (Jupyter Notebook)

- Python (libraries installed as needed)

- MySQL

- Mongo DB

- Hadoop

- Spark

This environment migration took time since the entire environment and the programs had to be installed and configured, because this impacted on the project, a note was made necessary.

# Benchmarking

The Benchmarking tests were done using the YCSB tool, following the tutorials passed in the classroom. Some challenges were faced at this stage, such as the configuration of Mongo DB that not accepting the use of the connection with the Benchmarking tool. However, as this is a project requirement, the test was performed in the MySQL and Mongo DB databases, using workloads A through F. In all situations, Mongo DB performed better than MySQL.

Gráfico, Gráfico de barras

Descrição gerada automaticamente

Figure 1 - YCSB - Runtime indicators

As seen in Figure 1, MySQL averaged 19,806 ms while Mongo DB 1,654 ms at the 6 workloads runtime. Mogo DB also gets better results when analyzing Throughput on all workloads, as seen in the Figure 2.

Gráfico, Gráfico de barras

Descrição gerada automaticamente

Figure 2 - YCSB - Throughput indicators

The table below displays all results generated in the analysis.

Interface gráfica do usuário, Aplicativo, Tabela, Excel

Descrição gerada automaticamente

Table 1 - YCSB Results

# Data collection

This project has collected data from Twitter, between 01/11/2021 to 31/10/2022 (365 days) using the Internet Archive Project’s API. The Internet Archive project makes Twitter data available by API following a structure, where the main object is an “item”. In this case, the “item” corresponds to a month, like a folder, inside each “item” there is a compressed file by month’s day, and inside each day there is a collection of compressed json files, segmented by minute, with all the tweets from around the world at that minute.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Figure 3 - illustration of the tweets extracting process using the Internet Archive API.

Thus, each item has approximately 30 tar files, each tar file has approximately 1,411 json files, and each json has an average of approximately 2,742 tweets (so, we can conclude that in the analyzed period approximately 2,742 tweets were published per minute). At the end of the year, this has generated a volume of 917.96 GiB (2.51 GiB per day), corresponding to 1,412,214,396 tweets (3,869,081 tweets per day).

In order to collect, store and process the entire volume of data generated in the project, a code was made that could download the file per day, unzip the file, read the json files, select only the tweets that were related to the project and then store the necessary features in the database, as illustrated in Figure 3.

The average time to download the files was around 47 minutes per day, which would make the data collection phase of the project take approximately 12 days. Because de short projects frame time, the collection code was duplicated, and data collection was carried out simultaneously, running more than one Jupyter file at the same time. However, due to API server limitations and possible internet connection issues, the connection to the API was cut off and downloads stopped frequently. Hence, to guarantee the integrity of the collected data, avoiding the collection of duplicate files and/or loss of data, as well as the extension of the project collect phase, a database (table) was created to control the files processing.

The processing of the tweets consisted of filtering the tweets only in English and then searching the tweet text for the words “Vaccination”, “Vaccine” and “Vaccines”, if the tweet had any of these words, the tweet was stored in the tweets database, as illustrated in the Figure 4.

Desenho com traços pretos em fundo branco

Descrição gerada automaticamente com confiança média

Figure 4 - illustration of processing Tweets

The control took place as follows, the code requests information to the API about the item, the API returns a list of files (the days) that are inside the item, then the code query the database (control table) to find out which of those day has not yet been processed , when finding a day that has not been processed, it downloads the day's file, saves the file metadata in the control table (name, size, and download date time) and then processes it by filtering the tweets and storing the targeted tweets in the database (tweet table). It also writes into control table the json file metadata: size - Bytes, Number of tweets - total, and number of tweets corresponding to the search (targeted tweets).

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Figure 5 - Select from Control table.

After all the json files has been processed, the code updates the control table with the datetime the day’s file was processed and deletes the file from the computer.

However, the API was dropping parallel connections, so a second moment of the data collection phase was necessary, the code was adapted for use in Google Collab. As shown in Figure 6, instead of controlling and storing tweets on the local MySQL database, the cloud code started to create two CSV files in Google Drive per item (month) processed, one for the control and the other for storing the filtered tweets. The fact that the CSV files are in Google Drive allowed resuming the data collection, for the item being processed in the cloud, even after the Google Collab environment was turned off due to the time of use. Meanwhile, the local code was modified to perform an error check, waiting for 10 seconds, and trying to download again if the connection dropped (the code does this up to 5 times).

Uma imagem contendo Aplicativo

Descrição gerada automaticamente

Figure 6 - Moments of data collection

After collecting the data in the cloud, a function was created in the code to send the data to the local MySQL database. Even so, data collection took 7 days and 23 hours, a reduction of 4 days of the initial estimate. After all items have been processed, the database of filtered tweets had 763,267 tweets, an average of 2,091 tweets analyzed on the subject per day. The third and last moment of this part was the migration of the MySQL database to Hadoop, the files were saved in parquet format, segmented by month, through the Jupyter notebook.

