**CCT College Dublin**

**Assessment Cover Page**

|  |  |
| --- | --- |
| **Modules Title:** | Advanced Data Analytics  Big Data Storage and Processing |
| **Assessment Title:** | MSC\_DA\_BD\_ADAv6 |
| **Lecturers Name:** | *David McQuaid*  *Muhammad Iqbal* |
| **Student Full Name:** | Alaa Shammary |
| **Student Number:** | SBA23361 |
| **Assessment Due Date:** | *23/05/2025 11.59pm* |
| **Date of Submission:** | 23/05/2025 11.59pm |

**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. |

Contents

[Introduction 3](#_Toc198936020)

[Big Data Processing and Storage 3](#_Toc198936021)

[Data Sources and Initial Preparation 3](#_Toc198936022)

[Data Storage 3](#_Toc198936023)

[Data Processing Environment 4](#_Toc198936024)

[Merging Processed Datasets and Final Storage 4](#_Toc198936025)

[Exporting the Final Dataset to CSV 5](#_Toc198936026)

[Comparative Analysis 6](#_Toc198936027)

[Advanced Data Analytics 6](#_Toc198936028)

[Exploratory Data Analysis (EDA) 6](#_Toc198936029)

[Feature Engineering 7](#_Toc198936030)

[Processing Tweets and Sentiment Score 8](#_Toc198936031)

[Analysis and Forecast of Adjusted Close Prices Using Time-Series Models 10](#_Toc198936032)

[Tufte’s Principles in the Dashboard 12](#_Toc198936033)

# Introduction

This project investigates how public sentiment expressed on Twitter interacts with daily stock market movements and whether the two sources of information can be fused to improve short horizon price forecasts. The study combines 10 000 stock tagged tweets and one year of daily price data for large companies. After preprocessing, each tweet is scored for polarity; the resulting sentiment time series are aggregated to the ticker day level and joined to the corresponding market data.

# Big Data Processing and Storage

This section outlines the integration and processing of stock market and tweet datasets using distributed big data tools, specifically Cassandra, MongoDB, and Apache Spark.

## Data Sources and Initial Preparation

Two main datasets were used:

Tweet Dataset (stocktweet.csv): 10,000 entries containing tweet ID, timestamp, stock ticker, and tweet content.

Stock Prices Dataset (5 CSVs for AAPL, AMZN, MSFT, NVDA, TSLA): Each containing daily prices with columns including date, open, high, low, close, adjusted close, and volume.

The datasets were read using Pandas and a ticker column was added to each stock data frame for identification during merging and processing.

In this project, I implemented a robust pipeline for storing and processing large scale financial and social media data using modern Big Data technologies, with a focus on Cassandra, MongoDB, and Apache Spark. The rationale for the architecture and tool selection was guided by the nature of the data, scalability requirements, and the objective of performing real time and batch analytical operations efficiently.

## Data Storage

Stock Market Data (Structured Data):

The historical stock prices for companies such as AAPL, TSLA, MSFT, AMZN, and NVDA were stored in Cassandra, a NoSQL, distributed and highly scalable database optimised for time series data. Cassandra's schema was designed with a composite primary key (ticker, date) to enable fast queries by stock symbol and date range.

Justification: Cassandra was chosen for its high write throughput, fault tolerance and excellent support for time-based queries, which is ideal for stock price data that grows continuously over time.

Tweet Data (Semi-structured Text Data):

Tweets related to the selected stock tickers were stored in MongoDB, a flexible document-oriented database. Each tweet was inserted as a JSON-like document into a collection.

Justification: MongoDB was selected for its natural fit with semi structured data such as tweets, and its dynamic schema enabled quick ingestion and retrieval of diverse text entries.

## Data Processing Environment

Apache Spark:

Data processing, transformation and feature engineering were conducted using Apache Spark, specifically through PySpark. Spark was configured to read stock data from Cassandra and tweet data from MongoDB via appropriate connectors.

