

Github link: <https://github.com/2020491-v2/MSC_DA_IntegratedCA>

# I. Data storage and processing activities [20%]

The basic data storage and management requirements are the same for datasets of all sizes, however what sets Big Data apart from conventional data is its significantly larger size and the challenges that need to be considered when setting up a solution, whose purpose is to derive useful insight from such a large volume of data.

The data storage solution must be able to collect and retain all this data. The data management solution, on the other hand, must ensure the data ingestion and processing speeds are optimal and the data is available to all necessary users with the correct privileges.

Doug Laney (2001) defined Big Data characteristics using the “3 Vs”, which can be used to create a strategy for data storage and management:

· **Volume** is the size of the data and as a result defines how large the data storage solution must be in order to store and process the continuously growing dataset or datasets. Big Data datasets are vastly larger in size than traditional datasets and in the majority of the cases the processing requirements exceed that of a single device which means resource pooling, allocating and coordinating, usually in the form of breaking tasks into smaller manageable tasks with the help of a groups of devices in the form of Cloud Solutions.

· **Velocity** is the speed at which information passes through the system from source or sources to target. Occasionally the data may be being prepared before it reaches the target, e.g. filtered, altered or corrected. With Big Data, new data is constantly added which needs to be analysed for real-time insight. This requires robust data pipelines with the highest availability.

· **Variety** describes the data structure and type that will be stored and analysed. Around 90% of Bid Data is unstructured (2023, Agile Solutions) and tends to be very varied. The sources can be from relational databases like business transactions, social media feeds, logs, APIs, etc. The data itself can be of varying formats too, e.g. text, csv, images, video, audio, logs, etc. Legacy approaches are compatible with structured data which is also usually labelled and only partially compatible with Big Data which can be structured or unstructured. Therefore, any Big Data solutions must be able to bring all these varied sources and data types together into a system and process and analyse them effectively despite the variety.

This is usually accomplished using data management platforms such as databases, data lakes or data warehouses. A data warehouse stores relational data collected from transactions and since it’s structured, it can be queried and analysed faster. A data lake is a data repository that stores structured and unstructured data at any scale which makes collection and storage quicker however analysis will need to come later as the data is structured and the business questions that need answering developed later.

Ellingwood, J. (2016) offers 4 steps when designing Big Data systems as follows:

1. **Data Ingestion**, more commonly known as ETL (Extract Transform Load) looks after collecting the raw data from the sources and transferring them to the target. This ingestion will be handled by data pipelines which filters, labels, sorts, modifies and can even analyse the data during this process. The type of data load is also important for computational resources. Smaller datasets can afford to do a Full load which grabs the entire source dataset or datasets and moves them to the target, replacing the entire table. With large datasets usually incremental loads are encouraged as it will only transfer new rows or updated rows over into the target, which means less data will need to be transferred and processed.

2. **Data Storage Persisting** is the next step where the data is stored in a robust and available way. This means the storage resource has to be large enough to keep all that large volume of data within a data cluster and accessible to the users and processes that need them.

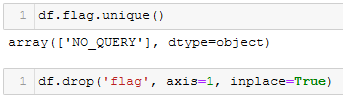
3. **Computing and Analyzing data** is the next step to gain insight from the large volume of data real time. The analysis process begins to analyse the data. The data may be analysed continuously in a loop to reach that insight or analysed in a batch (broken down into a smaller tasks) either scheduled to run at different times or have smaller datasets run in parallel across multiple computing resources to handle such a large dataset effectively. This is also referred to as splitting, mapping, shuffling, reducing, and assembling. Another type of data analysis is real-time processing where data is processed and its result returned as soon as possible even as new data is being added to the source or sources. This is accomplished by using in-memory computing, which creates a representation of the data in the data cluster’s memory in order to skip the step of having to write to the disk.

4. **Visualizing the results** is the final step that is the goal of the Data Analysis process, finding insight and recognizing trends.

**Processing Steps of Data:**

*Irrelevant Data Removal:*

* *The flag variable only has NO\_QUERY value hence it does not serve any data and hence removed.*

**

* *ids column is unnecessary for the analysis, since the best way to sort this data is with the date of the tweet and then indexing it accordingly without the need to store and handle large integer values (i.e. Big integers).*

*Not to mention the different issues that may arise from big integers such as:*

* ***Memory Usage*** *which limits the capacity of the analysis*
* ***Performing arithmetic*** *operations slow down with increased bit size for calculations*
* ***Computation*** *Libraries/Functions might not be optimized for big integers, leading to slower computations or potential errors*
* ***Overflow*** *operations on big integers may cause overflow errors if arbitrary precision arithmetic not handled*
* ***Visualization*** *Big integers can cause issues with data visualization libraries, e.g. when generating charts & graphs. Also, Scaling Big integer axes might also lead to distorted visualizations.*

*For these reasons the ids column can be removed.*

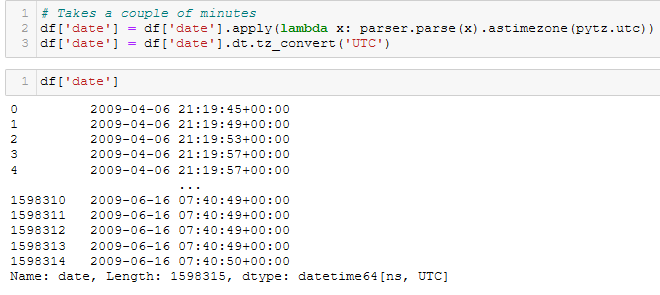
**

*Text Formatting:*

Text needs to be converted from object to string format to be analysed.