At first, MySQL was chosen for this project due to the need for constant updates and selects in the database, so the database should allow changes and be fast, otherwise this project’s phase would take even longer. Another factor that influenced the use of MySQL was the familiarity with the tool, this allowed time savings in the elaboration the collection and processing code. After data collection, the data were migrated to Hadoop in parquet format, as changes would no longer be made directly into the tables, and they would be used only for queries. Hadoop was also chosen for its ease of use with Spark.

# Sentiment Analysis

Using Apache Spark the data were loaded into a Spark data frame where it was used for sentiment analysis. Was performed 3 sentiment analyzes, as described below:

## Machine Learning:

A Logistic Regression Classifier model was made, model training used a movie review dataset, available on ardianumam's GitHub (2022). The model performed an accuracy of 98.32% with test dataset (20% of movie review dataset)

To training and testing the model, the tweet text was cleaned up, removing usernames, links, special strings, RT tags, and extra spaces. After the clean text, it was prepared for the model, with tokenizer, stopwords removal and hashing transformation. Then these features were used to classify the database tweets and a column with the model’s classification values was added to the spark data frame.

Tabela

Descrição gerada automaticamente

Figure 7 - Testing results Logistic Regression Classifier model

## Textblob

Textblob is a textual data processing library that can be used for sentiment analysis. It returns a score between -1 and 1 as a sentiment rating for a sentence. Although Textblob is a tool capable of analyzing the sentiments of a tweet without the need to clean it, since the text of the base tweets was already cleaned in the Machine Learning stage, the text used in this stage was the clean text. The TextBlob score was also stored in a column.

## Vader

“VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media” (Hutto and Gilbert, 2014) , just like Textblob, VADER returns a score between -1 and 1. Using the same clean text, VADER was used to do the sentiment classification, a column was also added to the Spark data frame with the score generated in VADER.

Once the 3 analyzes were completed, a new score was made using the 3 analyses. For this, a weighted average was made, where the Textblob and VADER analyzes received a weight of 1.5, since these libraries are specialized in this type of analysis, the confidence in their results is greater than that performed by Machine learning that was trained with a database of movie reviews and not on social data, which is why ML was given a weight of 1 in the weighted average, as shown in Figure 8. This new score was saved in the Spark data frame and the database with these scores was stored in Hadoop, using the partitionBy function to segment it by month.

Uma imagem contendo Tabela

Descrição gerada automaticamente

Figure 8 - Sentiment Analyses on Spark Data frame

Since the theme is vaccination and the period comprises a short time after the start of COVID vaccination, in the next steps it is possible to see that at the end of 2021 there were a lot of people talking about it. However, the volume reduced close to March 2022 and remained stable, this stability is one of the reasons the data were considered stationary. Also notice that the growth of positive and negative tweets is close, which appears to be a correlation between them. More details can be found in the next section’s graphics.

# Forecast

Once more, using a spark data frame, the score created from the 3 sentiment analyses was transformed and created a new feature called sentiment, a binary column, where scores above 0 were defined as “1” (positive) and equal to or below zero as “0” (negative). From there, a new Spark data frame (called tweet\_hour), was created. It contains the count of tweets and the mean of customized score per hour, segmented by feelings (positive and negative), as can be seen in Figure 9.

Tabela

Descrição gerada automaticamente

Figure 9 - "Tweets\_hour" - New Spark Data frame

As the tweets\_hour would be the database for training and testing the models, an initial analysis of the data was made and as observed in the Figure 10 there is no nominal distribution in the data.

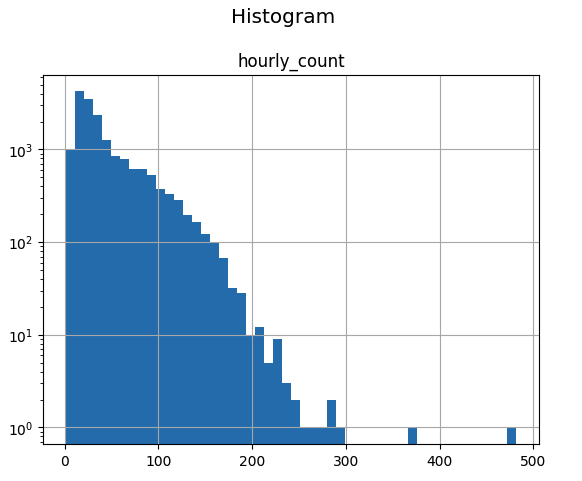


Figure 10 - Tweet\_hour – Histogram

Initially, to forecast, tools, and libraries compatible with Spark Data frame were used. As demonstrated in the next subsections, these tools had to be discarded and the forecast was performed with Pandas Data frame. The database (tweet\_hour) was split between the first 11 months to train (91.5%) the model and the twelfth (8.23%) was used for testing (Figure 11 and Figure 12 illustrate the split, segmented by sentiment).