Key processing activities:

* Handling missing values (e.g., lagged values in stock data).
* Generating lagged close prices and future price columns (e.g., close\_t\_plus\_1, close\_t\_plus\_3).
* Creating new features such as daily returns and moving averages using Spark SQL functions and windowing operations.
* Performing aggregations to understand stock activity over time per ticker.
* Sentiment analysis on tweet content using PySpark, where tweets were classified as positive, negative, or neutral based on keyword heuristics. This enabled sentiment-based correlation with market movements and trading signals.

Justification: Spark was used for its in-memory distributed computing capabilities, which allow for fast processing of large datasets across clusters. Spark's integration with both Cassandra and MongoDB made it a natural choice for centralising processing without data migration.

## Merging Processed Datasets and Final Storage

After performing individual preprocessing steps on both datasets: tweets and stock prices using Apache Spark, the two data sets were joined to create a unified dataset. This merge was based on the stock ticker and the date of the stock market activity aligned with the tweet timestamp truncated to the same day for consistency.

The merging process involved the following Spark operations:

* Timestamp normalisation: Tweet timestamps were converted to date format using Spark SQL functions to enable joining on the same date field with the stock data.
* Inner join between the tweets and stock data based on the ticker and date columns.
* Selective filtering to retain relevant columns such as sentiment score, stock returns, moving averages, and future price indicators.

Storing the Merged Dataset in Cassandra

Once the unified dataset was created, it was stored back to Cassandra for durable and distributed storage. This step used the Spark-Cassandra connector, which allows writing a Spark data frame directly into a Cassandra table with appropriate schema mapping.

Justification:

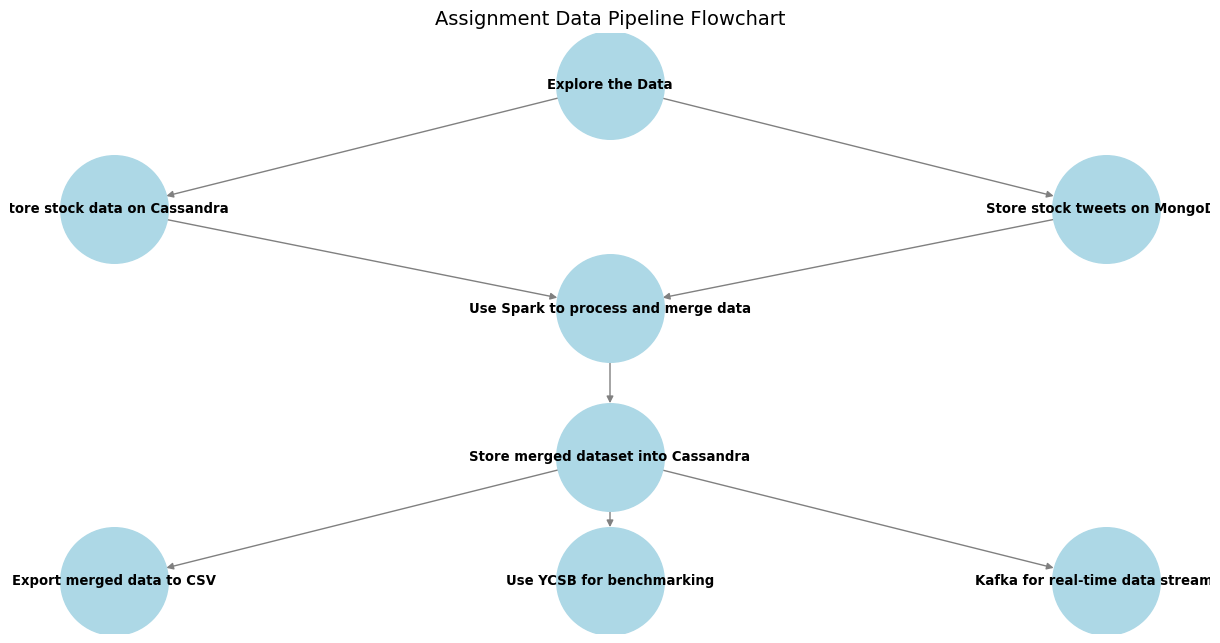
Cassandra was used again at this stage due to its low latency support for both read and write operations, and its ability to efficiently store the resulting time series data by composite keys (ticker, date).

