**CCT College Dublin**

**Assessment Cover Page**

| **Module Title:** | Advanced Data Analytics, Big Data Storage and Processing |
| --- | --- |
| **Assessment Title:** | A Time-series Forecast of Tweet Sentiment Scores. |
| **Lecturer Name:** | David McQuaid, Dr. Muhammad Iqbal |
| **Student Full Name:** | Paul Ryan |
| **Student Number:** | sbs23013 |
| **Assessment Due Date:** | 10/11/2023 |
| **Date of Submission:** | 10/11/2023 |
| **Word Count** | 3 |
| **GitHub Repository** | https://github.com/paulr28/.git |

**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. |
| --- |

**Abstract**

*This paper looks at a dataset of tweets scraped from Twitter. They are stored in a MySQL database and using Apache Spark, sentiment scores are generated and the data is then stored using Apache Hive. The sentiment scores are then used in a time series analysis, with predictions being made at 7 day, 31 day, and 90 days periods. Two time series models are used, an Arima and a RandomForest, with the RandomForest chosen as the best performing, and used as the basis for an interactive dashboard created using Dash in Python.*

**Keywords:** MySQL, Apache Spark, Apache Hive, big data processing, sentiment analysis, time series analysis, ARIMA, RandomForest

**Introduction**

Twitter, now named X, is a social media platform where users can post messages of 140 characters or less, as well as other media types. The tweets can be on any topic and may be viewed or reposted by other users.

The dataset provided has roughly 1.6million tweets, with the tweet, the user and the date of the tweet included. With this information, it is intended to create a sentiment analysis of the tweets. This will be done while processing the data and moving it from one database to another, using big data storage and processing techniques such as Apache Spark and the Hadoop Distributed File System. The sentiment score is then to be analysed and used as the basis for a time series analysis. This analysis will look at two different models, with parameter tuning and predictions being made with both. The most effective model will be chosen to predict the sentiment of future tweets, with time periods of 1 week, 1 month and 3 months into the future.

These future predictions will then be plotted and displayed using an interactive dashboard, allowing for the selection of the time period for which you would like to see the predicted values.

**Materials**

Data Source / Initial Loading

*Data Source*

The initial data source for this project is a csv file named ‘ProjectTweets’, provided by the CCT College moodle page at ‘<https://moodle.cct.ie/mod/assign/view.php?id=136451>’. When loaded as a dataframe into a Python environment using the pandas library it can be seen to hold 1.6 million tweets extracted from April to June, with the columns for datetime, user, the tweet text itself, and others.

As the datasource will be initially stored into a MySQL database before processing, an initial processing where special characters are removed from the text column is performed, this is to avoid any separation issue when loading the data. The amended csv file is saved as ‘ProjectTweets\_2’

*Initial Loading*

This csv file is then loaded into a MySQL database named ‘Tweets’, into a table created for this purpose, named ‘tweetable’. It is stored directly via the MySQL terminal using a ‘DATA INFILE’ command to store the entire csv at once.

**Methods**

Database Comparison

*YCSB*

The Yahoo! Cloud Service Benchmark (YCSB) tool allows for comparison of database types based on a variety of different criteria, using custom or provided sample datasets. In this case the test will be on a MySQL database versus a MongoDB NoSQL database. In this case the provided ‘workloada’ is used as it is “a mix of 50/50 reads and writes” (Busbey, 2020), which is suitable for the type of work we will be using it for.

Both the MySql and MongoDB services are started via the command terminal, and a MySQL table is created to store the sample workload, MongoDB will create the table on loading.

Once the YCSB service has been configured the workload is executed against the two different databases and the results are stored in .txt files for analysis.

Sentiment Analysis

*MySQL Retrieval*

The data is retrieved from the MySQL database using Apache Spark running in a Jupyter notebook. Spark was chosen as it is ‘an open source open source data-processing engine for large data sets’ (IBM, n.d.), which will allow for loading, processing, and storage of the twitter data.

A Spark session is created, the MySQL connection settings are established, and the dataset is loaded from the ‘tweetable’ to a Spark dataframe.

A sentiment analysis function is created using the ‘HuggingFace’ library to import a sentiment analysis pipeline using the ‘distilbert-base-uncased-finetuned-sst-2-english’ model. This is then registered as a user defined function (UDF) in the Spark session. A second UDF is created and registered which will shorten the date column to only include the day month and year, removing the timezone.

*Processing*

After creating the Spark df containing the tweet dataset, the two sentiment and date processing UDF’s are applied to the ‘text’ and ‘date’ columns respectively. The resultant columns are ‘sentiment\_score’ and ‘full\_date’. The ‘sentiment\_score’ column is changed from string to float using the ‘cast’ function. The two created columns are then used to create a new dataframe, ‘df1’, with just ‘full\_date’ and ‘sentiment\_score’, which is converted to a pandas dataframe. This is saved as a csv file, ‘ProjectTweets\_3’ as a backup.

