**Introduction**

Twitter, as of 2023, has approx 530 million active monthly users, whose sentiments, when related to a particular topic, business or product, can be used to estimate the public perception of a brand, monitor the response to a new product or service launch, monitor a competitor’s activity, and so on (Hutt, 2023). Sentiment analysis – applied to a Twitter feed, or any other body or bodies of text – is the mining of text data to substract subjective information from the data (i.e. the sntiment underlying it). Sentiment analysis is the most commonly applied and utilised form of text classification (Gupta, 2018).

A time series dataset of 6 million tweets were to be analysed for sentiment, and this sentiment forecasted for 3 time periods, 1 week, 1 month, and 3 months.

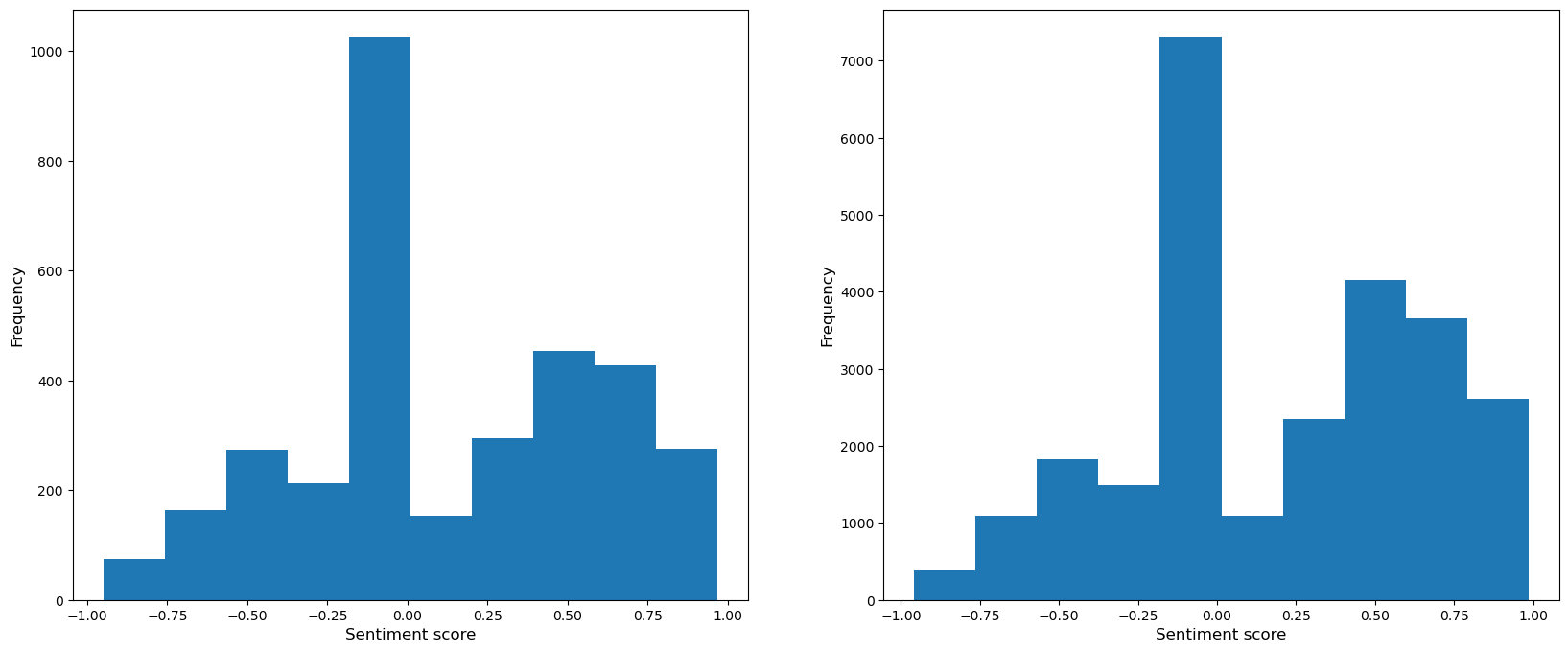
**Method**

The first step was to do some initial EDA to start to understand the content and shape of the data. With 6 millions tweets, and 6 variables, there was no missing data, though there were also no queries associated with any of the tweets. Also, many of the dates at which the tweets were recorded were not unique, as the field only recorded to the nearest second, while milliseconds or even smaller denominations may have be required to distinguish tweet submission time. Finally, the username field was not thoroughly examined, because it was not going to be used in this analysis of tweet sentiment over time, but it was found that there were a number of users who regular tweeted. It was also found that the tweet ID was, unexpectedly, not unique. This was because some of the tweet entries were duplicates. The mapper was used to remove these duplicated values by selecting only unique tweet IDs, leaving 1,598,315 unique tweets. This course of action was taken because it was assumed that the tweet duplication was a mistake. If these tweets had been posted twice deliberately - maybe for emphasis - they arguably should have been retained, but in that case the tweet ID would have been unique, and they were not.