Storing the enriched dataset (with both market and sentiment signals) in Cassandra enables downstream applications such as dashboards, alerts, or model training systems to access cleaned, feature rich data quickly.

## Exporting the Final Dataset to CSV

For advanced analysis using machine learning models built in traditional tools (e.g., Pandas, scikit-learn), the merged dataset was exported as a CSV file using PySpark’s.write.csv() method.

The following flowchart illustrates the sequence of steps followed in the Big Data pipeline for this project, from initial data exploration through processing, storage, export, and benchmarking.



## Comparative Analysis

To benchmark the performance of a relational database (MySQL) against a NoSQL database (MongoDB), the YCSB tool was used with Workload A, which represents a balanced mix of 50% reads and 50% updates. The results indicate that MongoDB significantly outperformed MySQL across all key metrics. MongoDB achieved a throughput of 254.12 operations per second, compared to 73.60 ops/sec for MySQL. Additionally, MongoDB demonstrated lower latencies in both read and update operations, with an average read latency of 853.42 µs versus 1508.91 µs in MySQL, and an average update latency of 6924.21 µs compared to 12865.35 µs. The 95th and 99th percentile latencies for reads were also lower in MongoDB, reflecting more consistent performance under load. These results suggest that MongoDB handles mixed workloads more efficiently, likely due to its document-based architecture and more flexible write model.

# Advanced Data Analytics

## Exploratory Data Analysis (EDA)

The dataset contains 1,235 records and 17 columns related to stock prices, volumes, previous closes, returns, moving averages, and sentiment scores for multiple tickers. Initial preprocessing included converting the date column to datetime format, sorting by ticker and date and handling missing values.

EDA involved:

* Visualising feature correlations via a heatmap to identify relationships between numerical variables.
* Plotting time series of adjusted closing prices for each ticker to observe price trends over time.
* Analysing sentiment score distribution and its evolution over time by ticker.
* Examining the distribution of daily returns overall and by ticker.
* Investigating the relationship between sentiment scores and daily returns using scatterplots and boxplots.
* Comparing closing prices with their 5-day moving averages to assess short-term trends.
* Grouping by sentiment labels (Positive, Neutral, Negative) to analyse their impact on next-day price changes and average returns.

For missing data handling, rows with null values in key lagged prices and daily returns were dropped to maintain data integrity since these features are critical for temporal continuity. Sentiment missing values were imputed with zero, reflecting a neutral sentiment assumption, as missing sentiment may indicate no significant market news or neutral investor mood on those days. This approach avoids biasing the model towards positive or negative sentiment.

## Feature Engineering

Feature engineering plays a pivotal role in enhancing the predictive performance of machine learning models, particularly in time series forecasting tasks such as stock price prediction. In this project, several new features were created based on the original stock market dataset. These features were selected and engineered to provide the model with relevant temporal and statistical information that can improve its ability to identify patterns and trends in stock price movements.

Lag Features

Features: prev\_close\_1, prev\_close\_2, prev\_close\_3

These variables represent the stock’s closing prices on the previous 1, 2, and 3 trading days, respectively.

Rationale: Stock prices exhibit autocorrelation, meaning that past prices can provide useful information about future values. Including lagged prices allows the model to learn from recent historical trends.