*Apache Hive Storage*

To store the resulting ‘df1’ using Apache Hive, it is first necessary to start the Hadoop Distributed File System (HDFS) and its YARN monitor, by running the ‘start-dfs.sh’ and ‘start-yarn.sh’ commands within the terminal. It is then necessary to start the Hive data warehouse system, again by calling ‘Hive’ within the terminal. Once all systems are running, a Spark session is created in the notebook with the Hive connection settings. The pandas df is converted to a Spark dataframe, and using the ‘.write.saveAsTable()’ command, the data is written to a new table in the database. Selecting the top five rows from the resulting table shows the data is stored there correctly.

Exploratory Data Analysis (EDA)

*Processing*

Taking the created dataset, an initial overview of it shows 1,598,314 rows with 2 columns. The ‘full\_date’ column is converted from a string to datetime, and a list of missing dates is created based on the first and last date contained in the dataset.

The missing dates are added, with the corresponding sentiment scores being set to NA. The data is then grouped by the date column, aggregating the ‘sentiment\_score’ column by mean value.

This aggregated set is then checked for NA sentiment scores, and those that are missing are filled using linear interpolation, “used to estimate unknown values that lie between known values, [particularly for] evenly-spaced data” (Columbia University, n.d.).

*EDA*

EDA was performed on the aggregated date, with the shape, data types and other typical features of the dataset being examined, and the sentiment score plotted using a box plot. Minimum, maximum and mean sentiment scores were examined. The entire sentiment score was also plotted via line plot across the time period represented.

Time Series Analysis

*Data Processing*

Before creating a time series analysis, certain tests were performed to check for trend and seasonality as well as whether the data is stationary. “A stationary time series is one whose statistical properties such as mean, variance, autocorrelation, etc. are all constant over time” (Duke University, 2019).

A Dickey-Fuller test is applied to the data with the results indicating that the data is non-stationary. The data is made stationary by applying differentiation, which can “help stabilise the mean of a time series by removing changes in the level of a time series, and therefore eliminating (or reducing) trend and seasonality” (Hyndman and Athanasopoulos, 2018). The NA value created by this transformation is filled using the column mean, interpolation is not possible in this case as there is no prior value.

Seasonality, trends and residuals are plotted, with a seasonally decomposed column being created containing the sentiment scores. The created columns are tested to find which has the lowest p-value, and is therefore most stationary.

*ARIMA Model Overview*

The Autoregressive Integrated Moving Average (ARIMA) model is a popular time series forecasting model, able to represent several different types of time series and particularly suitable for “data [which has been] differenced in order to achieve weak stationarity” (Lim, 2019)

The data is normalised using a MinMaxScaler() function, to attempt to improve the results of the model. The data is then split into training and testing sets, with 65 days in the train set and 16 in the test, roughly an 80/20 split.

*Hyperparameter Tuning*

A function is created to find the best parameters for the ARIMA model, by testing a range of p, d and q values, and returning the combination which provides the best result. This is run on the training set, and the results are applied as parameters in the ARIMA model.

*Model*

The model is then created and fitted using the training data. A report is printed and the residuals are plotted. Predictions are made against the test set, the result is plotted and the Root Mean Square Error (RMSE) is calculated.

Predications are also plotted against the entire dataset to visualise the variance.

*Predications*

Using the model created, predictions are made at 7 day, 31 day and 90 day periods into the future, representing a week, a month, and three months respectively.

These are then visualised as an extension of the existing time series data.

*Forecaster / RandomForest Model Overview*

ForrecasterAutoReg is a tool which allows regressors such as RandomForest to be used as a recursive autoregressive (multi-step) forecaster. “The library is built using the widely used scikit-learn API” (Ortiz, 2021), enabling easy integration into Python workflows.

A small amount of data processing is done where the date column is set as the index and the frequency is set to daily. Train and test sets are again created with 65 and 16 occurrences respectively.

*Hyperparameter Tuning*

Hyperparameter tuning is performed by creating sets of parameters; in this case lags, the number of estimators and the maximum depth, and using a grid search to return the combination which provides the optimal parameters. The results are printed and the optimal parameters are chosen for the final model.

*Model*

The model is created and fit using the training set, and the predictions are made against the test set. The predictions are then plotted and the MSE is calculated.

The entire dataset is then used to train the model, as future predictions will be made.

*Predications*

Using the model created, predictions are made at 7 day, 31 day and 90 day periods into the future, representing a week, a month, and three months respectively.

These are then visualised as an extension of the existing time series data.

Interactive Dashboard

*Dashboard*

A dashboard is created using ‘Dash’, a framework for “rapidly building data apps in Python” (Plotly, 2023). The layout is created with titles and subtitles to describe the data, and a dropdown selector created to switch between time periods. A dark theme is chosen for the surrounding dashboard, this will contrast and highlight the chart being displayed. The callback function is created to enable the switching of dashboards shown based on the dropdown value created.

This is then run and can be viewed via web browser using the address provided in the output.