*Date Formatting & Sorting:*

To be able to sort the data based on date, the date needs to be converted to datetime format. For this the date string format needs to be processed to UTC to avoid timezone related issues (possibly in e.g. plotting).



Reindexing the data by sorting based on date from oldest to most recent tweet.



### **Possible Scam & Irrelevant Data**

Any tweet has the probability to contain malicious link to check its validity would take extensive analysis not to mention how it is an irrelevant data for the analysis to be conducted. For the these reasons it is removed for further analysis.



### **HTML Character Entities**

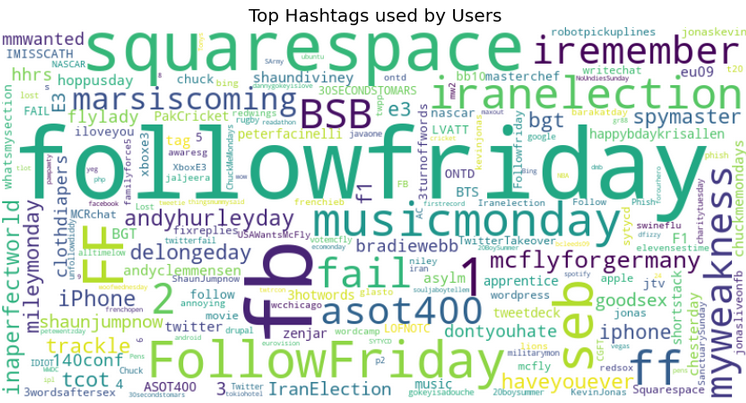
Having Entity names could affect the analysis for a number of reasons including:

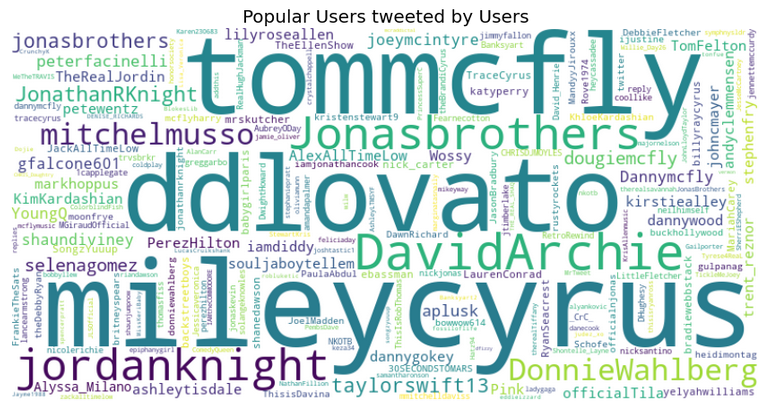
* *Interpretation error* e.g. sentiment analysis may have the meaning of the text distorted
* *Tokenization Disruption* with possible incorrect word counts
* *ML* introducing additional features not contributing to text meaning could impact the *model*'s performance & generalization
* *Data Cleaning* HTML entities can be considered noise during text analysis
* ...

To avoid it these are removed from the text.

### **Hashtags, Popular Users, Special Characters, Digits & Extra Spaces Removal**

To find the sentiment of the users hashtags and user names do not contribute to further analysis. So they are removed.





### 

### **User Column Removal**

### For the overall sentiment analysis it is irrelevant which user tweeted, the overall sentiment of the data is crucial.

## **Expanding Contractions**

### For the efficiency of the next step, i.e. removing stopwords, it is crucial to expand the contractions to the groups of words that they represent. e.g. didn't to did not...

### 

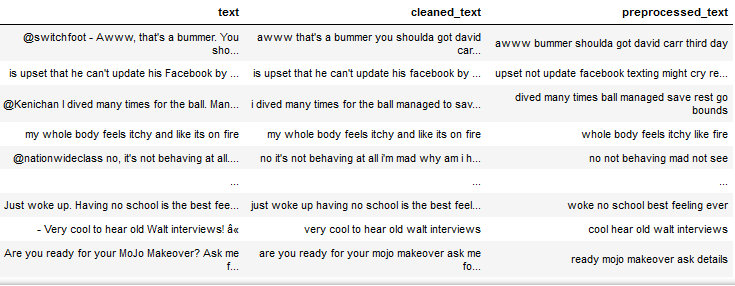
## **Removing Stopwords**

Stopwords (e.g. the, and, is, of, in etc.) being commonly used words that generally don't carry meaning. I will remove them for a number of reasons:

* reduce noise (in text analysis)
* improve efficiency reducing word size to process speeding up algorithms and save memory
* better interpretation for topic modelling more meaningful and interpretable topics

It is important to consider how sometimes stopwords might be crucial for instance: in sentiment analysis some stopwords like **not** can change the meaning. In such cases, retaining certain stopwords.