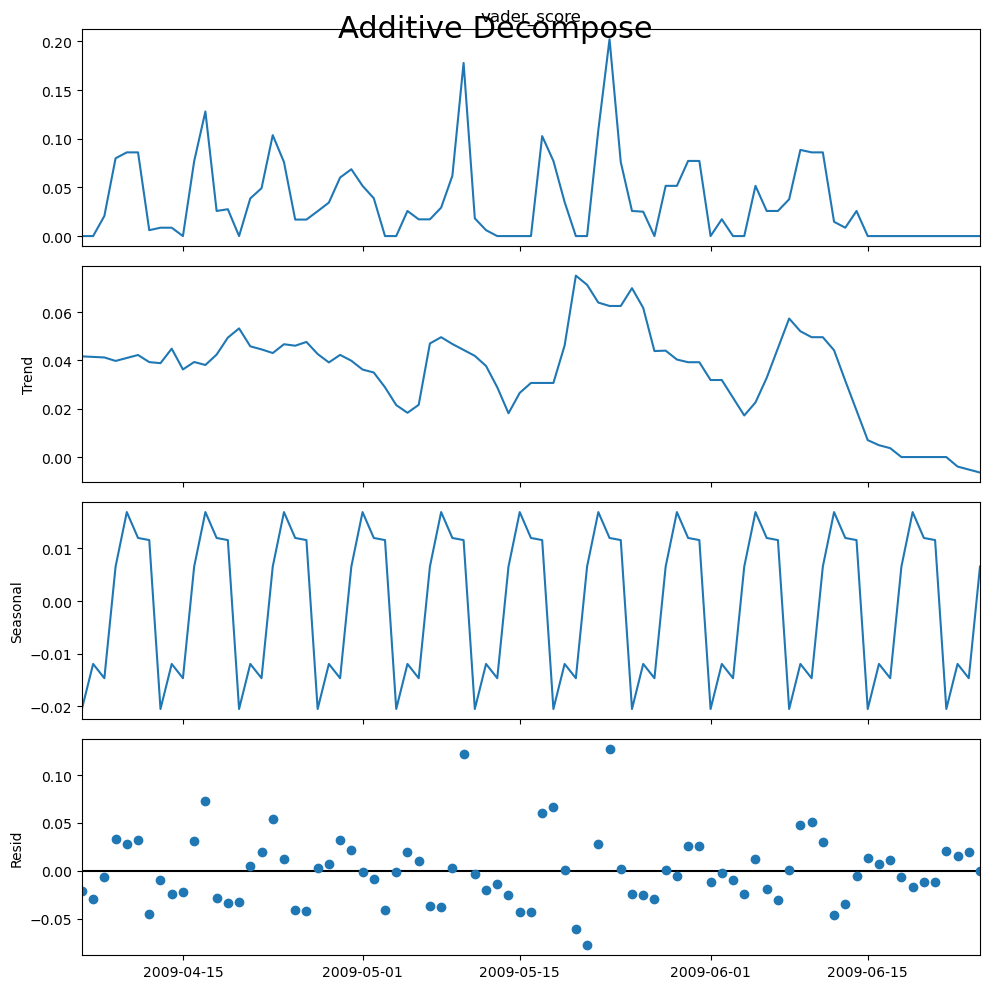
The MapReduce program used took the raw data (saved to the Hadoop HDFS as shown in the Appendix) and returned only the date and punctuationless tweet. Screenshots of the MapReduce program running are inclusing in the Appendix. A complication was that the tweets themselves often contained commas, and the mapper – programmed to split the data on a comma - was truncating those tweets with commas. To resolve this, a for loop in the mapper acted on any line longer than expected (i.e. any line that had been split into more parts due to the presence of a comma in the tweet) and concatenated the tweet segments together to form a single output tweet. The mapper returned the tweet number, the date, and the un-edited (other than removed commas) tweet. The reducer took this output and stripped all punctuation from it (i.e. removed all non-alpha characters), then returned the tweet number, the date, and the tweet, and output them to a comma separated file. The reason that the tweet letters were not all changed to lowercase was because the VADER model used to determine the sentiment distinguishes between capital and lowercase (Beri, 2020), and the case can impact the sentiment. VADER also assigns meaning to punctuation, so it may be advisable to leave the punctuation, or only remove aspects of it, such as ellipses or tildes, and leave more potentially meaningful ones such as exclamation marks, or question marks. However, punctuation was removed here to demonstrate a reducer programme. Stopwords, however, were not, to avoid removing context, especially as VADER can understand that adverbs, such as ‘not’, can change the sentiment of a word (e.g. ‘happy’, vs ‘not happy’) (Beri, 2020). The reducer output was used for sentiment analysis (after renaming to add an extension, .txt, to allow it to be read by pandas in Python).