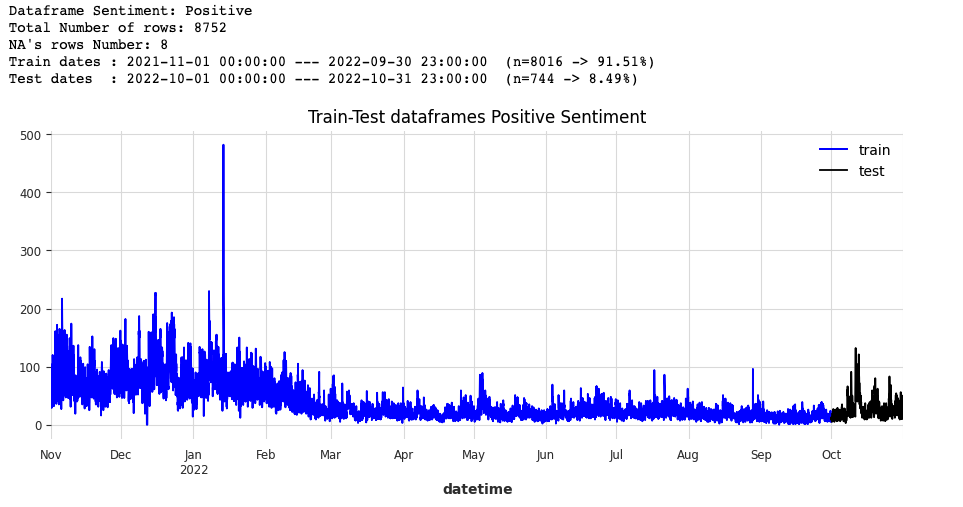


Figure 11 - Training and test data distribution graph – Positive sentiment

Gráfico

Descrição gerada automaticamente

Figure 12 - Training and test data distribution graph – Negative sentiment

## ARIMA

The first attempt used the ARIMA model from the Statsforecast library. With a confidence level of 90% the model did not perform well, it obtained an RMSE of 25.07 and a MAPE of 55.32%. This solution also proved to be very time consuming.

## ML – Classification Models

For the use of these models, a pipeline was created in Spark to prepare the database, with the following steps: string indexer, One Hot Encoder, and feature assembler. Once this pipeline was created, it was possible to create two classification machine models: Decision Tree Regressor classification and GB Tree classification. As with ARIMA, these models were trained with data of the most recent month in the database and then tested with this month database (Figure 11).

The two Models performed well, the Decision Tree Model obtained the RMSE Score of 1.2257 and the R-Squared Value of 0.9959 (Figure 13), while the GB Tree Model obtained the RMSE Score of 1.3614 and the R-Squared Value of 0.9949 (Figure 14). In this way, the best model indicated to forecast would be the Decision Tree Model, as it has performed a little above the GB Tree, however, it was not possible to make the prediction with this model, since I could not use them, and I stayed for days trying to tweak these models or to use others model in Spark.

Padrão do plano de fundo

Descrição gerada automaticamente

Figure 13 - Metrics of Decision Tree Model

Uma imagem contendo Tabela

Descrição gerada automaticamente

Figure 14 - Metrics of GB Tree Model

Due to the short time to carry out the project, it was not possible to delve into the problem and find a solution, so the tweet\_hour data frame was migrated to pandas and a new EDA was possible to do in more detail. The existence of trend and seasonality in the data was verified, through the scale of the count of positive or negative tweets per hour. It was not possible to find an evident trend or seasonality, in either of the two feelings, as shown in the figures below.

Linha do tempo

Descrição gerada automaticamente

Figure 15 - Trend analysis and seasonality - Positive and Negative Sentiment

To make sure that the data had no trend or seasonality, the Augmented Dickey-Fuller (ADF) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests were conducted on both sentiments.

**Positive Sentiment**: in the ADF test, the series obtained a p-value of 0.00023 and a score of -4.46, thus rejecting the null hypothesis and being considered a stationary series. In the KPSS test, it obtained a score of 10.27 and a p-value of 0.01, failing to reject the null hypothesis and being considered a stationary series. Figure 16 demonstrates trend, seasonality, and residual data for positive sentiment.