Justification: Lag features help capture short term dynamics in the time series and are particularly beneficial for models that do not natively handle temporal sequences

Future Target Variables

Features: close\_t\_plus\_1, close\_t\_plus\_3, close\_t\_plus\_7

These features represent the closing stock prices projected 1, 3 and 7 days into the future. They serve as target variables for predictive models.

Rationale: Creating multiple forecasting horizons allows for tailored predictions to accommodate different types of market participants.

Justification: These target variables enable comparative analysis across different timeframes and provide flexibility in model deployment depending on the required prediction interval.

Moving Average

Feature: ma\_5 (5-day moving average)

This is the mean of the closing prices over the preceding five days.

Rationale: Moving averages are widely used in financial analysis to smooth out short term fluctuations and highlight underlying trends.

Justification: The inclusion of a moving average reduces the impact of daily noise and helps the model better capture sustained directional movements in the stock price.

Daily Return

Feature: daily\_return

Calculated as the percentage change in the closing price compared to the previous day.​

Rationale: Daily return quantifies the relative change in stock value and provides a normalised indicator of market volatility and momentum.

Justification: Return based features are scale invariant, facilitating better model generalisation and comparability across stocks. This feature also assists in identifying momentum-based price movement patterns.

Next Day Return

**next\_day\_return** (absolute price change) and **adj\_next\_day\_return** (price change relative to adjusted close) were created to quantify short term price movement, which aids models in learning return dynamics and volatility.

The newly engineered features were designed to incorporate both short-term historical information and momentum indicators into the dataset. Their selection was grounded in financial domain expertise and supported by exploratory data analysis. These features provide a more informative and structured dataset for model training, thereby improving prediction accuracy and interpretability.

## Processing Tweets and Sentiment Score

To enhance the predictive power of the model by incorporating public perception and investor sentiment, tweets related to five stocks were collected and processed. The sentiment expressed in these tweets was quantified into a numerical score that could be used as a feature in the machine learning models.

**Preprocessing and Cleaning**

The raw tweet text underwent a series of preprocessing steps to ensure the textual data was clean, consistent and free of noise that could degrade sentiment analysis accuracy. The following transformations were applied:

* Lowercasing: All text was converted to lowercase to avoid duplication of semantically identical words (e.g., "Apple" vs. "apple").
* URL Removal: Links were removed as they do not contribute meaningful sentiment information.
* Mentions and Hashtags: User mentions (e.g., @user) and hashtags were stripped to retain only the core message.
* Punctuation and Special Characters: Removed to eliminate unnecessary noise.
* Stopword Removal: Common words with little semantic value (e.g., "is", "the", "at") were excluded.
* Tokenisation: Tweets were split into individual words or tokens to facilitate analysis.

These steps were critical to improving the quality of the input for sentiment analysis, allowing the model to focus on meaningful content.

Sentiment Analysis Technique

The sentiment of each tweet was analysed using VADER (Valence Aware Dictionary for Sentiment Reasoning), a rule-based sentiment analysis tool designed specifically for social media texts. VADER provides four sentiment metrics:

* Positive
* Negative
* Neutral
* Compound (an aggregated score that ranges from -1 [most negative] to +1 [most positive])

The compound score was used as the primary sentiment metric due to its ability to summarise the overall sentiment polarity of the text in a single number. This value was calculated for each individual tweet and retained as the sentiment score.

**Justification for Using VADER**

VADER was chosen due to its suitability for analysing short, informal text such as tweets. Unlike traditional natural language processing tools that require large amounts of training data, VADER is pre-trained and optimised for social media content. Its lexicon includes emoticons, slang and acronyms frequently found in tweets, which enhances accuracy in this domain.

Additionally, VADER’s rule-based approach offers explainability and efficiency, making it appropriate for large scale sentiment scoring without the computational overhead of deep learning models.

**Aggregating Sentiment**

Since stock market trends are typically influenced by daily or intraday sentiment, the individual sentiment scores were aggregated at the daily level. The mean compound sentiment score per day was calculated to provide a representative value of the market sentiment for each trading day. This value was then merged with the stock price dataset using the date as a common key, creating a new feature: sentiment.