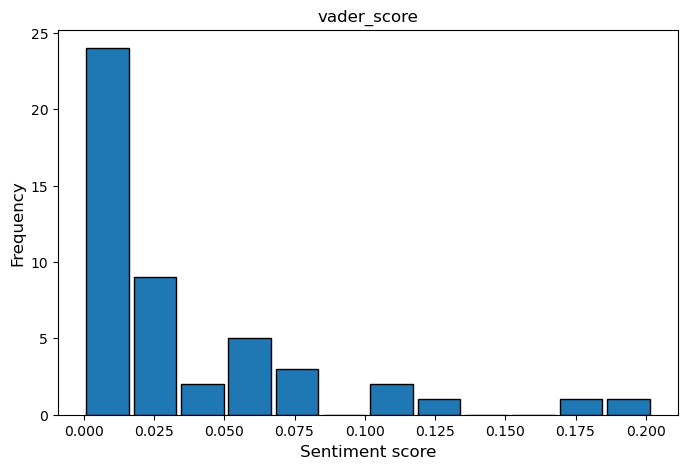
The VADER model scores the sentiment of each word in a piece of text, and sums them to determine the sentiment f the text as a whole (Beri, 2020). Capitalisation and punctuation are considered by the model, for example, the word ‘Happy’ gives a score of 0.5719, while ‘Happy!’ gives 0.6114. It is part of the NLTK package, one of the most commonly used Python natural languauge processors, and is simple and fast (DiBattista, 2021). As a commonly used, reliable, and easy-to-use package, it was selected for use here. A score was determined for each tweet, then the daily mean sentiment score was calculated. A daily median was selected as the unit of time, as it was considered that an hourly median may invite inherent bias (for example, people tweeting as soon as they wake up, or in the evenings, in different parts of the world may be reasonably expected to be more negative in areas and regions affected by war, for example). Also, a daily median would be less likely to be impacted by missing values, and would usually be expected to contain a relatively large dataset, with a spread of sentiment scores approximating a statistical distribution that could be modelled. For example, the 4th of June 2009 saw 3,354 tweets, and the 10th of May 2009 saw 25,984 tweets. As shown in Figure 1 both sentiment score distributions appear bimodal (non-parametric) (hence ths use of the median as the measure of central tendancy).

Tests for stationarity showed that the time series could be considered stationary (augmented Dickey–Fuller statistic: -5.26, DF p-value: 6.48 x 10-6; KPSS statistic: 0.55, KPSS p-value: 0.03), but additive decomposition (on the data after missing values were imputed) indicates seasonality, as shown in Figure 2. A significant proportion of the data - 69% when considering daily medians – consisted of missing values. These needed to be dealt with before any forecasting was attempted. The data is assumed to be MCAR, as there is no known reason why there would be no tweets at all on the missing dates, and there is no common factor amongst the missing data (Yang and Chiang, 2020).