**Results**

Database Comparison

The results of the database comparison indicate that for the MySQL database;

* The total execution time was 11.25 seconds
* The average throughput was 88.88 operations/sec (across all threads)
* There were 1000 insert operations, with associated average, min, max, 95th and 99th percentile latencies of 9419.248 us, 2438 us, 2439 us, 19279 us, and 29583 us, respectively.
* All 1000 update operations had a return code of zero (success in this case)

And for the MongoDB database;

* The total execution time was 2.42 seconds
* The average throughput was 409.5 operations/sec (across all threads)
* There were 1000 insert operations, , with associated average, min, max, 95th and 99th percentile latencies of 1113.168 us, 130 us, 282623 us, 4563 us, and 11647 us, respectively.
* All 1000 update operations had a return code of zero (success in this case)

Sentiment Analysis

*Processing*

The retrieval of the tweet dataset from the MySQL database using Apache Spark was successful. Applying the UDF’s, including the sentiment analysis model, was also successful and checking the data type of the “sentiment\_score” showed it was a string, it was cast to a float type. The new dataframe was written to the Apache Hive database.

From there it was loaded as a pandas dataframe. The aggregation and imputation of missing values results in a dataframe of 81 rows and 2 columns.

*EDA*

Plotting the sentiment scores using a box plot shows that the mean sentiment is above 1.0, meaning its trending positive, however there are a greater number of more extreme outliers on the negative side of the score.

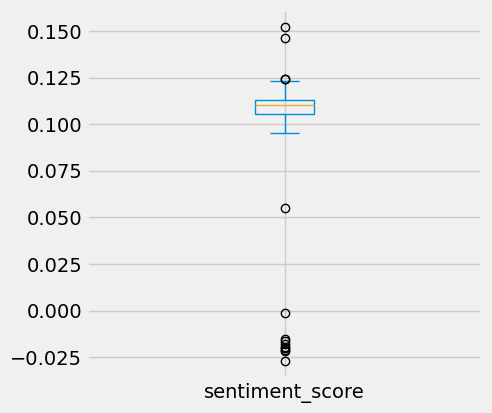


Figure 1: Boxplot of generated sentiment scores, Apr 6th 2009 - June 25th 2009 (Source: twitter.com)

Sentiment score metrics are records as follows:

* Mean: 0.096523
* Min: -0.027289
* 25%: 0.105577
* 50%: 0.110279
* 75%: 0.112952
* Max: 0.152430

The overall sentiment score was also plotted.