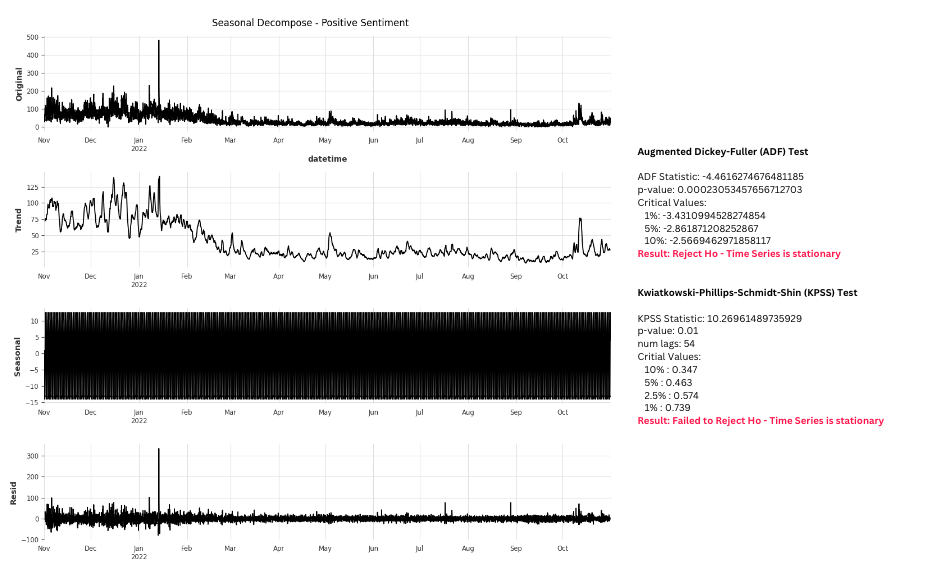


Figure 16 - Trend, Seasonality and Residual Analyses- Positive Sentiment

**Negative Sentiment**: Not unlike positive sentiment, negative sentiment was considered a stationary series in both tests, obtaining the following results: ADF score: -4.77, p-value: 6.24; KPSS score: 10.45, p-value: 0.01. Figure 17 demonstrates Trend, Seasonality and Residual graphs of negative sentiment.

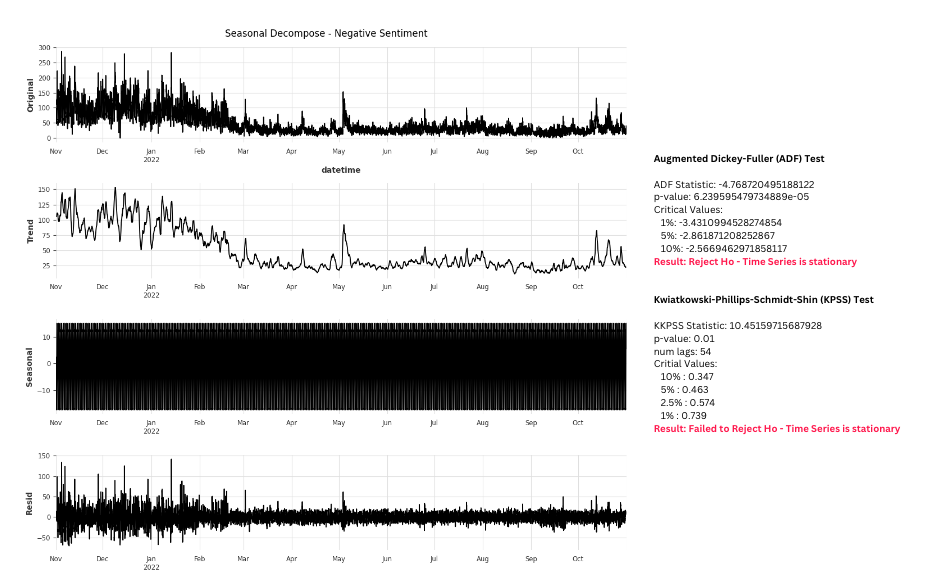


Figure 17 - Trend, Seasonality and Residual Analyses- Negative Sentiment

These results indicated that the data could be predicted by a simple linear model, so a few more models were tested, to obtain the model result for prediction. Using the Darts library, the following models were trained and tested with the database, then the full database was predicted, so a plot could be made showing how these model fitted.

### Naïve Seasonal

The Naïve Seasonal model, as seen in the figure below, shown an apparent good similarity between the predicted and the real data, however, these data presented indicators much lower than expected. That's why this template was dropped**.**

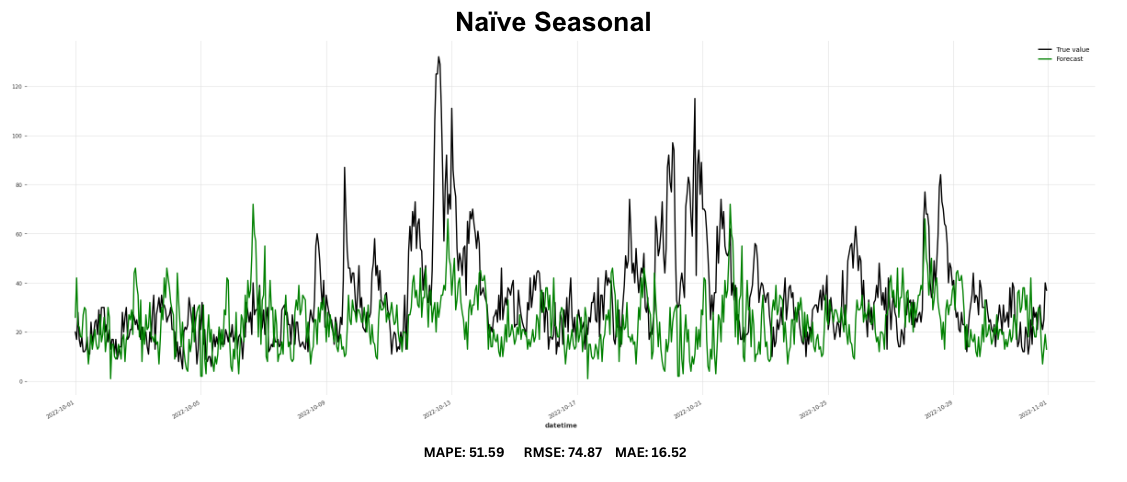


Figure 18 - Naive Seasonal Testing

### Prophet

As the Naïve Seasonal model, the Prophet, tried to follow the model, however, as seen in the chart and the following indicators, this model did not perform well, so it was also discarded to forecast the data.