Hence, I will retain the stopwords 'no' and 'not' to present the meaning.



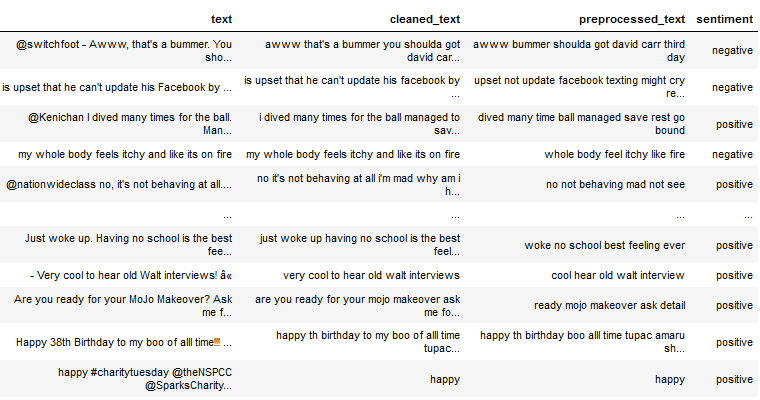
## **Text Tokenization & Stemming/Lemmatization**

Using Porter stemming algorithm, to Stem the text has resulted in incorrect stemming such as his became hi, body became bodi. For this reason I decided to use lemmatization to increase the chances to produce a valid base form (i.e. lemma) of the words.

# Sentiment Labeling

Assigning sentiment labels to the text: positive, negative or neutral based on the sentiment expressed in the text.

I will use the **VADER** sentiment analysis tool from the NLTK library, which is a pre-trained sentiment analysis models trained on large text datasets.



#### **Check Empty texts after preprocessing**

Making sure no empty rows are left in the data after processing of the data.

# **II.** R**ationale and justification of choices [40%]**

Big Data Environments need to be able to manage unstructured and semi structured data.

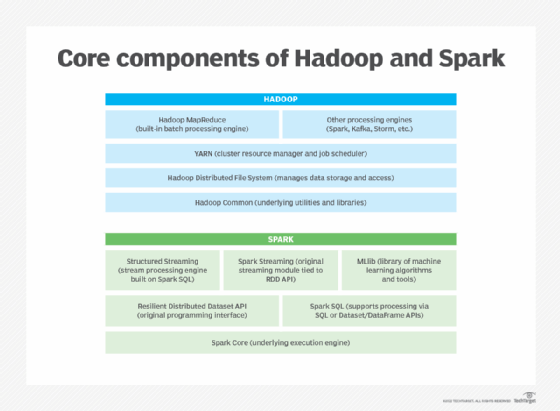
## **Hadoop vs Spark**

For the purposes of this project, it is worth comparing the most popular data processing frameworks for big data, which remains Hadoop and Spark (Lawton, 2022). A massive advantage of working with either is the vast ecosystem of open-source technologies that are readily available when it comes to the processing, managing and analysing of the big data.

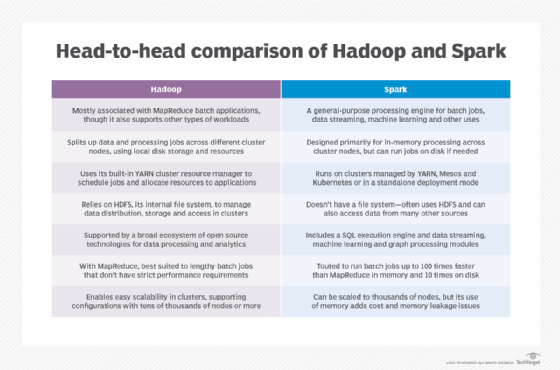
Spark can process batch jobs quicker than Hadoop. However, they can be used interchangeably as even Spark uses the Hadoop Distributed File System (HDFS) as one of its main data storage options since it lacks its own file system/repository.

Hadoop can process large amounts of data, efficiently breaking up large data processing problems across different computers, running computations locally and then combining the results (Lawton, 2022). Hadoop’s distributed processing architecture makes it easy to build big data applications for clusters containing a vast number of nodes.

Spark can scale in-memory processing across distributed cluster nodes more efficiently (Lawton, 2022). Spark is also known to process vast amounts of data by splitting up workloads on different nodes much faster than Hadoop. As a result, Spark can handle use cases, making it more general-purpose.



Although Hadoop has advantages when it comes to keeping costs down for large processing jobs, Spark has an advantage in delivering timely analytics insights due to its design to process data mostly in memory. While Hadoop is primarily suited for the analysis of historical data, Spark can optimise high-throughput data processing jobs, making it more suitable for various uses. As such, Spark is very much applicable in online applications and interactive data analysis. As one of its main disadvantages, Spark must be paired with Hadoop or other platforms for long-term data storage and management.



To sum up, while Hadoop is more cost-effective with better long-term data management capabilities, Spark is more adept at supporting analytics applications that run in interactive modes where multiple operations need to be performed simultaneously.