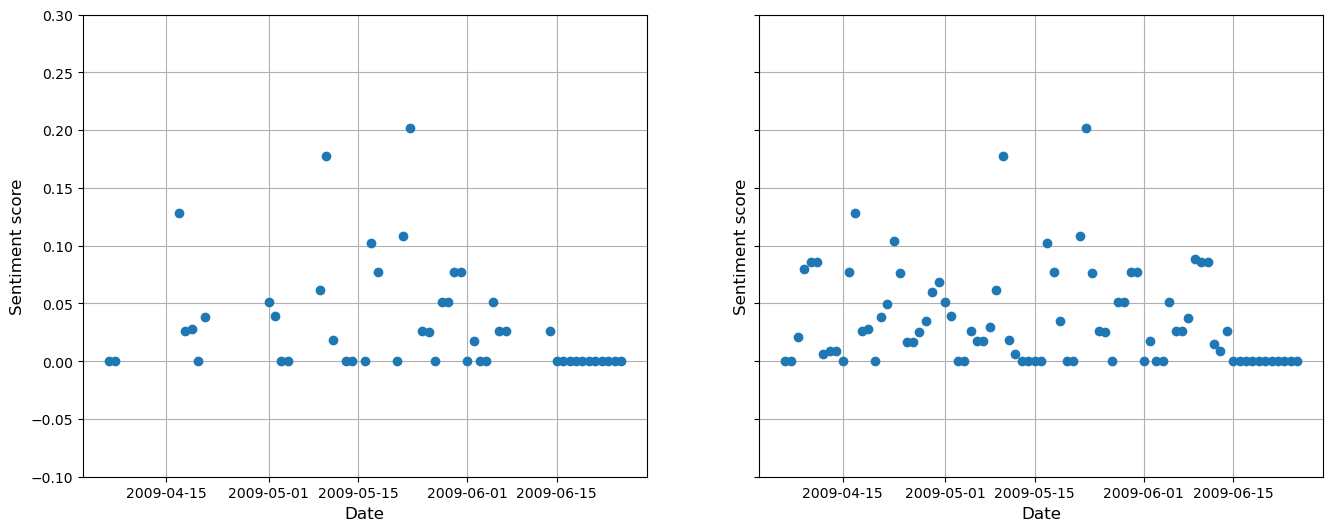
Figure 1: Two daily sentiment score distributions showing non-parametric nature of the data.

Not only was the sentiment score of all tweets in a day non-parametric, but so was the daily median score distribution for the 81 days available, as shown in Figure 3, and by the results of a [Shapiro–Wilk tes](https://en.wikipedia.org/wiki/Shapiro–Wilk_test)t (statistic=0.71, pvalue=2.51 x 10-8). As the data was non-parametric, non-parametric imputation, such as decision-tree and KNN methods, were most appropriate (Bertsimas, Pawlowski and Zhuo, 2018).

Figure 2: Additive decomposition on sentiment score time series data after missing values were imputed.