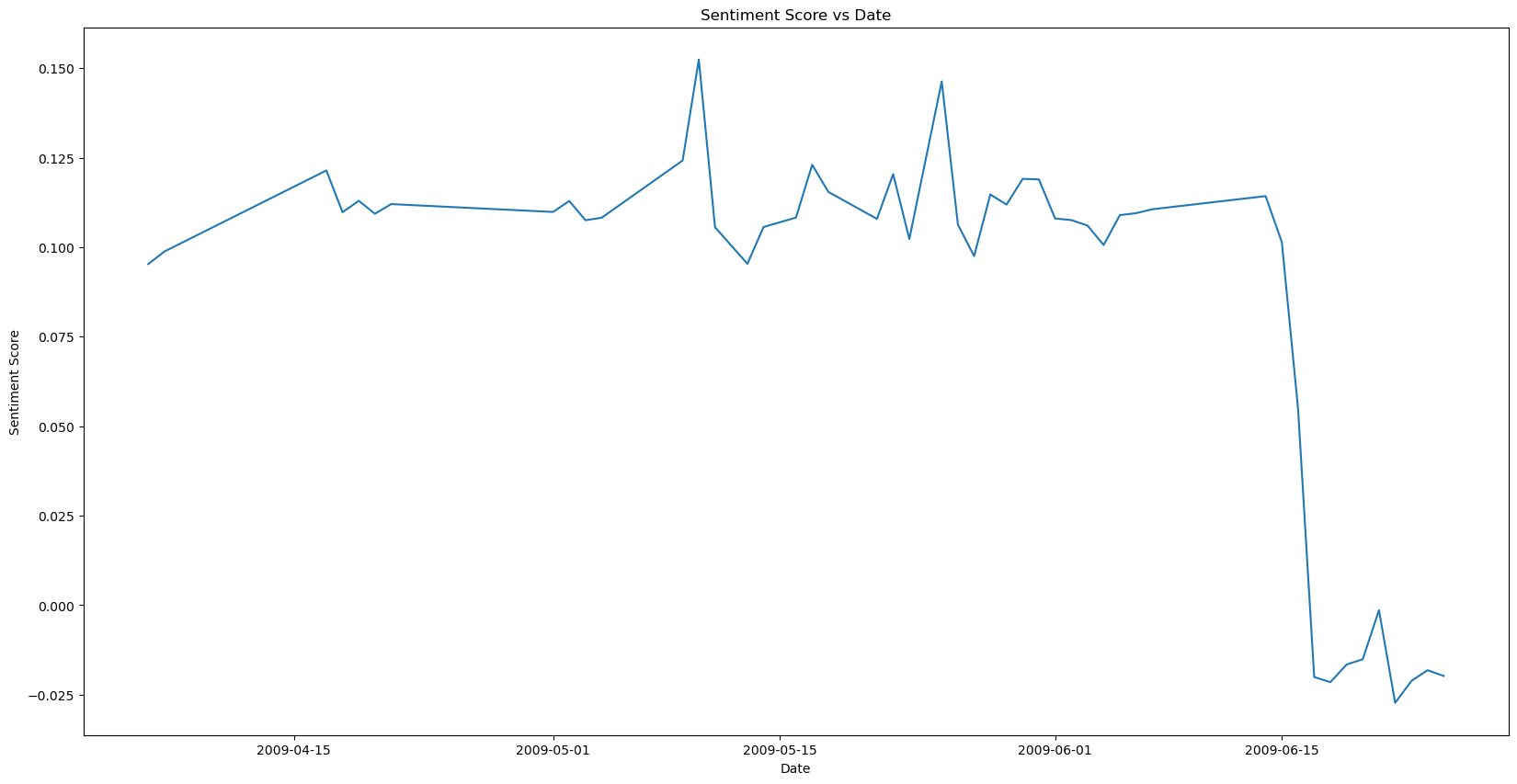


Figure 2: Line plot of sentiment score over time, Apr 6th 2009 - June 25th 2009 (Source: twitter.com)

Time Series Analysis

*Data Processing*

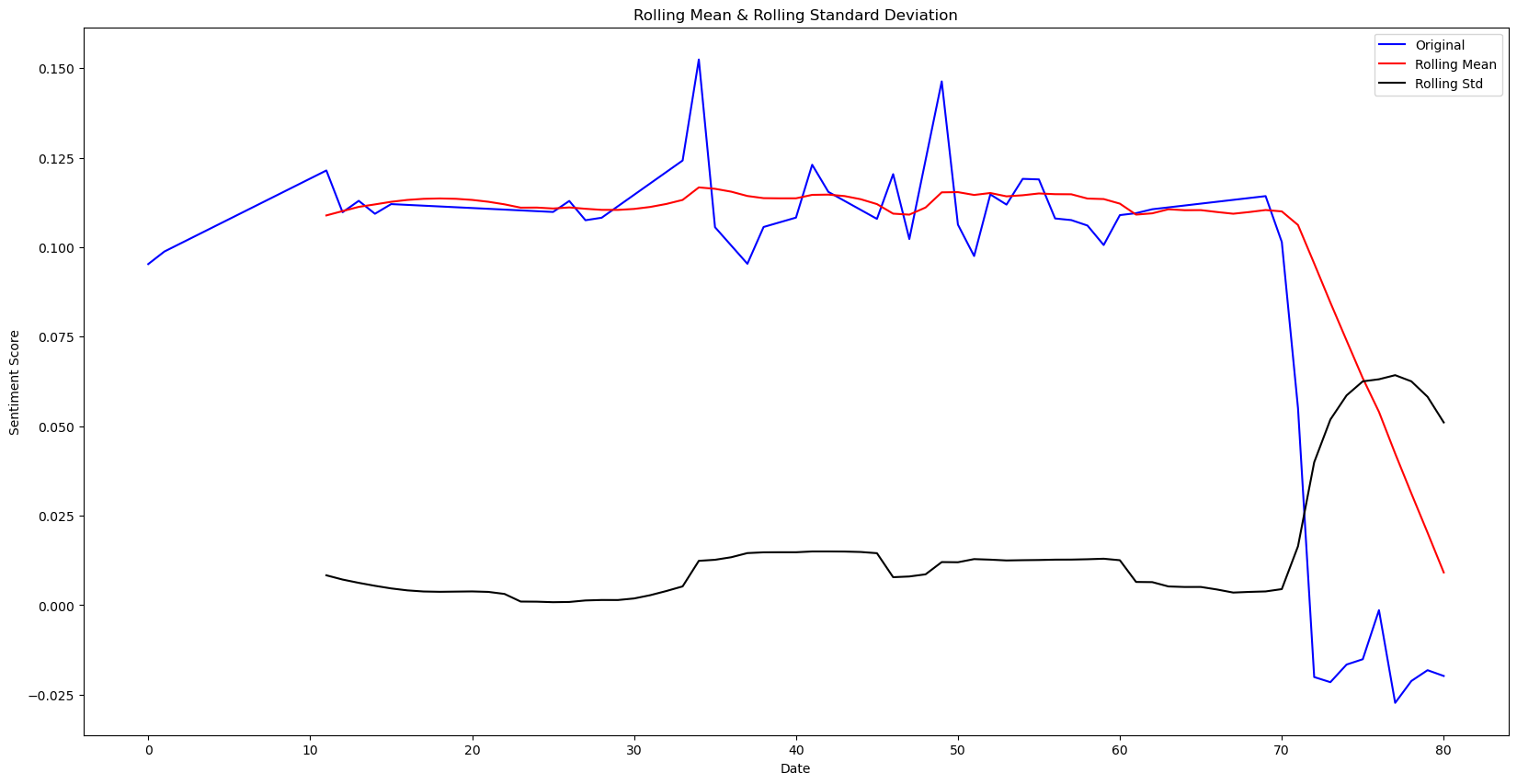
**

Figure 3: Line plot of sentiment score, rolling mean and rolling average over time, Apr 6th 2009 - June 25th 2009 (Source: twitter.com)

Viewing the results of the initial stationary data check, it can be seen that the rolling average is significantly lower than the original and rolling mean values. The p-value of the Dickey-Fuller Test is 0.933, indicating that the data is not stationary.