Uma imagem contendo Gráfico de linhas

Descrição gerada automaticamente

Figure 19 - Prophet Testing

The two previous models obtain good performance with data that have trend and seasonality, characteristics that the data of this project does not have. Therefore, a more linear approach was used in the following graphs.

### AUTOarima

The AUTOarima model performed better than the Prophet, however, slightly worse than the Naïve Seasonal and underwhelming, in this way, it was also discarded.

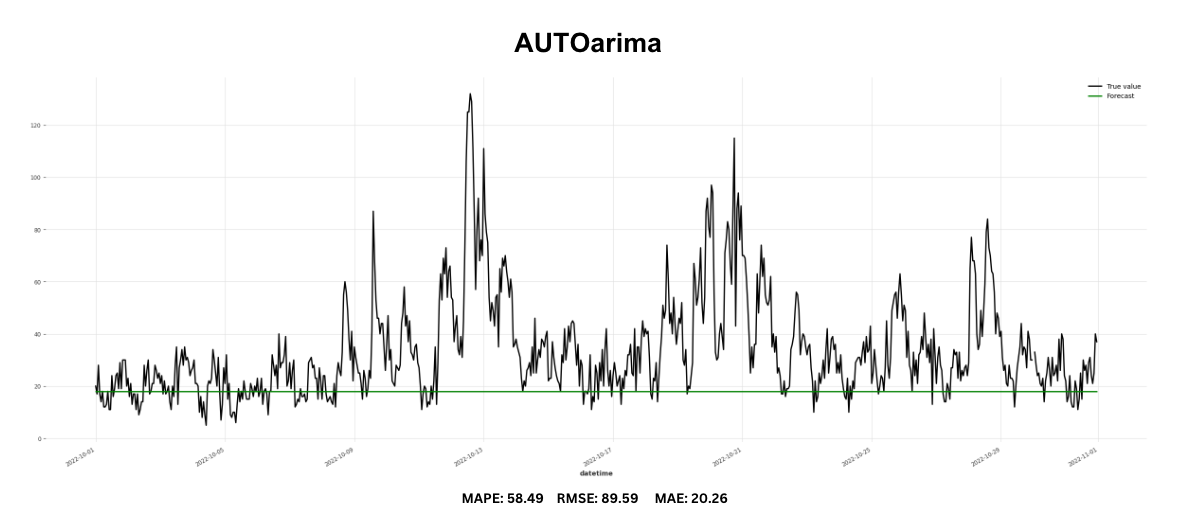


Figure 20 - AUTOarima testing

**Croston**

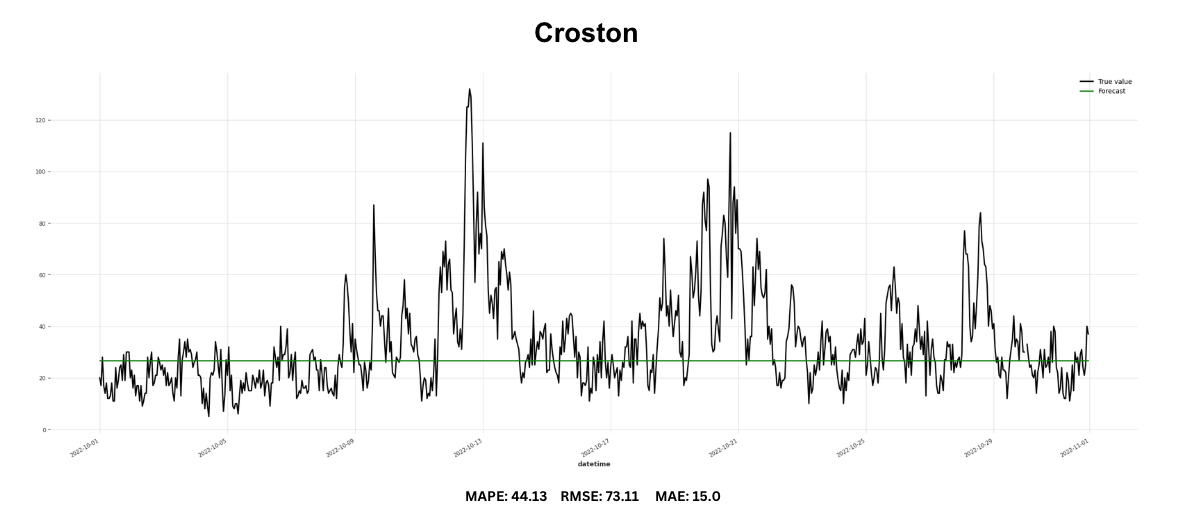
****

Figure 21 - Croston testing

As shown in the figures above, the model that performed best was Croston, which used a linear projection and was able to obtain an RMSE of 73.11 and a MAPE of 44.13. However, still far above what was achieved with Machine Learning models in Spark. Therefore, another test was conducted using the XGboost library to create a Machine Learning model in Pandas.