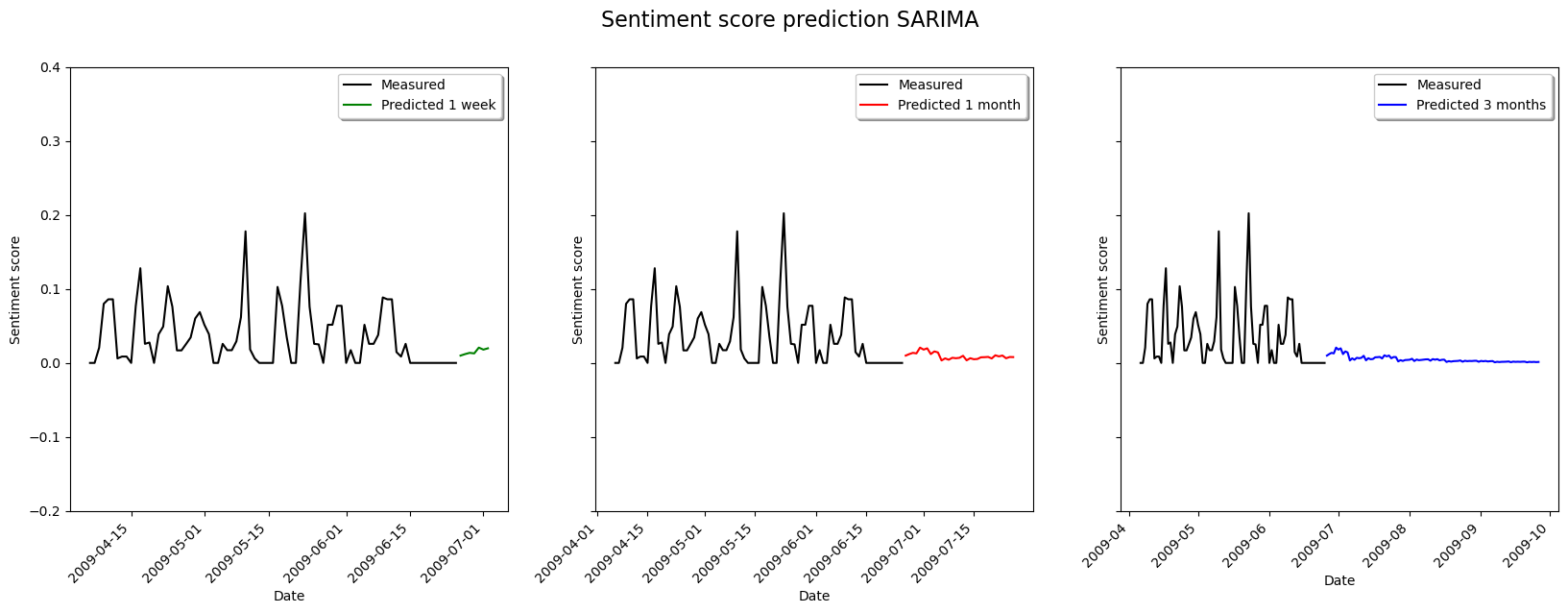
Figure 3: Histogram of the daily median setiment score

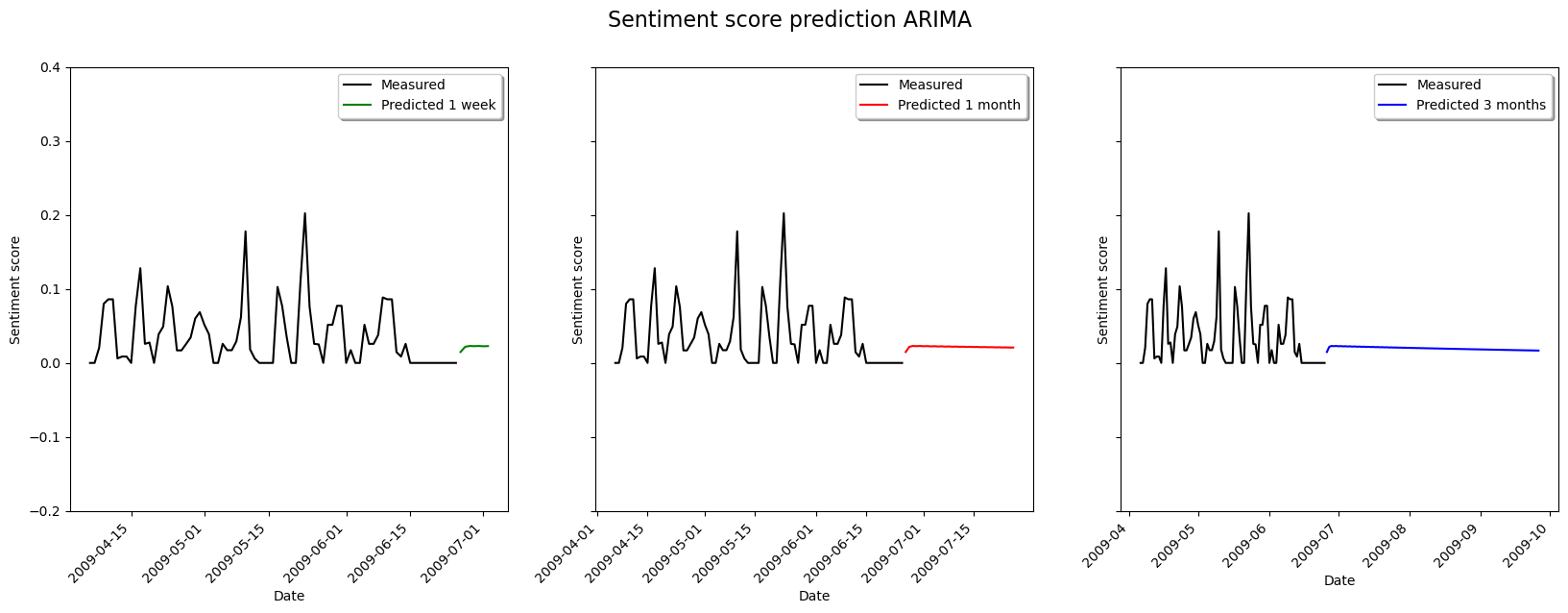
KNN imputation was used here, with 3 nearest neighbours selected. A comparison of the results with varying numbers of nearest neighbours can be seen on the dashboard. The day of the month and the month data were extracted for each data point, to allow the KNN imputer to work with these two aspects of the date independently. Ideally the data would have been split into test and train, and the parameters of the imputation selected by hyperparameter tuning, but there were no aspects of the time series long enough, and without missing values, to allow this. The imputed values in this case must be considered with some caution because of this, and because of the high percentage of them in the original dataset (which is potentially why there is still evidence of seasonaility in the rediduals in Figure 2). The time series after missing values were filled is shown in Figure 4.