After differentiating, the Rolling standard is much more in line, and the p-value is 1.824254e-12, indicating that the data is now stationary

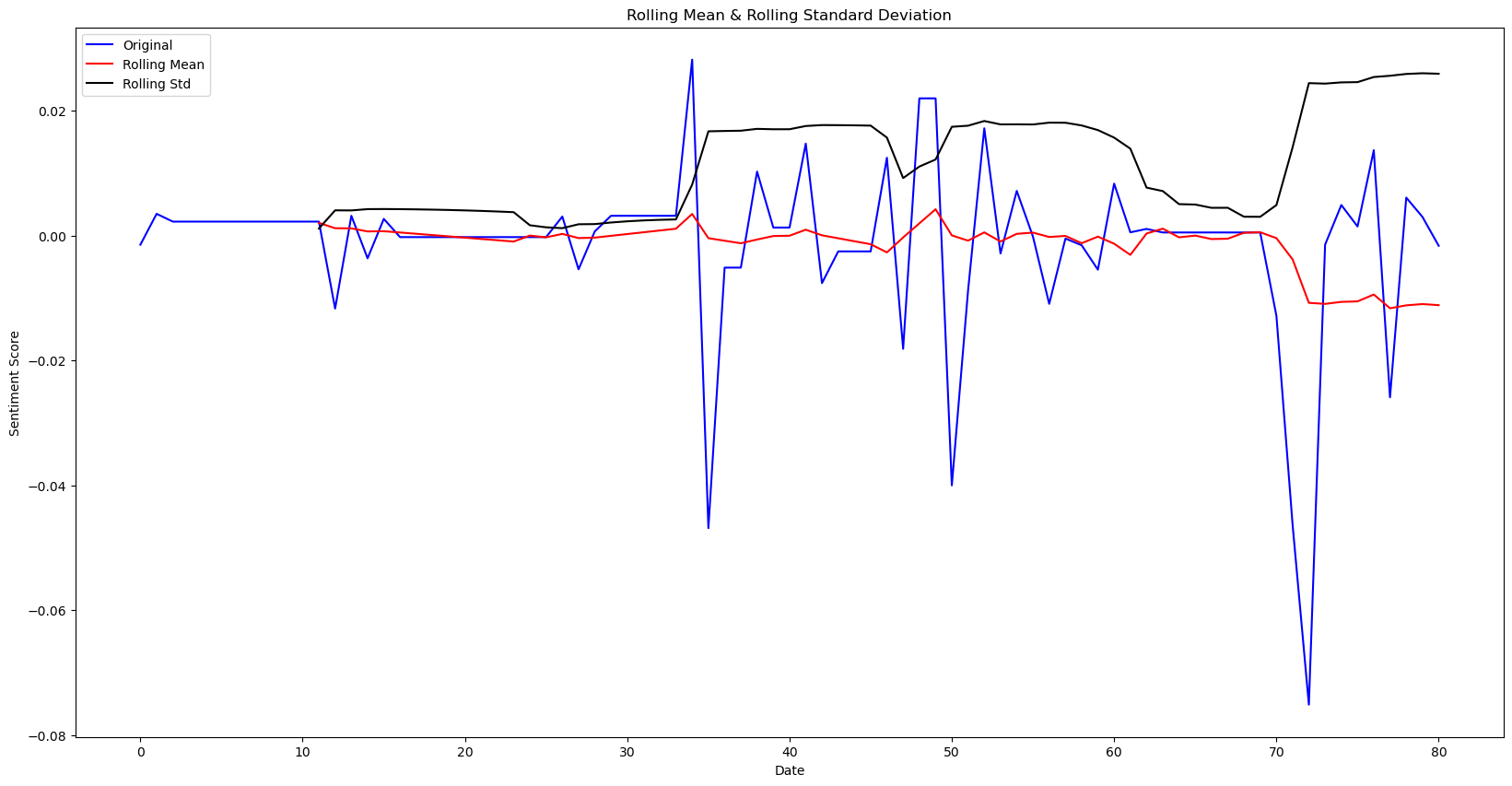


Figure 4: Line plot of stationary sentiment score, rolling mean and rolling average over time, Apr 6th 2009 - June 25th 2009 (Source: twitter.com)

After seasonal decomposition, the p-value became 1.795377e-11 which was even more stationary, and the reason this set of values was chosen for the model.