### XGboost – Regressor

For this model the test and training base was not split using the last month, it was split randomly. Thus, the model was trained with 80% of the base and tested with 20%. As shown in Figure 22 the model got an RMSE of 11.38 and an r-squared of 0.9519 this model was chosen to make the predictions.

****

Figure 22 - XGboost Indicators

## Forecasts for 7 days (1 week)

The forecast for the following 7 days maintained a stability in the data, with no visible increase or decrease in the number of tweets.

Gráfico, Histograma

Descrição gerada automaticamente

Figure 23 - Forecasts for 7 days (1 week)

## Forecasts for 30 days (1 month)

As expected for this database the forecast for 30 of the remained linear, with some peaks.

Gráfico, Histograma

Descrição gerada automaticamente

Figure 24 - Forecasts for 30 days (1 month)

## Forecasts for 90 days (3 months)

The 90-day forecast showed an increase in for the month of January 2023, probably the model understood that there is a seasonality in this period and thus projected an increase in the number of Tweets.

Gráfico, Histograma

Descrição gerada automaticamente

Figure 25 - Forecasts for 90 days (3 months)

Using the machine learning model, we see that there is an increase forecast for the first month of 2023. However, I believe that this is due to the short time used for training, since this period had an atypical situation. For there to be a greater level of confidence in the model's predictions, a longer period would be needed, with more examples of the months, in particular, those predicted.

**Dashboard**

The Dashboard created in this project was done in a simple way, since due to the complexity of the previous activities, as well as the existence of a very large second project, there was not enough time to work on more details and design (HTML and CSS).

The base was loaded from Hadoop using Spark, using JSON, emoji, word clouds among others analysis libraries, the following indicators were extracted and chosen for the dashboard. For organization this session is divided by rows of the Dashboard (). The Dashboard was created by Plotly's Dash library, which offers dynamic charts.

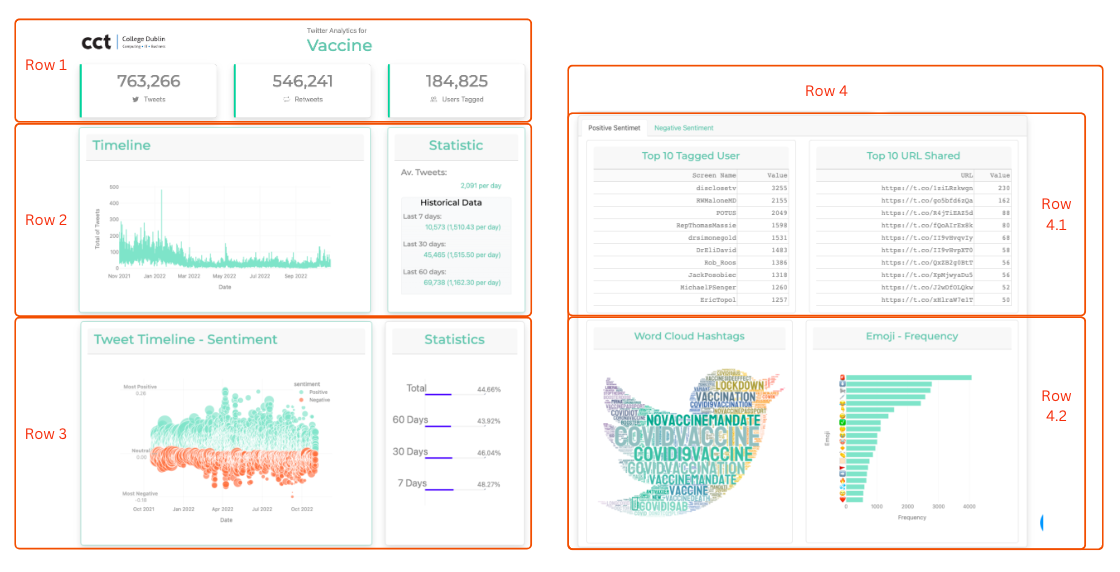


Figure 26 – Dashboard – Rows’ structure

## Row 1

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Figure 27 – Dashboard - Row 1

This session was used to put the name of the Dashboard and the visual identity (in this case the CCT logo), followed by 3 indicators:

* **Tweets** - count of all Tweets in the database.
* **Retweets** - count of Tweets with 'RT' character at the beginning, which proves to be a "retweet".
* **Users Tagged** - the count of people tagged in posts, for this indicator the person is counted only once regardless of how many times he was tagged.

## Row 2

Row two contains quantitative information, such as a Timeline plot displaying the volume of Tweets and the "Av Tweets" indicators with average amount of Tweets per day, and historical data in the "Historical Data" section where the amount and average of Tweets by "Last 7 Days", "Last 30 Days", and "Last 60 days" is shown. Considering the last day of the data (31/10/2022).

Interface gráfica do usuário

Descrição gerada automaticamente

Figure 28 - Dashboard - Row 2

## Row 3

Row three displays information about feelings, with a timeline displaying the intensity of feelings as well as their volume in time. Next to it displays the percentage of positive feelings in the "Total" and in the last 7, 30 and 60 days.