Figure 4: Before and after imputation (left, showing missing data) and right (showing imputed data)

This data with imputed values was imported to two databases, Cassandra and Hbase, as shown in the Appendix. These two databases were also load tested using ycsb and the default ycsb time series load tester, tsworkloada. The commands used to run these are in the appendix, along with the results of each test. To use ycsb with Cassandra a change was needed to conf/cassandra.yaml, namely to uncomment two commitlog\_sync lines (Brianfrankcooper, n.d.), and to run the Cassandra load test in ycsb a ycsb keyspace had to be created, as shown in the appendix, and similarly a usertable had to be created in hbase.

Finally, a time-series forecast was generated using ARIMA and, more appropriately, SARIMA. ARIMA does not account for seasonality, which this time series possessed, as per Figure 2, while SARIMA does. Autocorelation, partial autocorrelation and differencing were used to find the appropriate values of p,d and q, and the seasonal term in SARIMA was selected by examing the seasonal aspect of the additive decomposition in Figure 2. As the data was trend free, d was set to 0. The autoregressive term, p, was set to 3 as the number of lags that crossed the significance threshold in a partial-autocorrelation plot of the data, and q – the number of lagged forecast errors - was also set to 3 from an autocorrelation plot of the data (Verma, 2022). The results of the 1 week, 1 month, and 3 month SARIMA forecasts are shown in Figure 5. For comparison, the ARIMA model, which does not account for seasonality, is shown in Figure 6. This latter model forecasts the same median sentiment (just above 0) for each day into the future, with a slight downward trend. In contrast, the SARIMA model peters out to a median sentiment of 0 toward the end of the 3 month period.

Figure 5: SARIMA forecasts of sentiment score over 1 week, 1 month, and 3 month periods.

Figure 6: ARIMA forecasts of sentiment score over 1 week, 1 month, and 3 month periods.

**Results and Discussion**

While sentiment analysis is undoubtedly useful, and is very widely used, there are other – potentially equally informative – analyses that may be applied to data to extract subjective meaning from it, such as intent analysis – extracting the authors intention, e.g. marketing or querying, and contextualised semantic analysis which is segregating messages into groups such as different product areas, or staff vs product feedback, etc. (Gupta, 2018).

**Conclusion**

**References**

Beri, A. (2020). *SENTIMENTAL ANALYSIS USING VADER*. [online] Medium. Available at: <https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664>.

Bertsimas, D., Pawlowski, C., and Zhuo, Y. D. (2018). From Predictive Methods to Missing Data Imputation: An Optimization Approach. *Journal of Machine Learning Research*, 18, pp.1-39.

DiBattista, J. (2021). *The Best Python Sentiment Analysis Package (+1 Huge Common Mistake)*. [online] Medium. Available at: <https://towardsdatascience.com/the-best-python-sentiment-analysis-package-1-huge-common-mistake-d6da9ad6cdeb>.