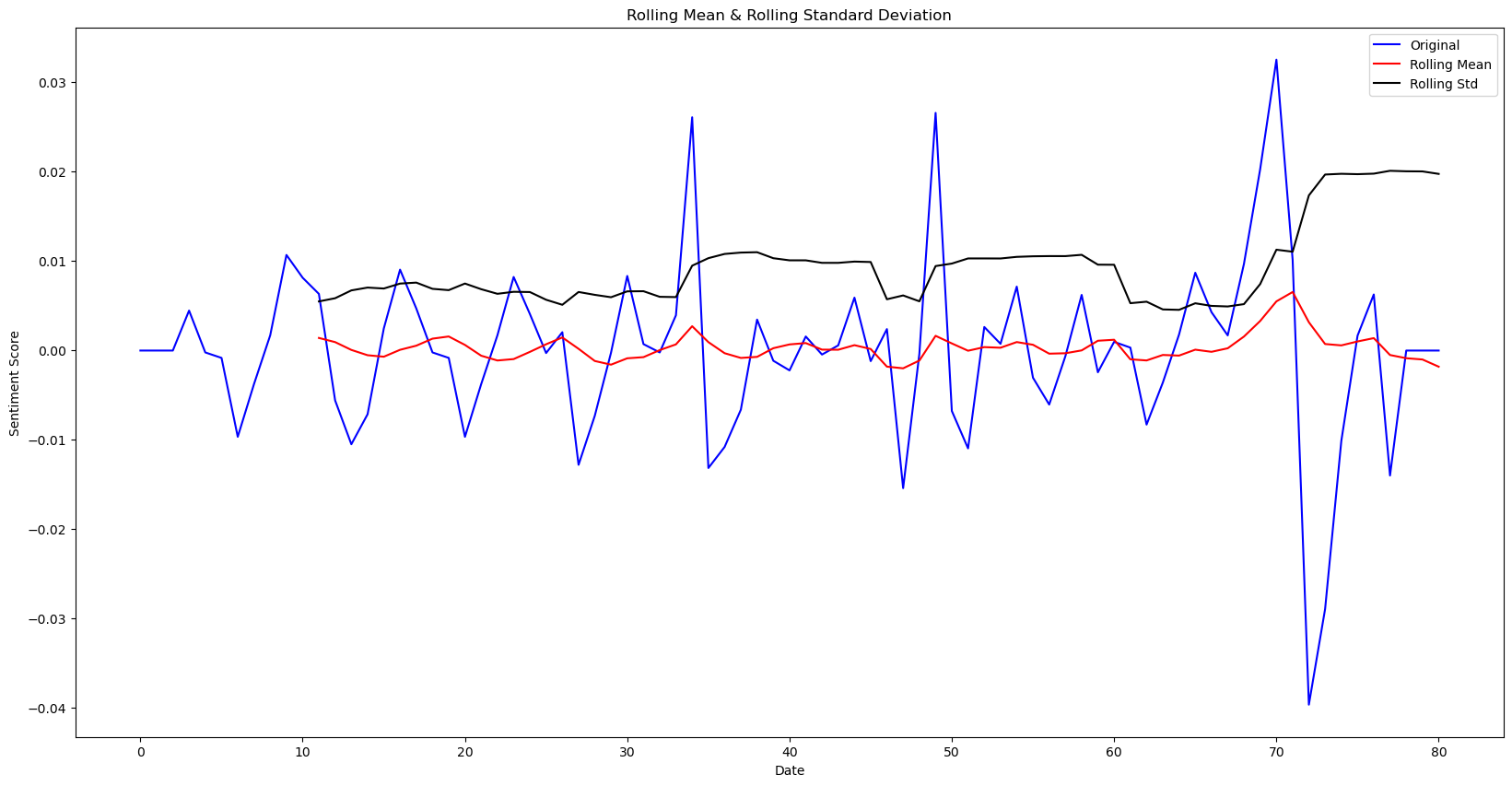


Figure 4: Line plot of seasonally decomposed sentiment scores, rolling mean and rolling average over time, Apr 6th 2009 - June 25th 2009 (Source: twitter.com)

*Arima*

*Hyperparameter Results*

The p, d, q value combination found to give the best model was ‘(2, 0, 2)’. These were applied to the ARIMA model used.

*Model*

When tested on the test set, the model produced a RMSE of 0.24 and an R2 score of -45.23

A low RSME error is good, indicating a good prediction of values, however the negative R2 score indicates that the model is not fitted correctly, does not explain the variance in the original data, and therefore is not fit for predictions.

*Predictions*

Predications at 1 week, 1 month and 3 month periods are created and then plotted.

Looking at the predictions the issues of the negative R2 score becomes apparent as the predictions are essentially flat at 0.000 for the entirety of the time frame.