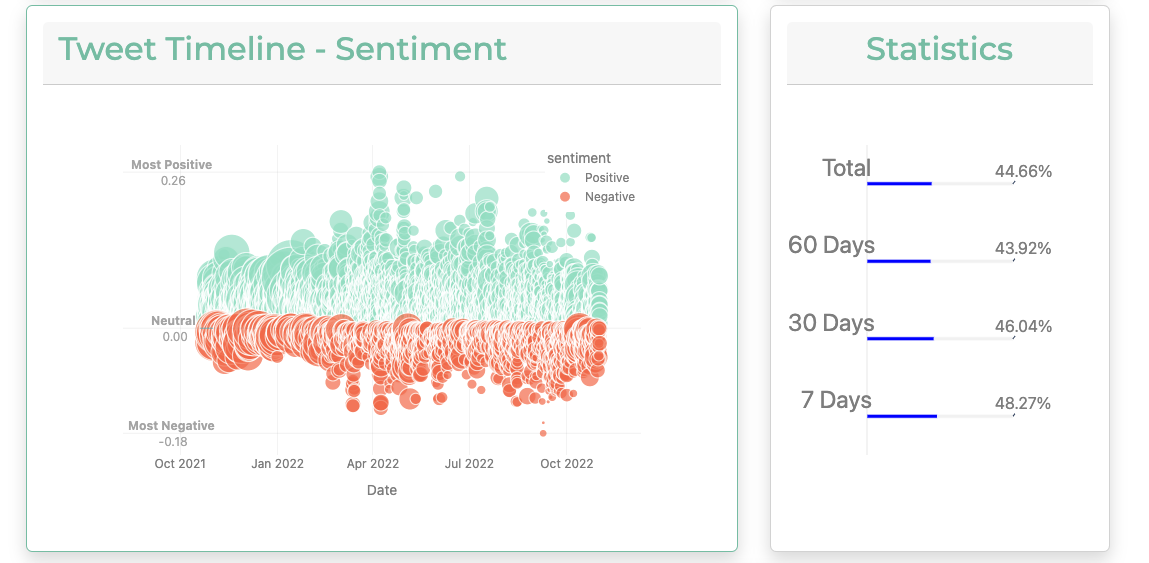


Figure 29 - Dashboard - Row 3

Row 4

Row four is segmented into two parts, and the two parts are filtered according to the sentiment chosen in the tab (Positive and Negative).

### Row 4 part 1

Part 1 of row 4 demonstrates the top 10 users tagged in tweets and the top 10 URLs shared, according to the selected sentiment.

Tabela

Descrição gerada automaticamente com confiança média

Figure 30 - Dashboard - Row 4.1 – Positive Sentiment

Tabela

Descrição gerada automaticamente

Figure 31 - Dashboard - Row 4.1 – Negative Sentiment

### Row 4.2

Row 4.2 displays a word cloud of the hashtags and a graph (top 20) with the top shared emojis, according to the selected sentiment.