Utilisation of a distributed data processing environment (e.g., Hadoop Map-reduce or Spark), for some part of the analysis.

## **Databases: SQL vs NoSQL**

The main differences are that SQL databases are relational to store structured datasets in the form of tables with rows, while NoSQL are non-relational for unstructured or semi-structured datasets like documents or JSON files, which are document, key-value, graph or wide column based. SQL database uses SQL and predefined schemas whereas NoSQL databases use dynamic schemas. This makes SQL databases vertically scalable and NoSQL horizontally scalable. This makes NoSQL databases more adaptable to adapt to new data source types and sizes, however SQL has a vastly more efficient way to query the data as it is in an easily digestible format for query languages.

Both SQL and NoSQL (MongoDB) were used to store the Tweets that were used in the sentiment analysis as a form of benchmark to compare the effectiveness of both.

## **Programming language**

Python was used instead of R for this project. Both languages are open source with a supportive community behind them. Although R is a programming language specialised for data analysis, Python is more versatile with more libraries specifically for text processing and sentiment analysis. It is also well structured. Whereas R consumes physical memory as it needs to load bigger datasets which results in a slower process and as we are working with larger datasets, Python made more sense. R generally is used for statistical and data visualisation tasks and Python for more advanced data analysis tasks.

## **Machine Learning models**

SVM, Random Forest and Naïve Bayes were used.

**Models Selection**

In sentiment analysis, the main goal is to classify text into different sentiment categories (i.e. positive, negative, neutral). The models selection is based on their ability to:

- handle text data

- capture patterns

- generalise well to new, unseen data

1. Naive Bayes simple & fast algorithm that works well with text data. It's based on Bayes' theorem and assumes that the features (i.e. words) are conditionally independent given the class label. Despite its "naive" assumption, it often performs well for text classification tasks.

2. Support Vector Machine (SVM) effective for text classification when combined with appropriate feature representations like TF-IDF. They aim to find a hyperplane that maximally separates different classes. SVMs can handle high-dimensional data well and are known for their ability to find complex decision boundaries.

3. Random Forest an ensemble method combining multiple decision trees to make predictions. It's robust, handles non-linear relationships, and can capture interactions between features. Random Forest can work well for text classification when combined with features like TF-IDF.

**Feature Extraction**

*Term Frequency-Inverse Document Frequency (TF-IDF)* a common feature representation choice for sentiment analysis that assigns weights to words based on their frequency in a document relative to their frequency in the entire corpus. It helps capture the importance of words in a document relative to their significance in the entire dataset.

**Justification**

*Interpretability*

- Naive Baye and SVM are interpretable models as they give insights into which features contribute to the prediction, making it easier to understand the decision-making process.

*Efficiency*

- Naive Bayes is computationally efficient and can handle large datasets well.

- SVM` is efficient for high-dimensional data allows it to work effectively in non-linear scenarios.

*Ensemble Learning*

- Random Forest is an ensemble method that reduces overfitting and increases accuracy. It combines the outputs of multiple decision trees to improve generalization.