Brianfrankcooper, GitHub. (n.d.). *Improvements for HBase and Cassandra clients by jbellis · Pull Request #98 · brianfrankcooper/YCSB*. [online] Available at: https://github.com/brianfrankcooper/YCSB/pull/98/files [Accessed 7 Nov. 2023].

Gupta, S. (2018). *Sentiment Analysis: Concept, Analysis and Applications*. [online] Towards Data Science. Available at: https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17.

‌

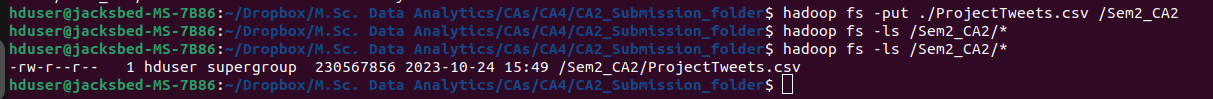
Hutt, S. (2023). *7 Advantages Of Twitter Sentiment Analysis*. [online] Available at: https://ecommercefastlane.com/7-advantages-of-twitter-sentiment-analysis/ [Accessed 7 Nov. 2023].

Verma, Y. (2022). *Quick way to find p, d and q values for ARIMA*. [online] Analytics India Magazine. Available at: <https://analyticsindiamag.com/quick-way-to-find-p-d-and-q-values-for-arima/>.

Yang, L. and Chiang, J.A. (2020) ‘Use case and performance analyses for missing data imputation methods in big data analytics’, *Proceedings of 2020 6th International Conference on Computing and Data Engineering* [Preprint]. Doi:10.1145/3379247.3379270.

**Appendix**

**Big data operations carried out ouside of Jupyter notebook**

Figure 7: Saving raw data to Hadoop HDFS before MapReduce carried out.

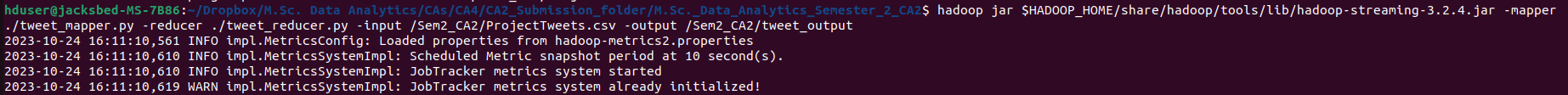
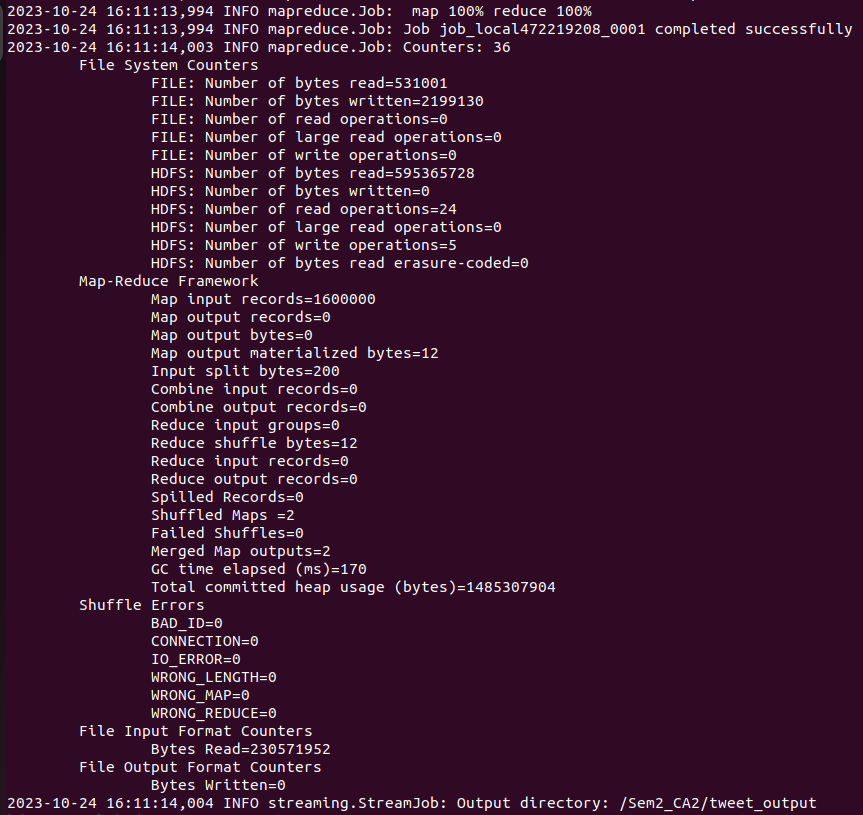
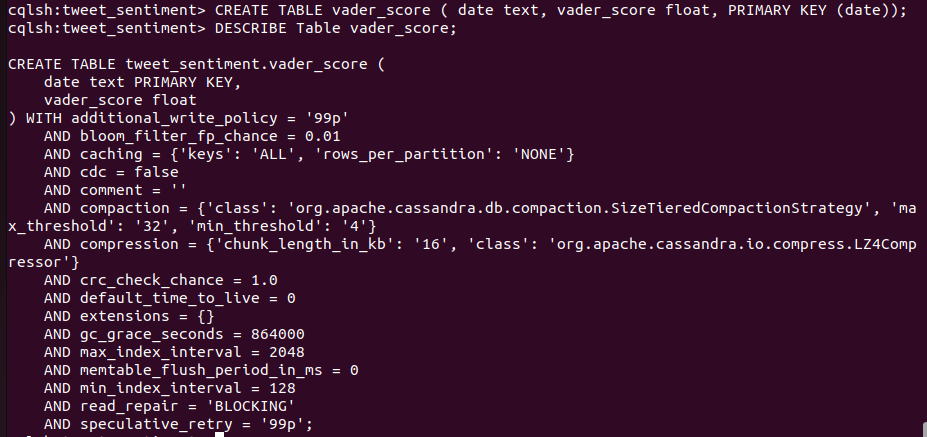
Figure 8: MapReduce program in operation