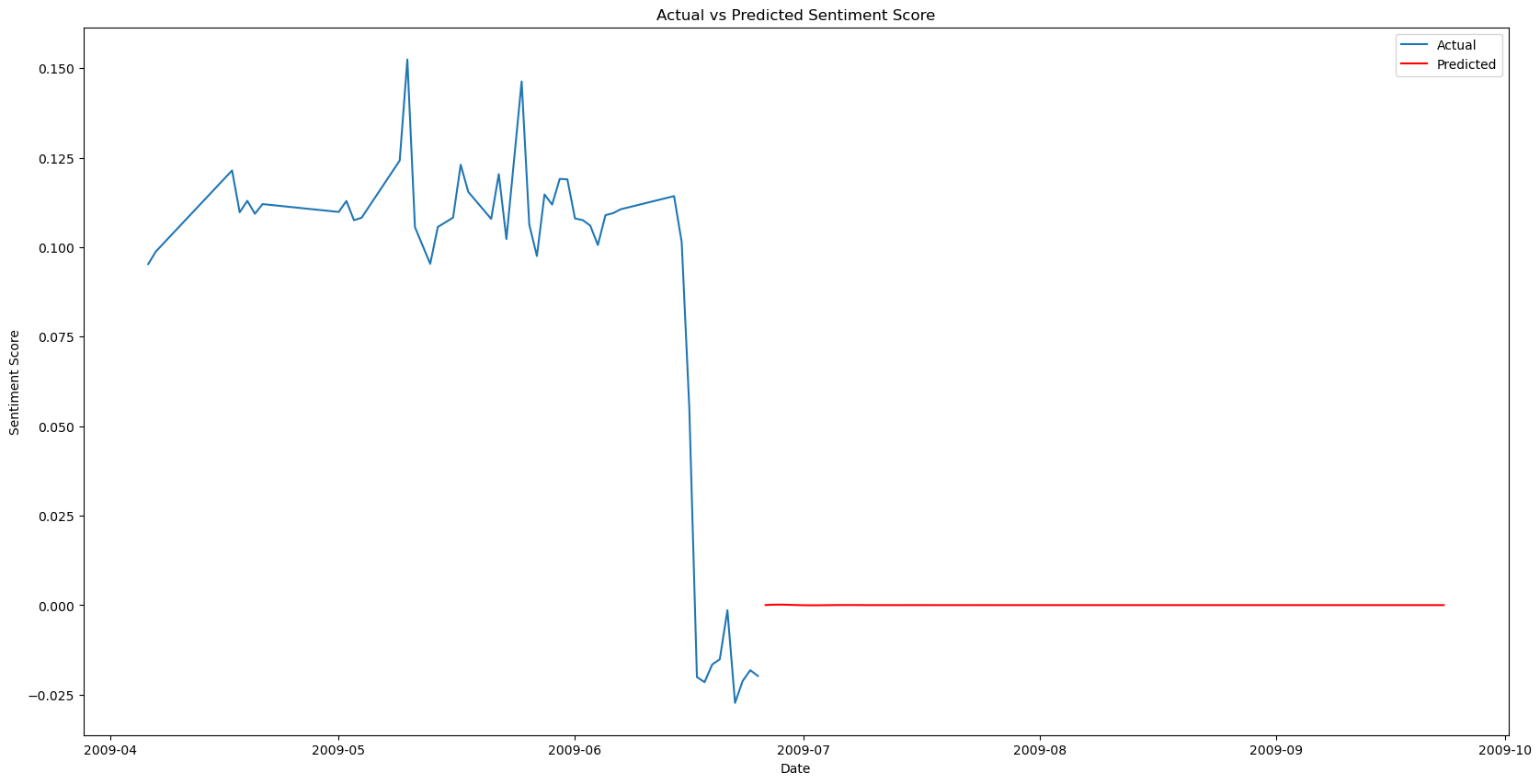


Figure 5: Line plot of sentiment scores, with 90 days predictions appended, Apr 6th 2009 - Sep 23rd 2009 (Source: twitter.com)

*Forecaster*

*Hyperparameter Results*

After printing the results of the tuning grid the best parameters were found to be:

* Lags: 20
* Max Depth: 3
* Number of Estimators: 100

These were then applied to the model.

*Model*

The model was created and fit, and then tested using the test set.

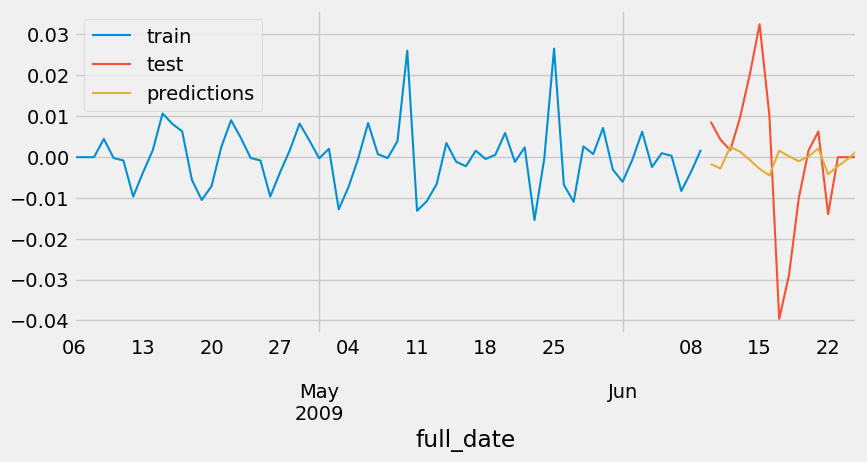


Figure 6: Line plot of sentiment scores, with last 16 days prediction added, Apr 6th 2009 - Sep 23rd 2009 (Source: twitter.com)

The MSE for the model created was 0.311.

The model was then trained on the entire dataset to create future predictions

*Predictions*

The predictions were forecasted at 1 week, 1 month, and 3 months periods, which are each plotted against the full dataset. The predictions show much more variance than the ARIMA model predictions, although the further out the predictions the closer the predictions converge towards 0.

Interactive Dashboard

*Dashboard*

The interactive dashboard was created and used the RandomForest version of the predictions for its data selection.

When run, it can successfully be launched on a web browser and the three different time period predictions can be chosen from.