# **III.** C**omparative analysis of databases [10%]**

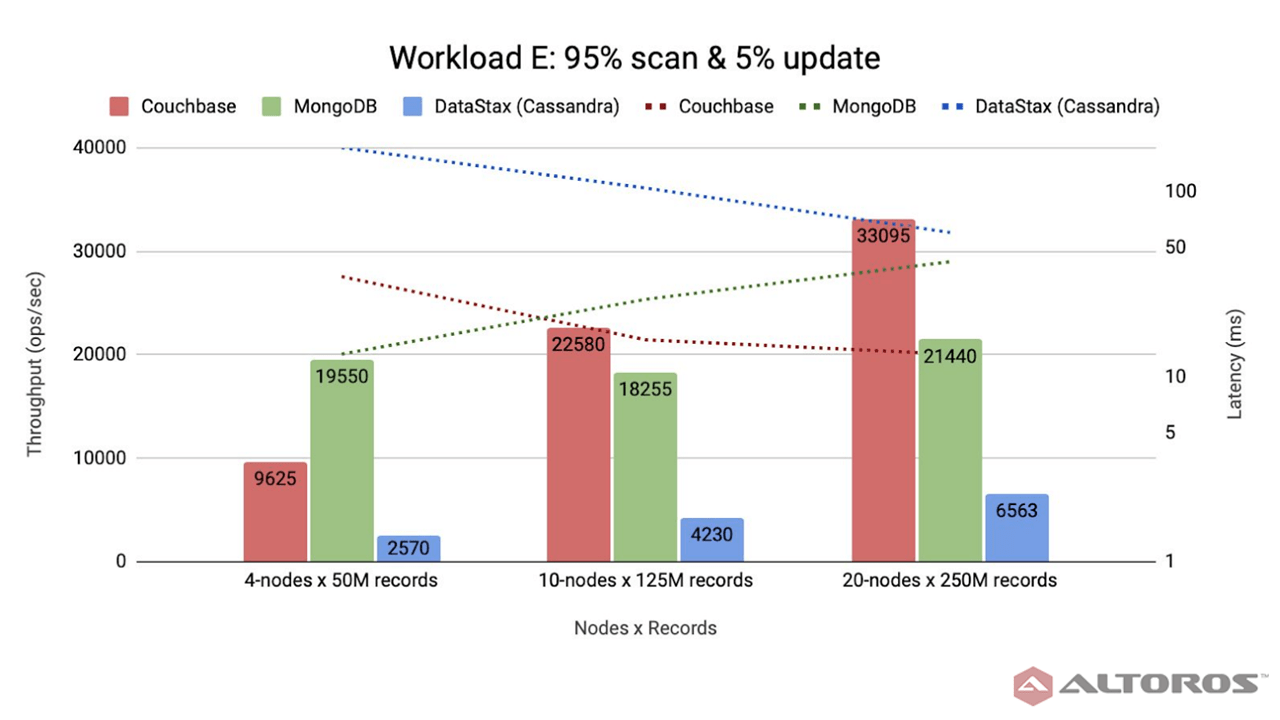
## **Database capabilities**

**Apache Cassandra** is an open-source wide-column based NoSQL database with high availability, performance and scalability. It is also a Cloud solution. It uses its own Cassandra Query Language (CQL) which is very similar to SQL. It supports 13 programming languages incl. Python. It does not support server-side scripts. It also offers high availability and no single point of failure (this is usually accomplished with the help of redundancies).

**Couchbase Server** is a document- oriented and key-value distributed NoSQL database. It uses a Declarative Query language which extends ANSI SQL to JSON. It supports only 9 programming languages incl. Python. It supports server-side scripts in JavaScript. It is generally recommended to use where high availability is more important than data consistency (redundant data is identical) as it scales computing resources well depending on the type of workload.

**MongoDB** generally has more functions than the above 2 databases. It is an open-source document-oriented NoSQL database that is also compatible with AWS, Microsoft Azure and Google Cloud, offering a highly available built-in automation for workload optimisation. It uses read-only SQL queries in its UI. It supports more than double the amount of programming languages than Cassandra, 30, incl. both Python and R. It supports server-side scripts in JavaScript.

MongoDB has a higher data consistency than availability and operational tests (Altoros, 2021) showed that for smaller nodes (around 4-nodes per 50 million records) it outperformed the other 2 however as the datasets got larger, Couchbase started to significantly outperform the other 2 while Cassandra remained consistent. In a project, which is all about the analysis itself and the data or results do not need to be shared with other users, data consistency would be key and the dataset itself will not be this large either therefore the fastest performance is also ideal.



Database comparison based on dataset size (Altoros, 2021)

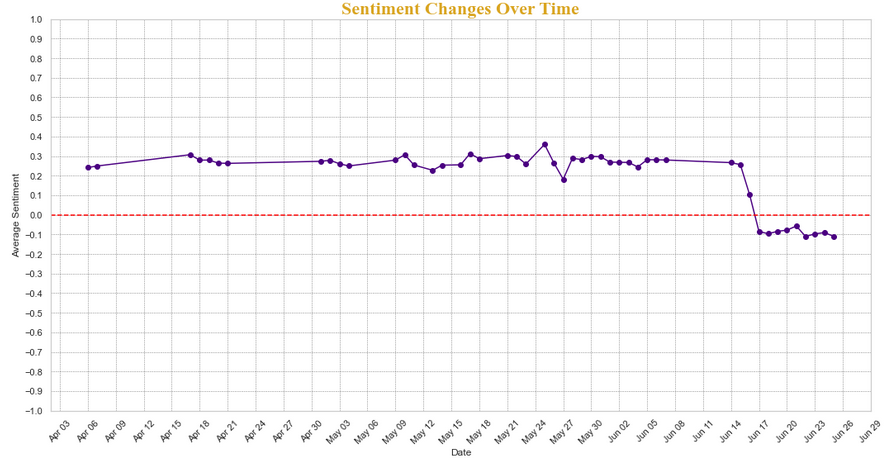
# **IV.** **Analysis of change sentiment [10%]**

Sentiment analysis, specifically change sentiment in the case of this project is a subsection of Machine Learning referred to as Natural Language Processing (NLP), which analyses human language. NLP can be applied in a number of ways. Machine Translation translates a language into another taking note of the grammar, structure and synonyms. Text Categorization groups data into separate groups, e.g. directing customer request tickets to the correct department depending on its description. Information Extraction retrieves useful information from text, e.g. work experience information from a submitted resume. Spam Filtering analyses the content of emails and mark them as spam if they contain special expressions that are usually present in spam messages. Currently, Machine Learning is already used to analyse spoken language too, however its accuracy in detecting the words spoken to it depends on the datasets they were trained with. As a result, English detection (one of the most common datasets used for training) tends to be highly accurate.

Machine Learning improves the more data it has access to and the more it is trained. 5 NLP datasets were mentioned by Khurana, D. et al (2022), all with datasets prime for sentiment analysis. Stanford Sentiment Treebank and IMDB have a list of movie reviews. Sentiment140 contains tweets, Paper Reviews has reviews in English and Spanish for IT conferences. Sentiraama is the most varied dataset with reviews for books, products, movies and song lyrics.