Gráfico

Descrição gerada automaticamente

Figure 32 - Dashboard - Row 4.2 – Positive Sentiment

Gráfico

Descrição gerada automaticamente com confiança média

Figure 33 - Row 4.2 – Negative Sentiment

# Bibliography

1. *Adding Themes/Figure Templates* (no date). Available at: <https://hellodash.pythonanywhere.com/adding-themes/figure-templates> (Accessed: 26 May 2023).
2. Ardianumam - GitHub (2022) *ApacheSparkVideoSeries/dataset*. Available at: <https://github.com/ardianumam/compilations/blob/master/ApacheSparkVideoSeries/dataset/README.md> (Accessed: 26 May 2023).
3. *Dash-by-Plotly/minimal\_card.py at master · Coding-with-Adam/Dash-by-Plotly · GitHub* (no date). Available at: <https://github.com/Coding-with-Adam/Dash-by-Plotly/blob/master/Bootstrap/Card-Advanced/minimal_card.py> (Accessed: 29 May 2023).
4. Es-Sabery, F. *et al.* (2021) ‘A MapReduce Opinion Mining for COVID-19-Related Tweets Classification Using Enhanced ID3 Decision Tree Classifier’, *IEEE Access*, 9, pp. 58706–58739. Available at: <https://doi.org/10.1109/ACCESS.2021.3073215>.
5. Fuster-Casanovas, A. *et al.* (2022) ‘The #VaccinesWork Hashtag on Twitter in the Context of the COVID-19 Pandemic: Network Analysis’, *JMIR Public Health and Surveillance*, 8(10), p. e38153. Available at: <https://doi.org/10.2196/38153>.
6. *Hashtag Analytics in real-time for Twitter - TrackMyHashtag* (no date). Available at: <https://www.trackmyhashtag.com/preview/overview?_token=0kWLdSZIoPWyhm009oV3EYui1Dl4hUq5KgS8wJjt&req_url=https%3A%2F%2Fwww.trackmyhashtag.com&report_type=home&search_keyword=vaccine> (Accessed: 25 May 2023).
7. Hutto, C.J. and Gilbert, E.E. (2014) ‘VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text’, *).. Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, June.
8. *Independent multi-time series forecasting - Skforecast Docs* (no date). Available at: <https://skforecast.org/0.8.0/user_guides/independent-multi-time-series-forecasting.html> (Accessed: 12 May 2023).
9. *Internet Archive: About IA* (no date). Available at: <https://archive.org/about/> (Accessed: 29 April 2023).
10. *Internet Archive Terms of Use* (no date). Available at: <https://archive.org/about/terms.php> (Accessed: 29 April 2023).
11. Peixeiro, M. (2022) *The Complete Guide to Time Series Forecasting Using Sklearn, Pandas, and Numpy*, *Medium*. Available at: <https://towardsdatascience.com/the-complete-guide-to-time-series-forecasting-using-sklearn-pandas-and-numpy-7694c90e45c1> (Accessed: 14 May 2023).
12. Prabhakaran, S. (2019) ‘Augmented Dickey-Fuller (ADF) Test - Must Read Guide - ML+’, *Machine Learning Plus*, 2 November. Available at: <https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/> (Accessed: 14 May 2023).
13. ramírez, fede garza (2022) *Distributed Forecast of 1M Time Series in Under 15 Minutes with Spark, Nixtla, and Fugue*, *Medium*. Available at: <https://towardsdatascience.com/distributed-forecast-of-1m-time-series-in-under-15-minutes-with-spark-nixtla-and-fugue-e9892da6fd5c> (Accessed: 13 May 2023).
14. Sandha, S.S. (2023) *Deep-Learning for Time Series Forecasting: LSTM and CNN Neur*, *Medium*. Available at: <https://medium.com/@sandha.iitr/deep-learning-for-time-series-forecasting-lstm-and-cnn-neur-4c934cb16707> (Accessed: 20 May 2023).
15. Sanh, V. *et al.* (2020) ‘DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter’. arXiv. Available at: <http://arxiv.org/abs/1910.01108> (Accessed: 18 May 2023).
16. *Sell Out Sell In Forecasting. Machine Learning for sales forecasting… | by Bartosz Szabłowski | Towards Data Science* (no date). Available at: <https://towardsdatascience.com/sell-out-sell-in-forecasting-45637005d6ee> (Accessed: 18 May 2023).
17. Sharma, R. (2021) ‘Forecast Time-Series With XGBoost’, *MLearning.ai*, 22 July. Available at: <https://medium.com/mlearning-ai/forecast-with-xgboost-a1abf870c1cc> (Accessed: 24 May 2023).
18. *Skforecast: time series forecasting with python and scikit learn* (no date). Available at: <https://www.cienciadedatos.net/documentos/py27-time-series-forecasting-python-scikitlearn.html> (Accessed: 24 May 2023).
19. Subbaraj, V. (2022) ‘User Defined function in PySpark’, *Medium*, 8 May. Available at: <https://medium.com/@vaishalisubbaraj/user-defined-function-in-pyspark-e9740dc2d3bd> (Accessed: 23 May 2023).
20. Szabłowski, B. (2023) *Forecast Multiple Time Series Like a Master*, *Medium*. Available at: <https://towardsdatascience.com/forecast-multiple-time-series-like-a-master-1579a2b6f18d> (Accessed: 23 May 2023).
21. *Time series / date functionality — pandas 2.0.1 documentation* (no date). Available at: <https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#offset-aliases> (Accessed: 23 May 2023).
22. *Timeseries — darts documentation* (no date). Available at: <https://unit8co.github.io/darts/generated_api/darts.timeseries.html> (Accessed: 23 May 2023).
23. *#TNW2019 Hashtag Analytics | RiteTag: Find the best hashtags* (no date). Available at: <https://ritetag.com/hashtag-stats/TNW2019?uuid=4073be07-a7ac-4559-9fde-a31e92e9e211> (Accessed: 25 May 2023).
24. *Tutorial: Quickstart — TextBlob 0.16.0 documentation* (no date). Available at: <https://textblob.readthedocs.io/en/dev/quickstart.html#sentiment-analysis> (Accessed: 26 May 2023).
25. *Twitter - Rules and policies* (no date). Available at: <https://help.twitter.com/en/rules-and-policies> (Accessed: 26 May 2023).
26. Umam, A. (2023) ‘compilations’. Available at: <https://github.com/ardianumam/compilations> (Accessed: 23 May 2023).
27. *vaccine hashtag analysis - BrandMentions* (no date). Available at: <https://app.brandmentions.com/h/p/602795/vaccine> (Accessed: 25 May 2023).
28. Verma, Y. (2021) *Complete Guide To Dickey-Fuller Test In Time-Series Analysis*, *Analytics India Magazine*. Available at: <https://analyticsindiamag.com/complete-guide-to-dickey-fuller-test-in-time-series-analysis/> (Accessed: 24 May 2023).
29. Yousefinaghani, S. *et al.* (2021) ‘An analysis of COVID-19 vaccine sentiments and opinions on Twitter’, *International Journal of Infectious Diseases*, 108, pp. 256–262. Available at: <https://doi.org/10.1016/j.ijid.2021.05.059>.