Figure 9: Mapper section complete

Figure 10: MapReduce complete

Figure 11: Creating a table for the sentiment score daily median data in Cassandra, with the date as the primary key.

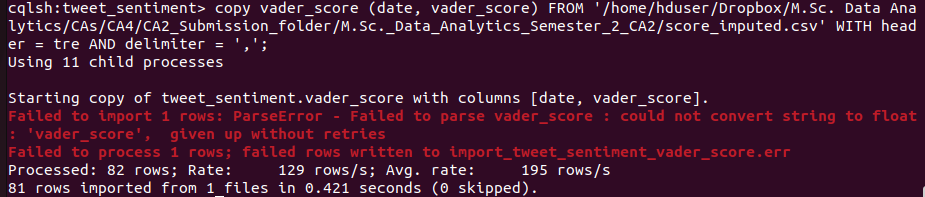
Figure 12: Copying the median daily sentiment score data to Cassandra

Figure 13: Selecting data from the sentiment score table in Cassandra

Figure 14: Creating a table for the sentiment score daily median data in HBase

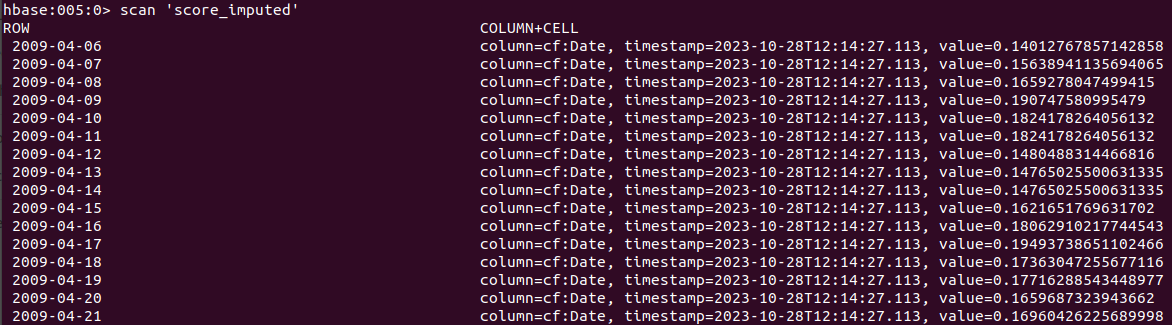
Figure 15: Scanning the HBase table for the sentiment score daily median data.

Figure 16: ycsb keyspace creation for the Cassandra load test

Figure 17: Running a ycsb load test on Cassandra cql