# Sentiment over a time period

To analyse changes in sentiment over a specific time period, the column date and sentiment data are used.



As seen above the overall Sentiment of the tweets are positive up to around **June 17th** from which point they remain negative.

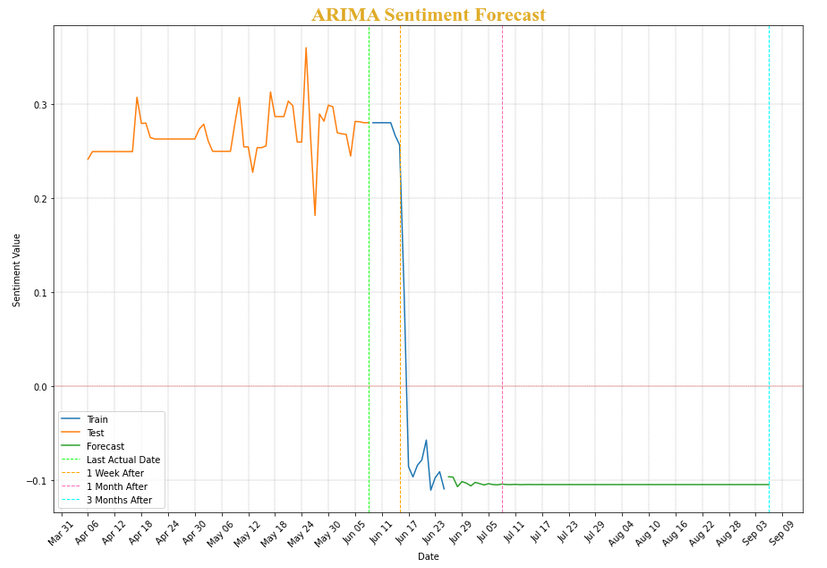
# **V.** F**orecast of sentiments [10%]**

Facebook’s Prophet and ARIMA were used for time series forecasting for 1 week, 1 month and 3 months.

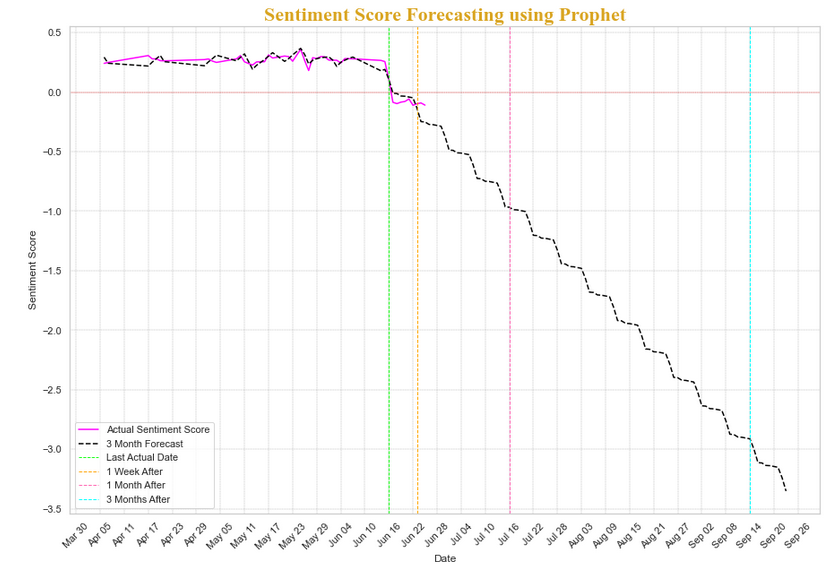
Due to the size of the dataset they have demonstrated to require extensive computational time.

I choose them for the below reasons:

**ARIMA** is a common time series forecasting method that captures autocorrelation and seasonality patterns in a time series data. It is efficient when the data exhibits clear trends and seasonality patterns that can be captured by the model. It's particularly effective when the data doesn't have complex non-linear relationships or external factors affecting it.



**Prophet** is a forecasting method designed to handle time series data with various components including trends, seasonality, and holidays. It's a more user-friendly and automated approach compared to traditional methods like ARIMA. Prophet is efficient with data that has strong seasonal components and multiple contributing factors.



In the context of forecasting 1 week, 1 month and 3 month data, both ARIMA and Prophet are effective. With the data having clear linear relationships, ARIMA works well. Although if the data shows any complex seasonality, trends, and possible holiday effects, Prophet is a better choice.

In both forecasting models a negative downward sentiment trend is found.