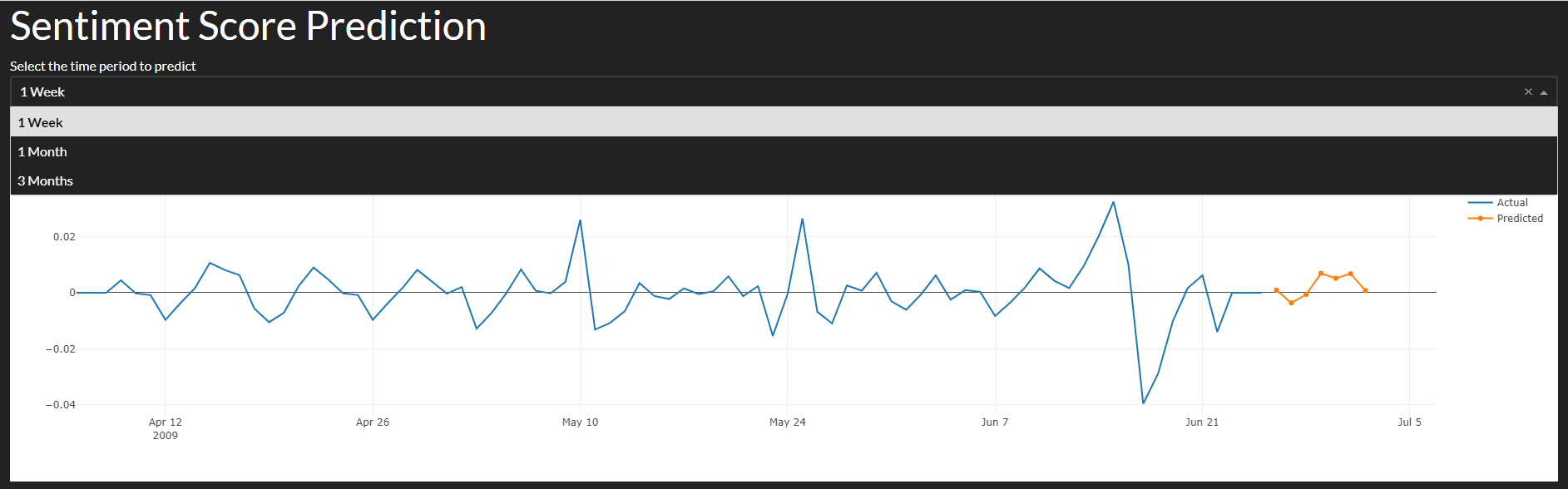


Figure 7: Line plot of sentiment scores, with last 16 days prediction added, Apr 6th 2009 - Sep 23rd 2009 (Source: twitter.com)

**Discussion**

Databases

For the initial choice of database to store the ‘ProjectTweets.csv’, a comparison of MySQL and MongoDB was made. MongoDB supports schemaless data, which is advantageous if the data you are storing is heterogeneous. \* MySQL requires a predefined schema for each table which can be more rigid and less flexible, however in this case the data is simple and flat and so suitable for the MySQL structure.

When using PySpark both databases can be accessed, with read and write capabilities. MySQL is generally faster at selecting data from a table, than MongoDB is from a collection, particularly if the data is indexed and the queries are simple, both of which are true in this case. For those reasons MySQL was chosen as the database to store the initial tweet dataset.

Data Processing

Apache Spark was chosen as the data processing tool as it is easily integrated with Python using PySpark. PySpark the Python API for Apache spark, allowing for the performance of “real-time, large scale data processing in a distributed environment using Python” (Apache, 2023).

Apache Hive was also chosen as a data warehouse tool. Hive is built on top of Hadoop and supports storage through the hdfs. It allows users to “read, write and manage petabytes of data using SQL” (Apache Software Foundation, 2013).

Machine Learning Models

When choosing between machine learning models there can be a number of factors to take into consideration. Interpretability of a model is how explainable the result was arrived at. ARIMA models provide a clear explanation of how the past values and error terms affect the current value. RandomForest models are more ‘black-box’ as they do not show how the individual trees perform their decisions. \*

AutoReg RandomForest models have a larger selection of hyperparameters which can be tuned to increase the performance of the model. Number of trees, maximum depth, minimum node size, lag, are some of the parameters, whereas ARIMA models only have the autoregressive, the differencing and the moving average components. The larger number of parameters for the RandomForest allows for greater customisation of the model, however discovering the best combination of parameters can be more computationally expensive.

**Conclusion**

After processing the data, the sentiment analysis pipeline provided by the ‘huggingface’ python library was the ideal choice to perform the analysis on the entire dataset using a UDF in PySpark. When the scores were initially plotted over time, it was seen that there was significant variation in the sentiment of the tweets. The ARIMA model did not perform well when predicting future sentiment scores, and was not chosen for this reason. The RandomForest forecaster was more adept at predicting the future values and it was decided to be the model used when creating the predicted values which would be used in the final dashboard.

The dashboard was created using the dash.plotly python library, and consisted of a title, a sub-title indicating a choice of time-periods, a drop-down menu from which each time period could be chosen, and a line chart with the full data set and selected prediction range displayed. A dark theme was chosen for the dashboard as it contrasts strongly against the white chart.

**References**