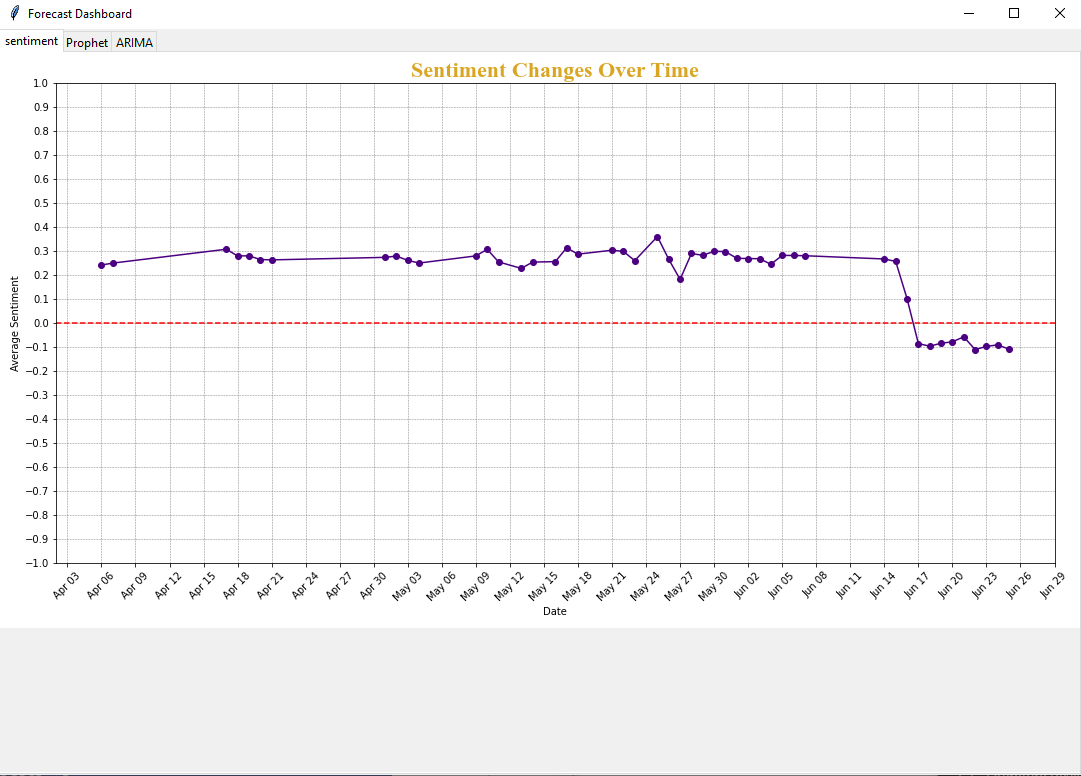
# **VI.** P**resentation of results [10%**]

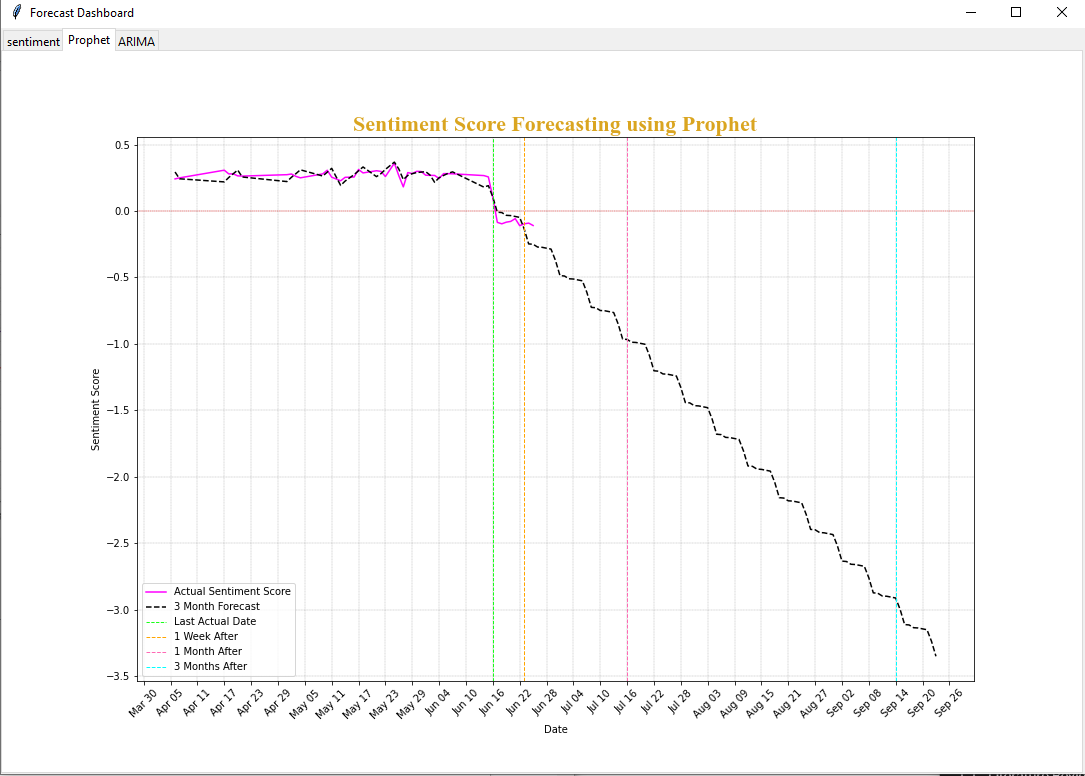
I used tkinter as a dashboard for the forecast results demonstration due to the number of pros it offers such as:

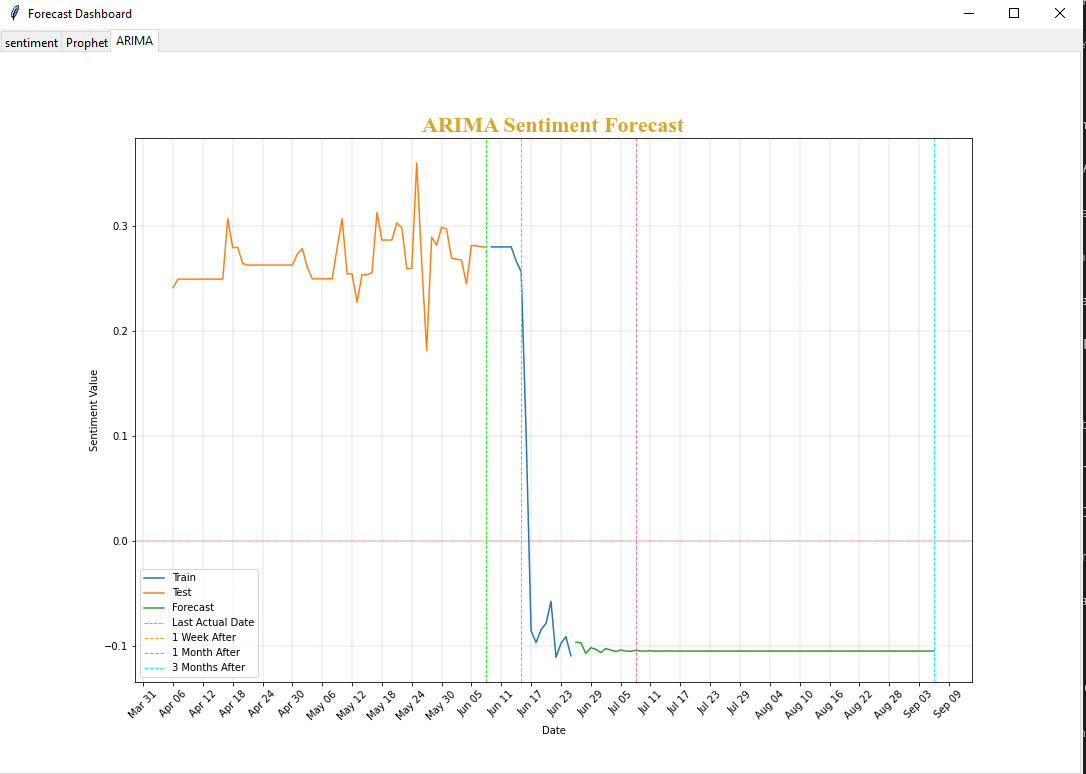
1. **Lightweight & Built-In:** As a standard GUI library it comes bundled with Python making it convenient for creating small applications without external dependencies
2. **Platform Independence:** Applications can run on various platforms e.g. Windows, macOS etc. without modifications
3. **Direct Integration with Matplotlib:** It integrates well with Matplotlib, allowing embed plots directly into the GUI, useful for displaying visualisations alongside other elements
4. **Local Deployment:** Application is deployed locally, beneficial for demonstrating models without requiring internet access or a web server
5. **Fast Prototyping:** For quick prototyping and demonstrating proof of concept, it gives a fast way to put together a GUI without investing extensive time in development

In conclusion, tkinter is an efficient choice for creating dashboards for model plot demonstrations due to its simplicity, local deployment, and quick prototyping.

Below is the dashboard with all its tabs demonstrated:







# **VII.** **References**

Alton, L. (2015). *What The 3 Vs of Big Data Mean for E-Commerce Businesses – London Datastore*. Available at: <https://data.london.gov.uk/blog/what-the-3-vs-of-big-data-mean-for-e-commerce-businesses/>. (Accessed on: 28 July 2023).

DB-ENGINES (2023). *Cassandra vs. Couchbase vs. MongoDB Comparison*. Available at: <https://db-engines.com/en/system/Cassandra%3bCouchbase%3bMongoDB> (Accessed on: 2 August 2023).

Ellingwood, J. (2016). *An Introduction to Big Data Concepts and Terminology*. Digitalocean.com. Available at:<https://www.digitalocean.com/community/tutorials/an-introduction-to-big-data-concepts-and-terminology>. (Accessed on: 1 August 2023).

InterviewBit. (2021). *Python Vs R: Know The Difference*. Available at: <https://www.interviewbit.com/blog/python-vs-r/>. (Accessed on: 1 August 2023).

Klein, E. (2023). *Cassandra vs. MongoDB vs. Hbase: A Comparison of NoSQL Databases*. [online] Available at: <https://logz.io/blog/nosql-database-comparison/>. (Accessed on: 28 July 2023).

Yudovin, A., Gutierrez, C. (2021). *NoSQL Comparison 2021: Couchbase Server, MongoDB, and Cassandra (DataStax)*. Available at: <https://www.altoros.com/blog/nosql-comparison-2021-couchbase-server-mongodb-and-cassandra-datastax/>. (Accessed on: 1 August 2023).