## Analysis and Forecast of Adjusted Close Prices Using Time-Series Models

**Stationarity Testing and Data Preparation**

To begin the analysis, the stationarity of the adjusted close price series for the companies was assessed using the Augmented Dickey-Fuller (ADF) test. The initial tests showed high p-values (e.g., 0.98 for the first company), indicating the series were non-stationary. This means the price data had trends or other non-stationary behaviour, which is unsuitable for direct time-series modelling.

To address this, first-order differencing was applied to the adjusted close price data. After differencing, the ADF tests returned very low p-values (e.g., 4.1e-29), confirming that the differenced series were now stationary and appropriate for modelling with autoregressive techniques.

**SARIMAX Modelling with Sentiment as Exogenous Variable**

Given the availability of sentiment scores derived from tweets, these were incorporated as an exogenous variable in the SARIMAX models to potentially improve forecast accuracy.

A SARIMAX (1,0,1) model was fit to the differenced adjusted close prices, with sentiment as an explanatory variable. Model fitting was performed on 80% of the historical data, reserving the remaining 20% for testing.

**Forecasting Results**

Forecasts for adjusted close prices were produced for 1 day, 3 days, and 7 days into the future by cumulatively summing the predicted differences from the SARIMAX model and adding these to the last known price.

* **Company 1** forecasted adjusted close prices increased gradually from approximately €219 on Day 1 to €226 on Day 7.
* **Amazon (AMZN)** prices showed a near flat forecast around €160 across the 7-day horizon.
* **Microsoft (MSFT)** displayed slight increases from €215.48 on Day 1 to €215.85 on Day 7.
* **Nvidia (NVDA)** and other companies similarly exhibited modest trends in their forecasts.

**Model Performance Evaluation**

Model accuracy was assessed using the Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) on the test data. Performance varied across companies:

* For **Microsoft (MSFT)**, the SARIMAX model performed well, achieving a low MAE of 5.66 and RMSE of 7.02, indicating reliable forecasts.
* Conversely, some companies exhibited higher error metrics (e.g., MAE around 24.5), suggesting that further model tuning or inclusion of additional variables might be necessary.

**Interpretation and Implications**

The SARIMAX approach incorporating tweet sentiment proved useful in capturing short-term price movements, as evidenced by reasonable forecast trends and acceptable accuracy for several companies. The use of differencing effectively handled the non-stationary nature of the price series, ensuring model assumptions were met.

The relatively stable forecasted prices for companies like Amazon suggest market expectations of little change in the near term, whereas more dynamic forecasts for others reflect greater market sensitivity.

LSTM

The forecasted prices for each stock were generated for three future time horizons: Day 1, Day 3, and Day 7. For Apple (AAPL), the forecasted prices were €126.26 on Day 1, €125.39 on Day 3, and €124.37 on Day 7. Amazon (AMZN) was projected to be priced at €157.56, €157.04, and €158.23 over the same respective periods. Microsoft (MSFT) showed a slight downward trend with forecasts of €219.99, €219.57, and €218.91. Nvidia (NVDA) forecasts indicated more fluctuation, with prices of €144.70, €143.28, and €146.13. Tesla (TSLA) followed a similar volatile pattern, with forecasted prices of €215.65, €212.86, and €219.43 across Day 1, Day 3, and Day 7, respectively.

To evaluate the performance of the forecasting models, two metrics were employed: Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). These metrics were calculated for each stock across all forecast horizons. Apple (AAPL) exhibited relatively low error rates with RMSE values of 4.51, 4.87, and 5.58, and corresponding MAE values of 3.55, 3.78, and 4.31 for Days 1, 3, and 7, respectively. Amazon (AMZN) showed a similar trend with RMSEs of 4.41, 4.56, and 5.66, and MAEs of 3.72, 3.63, and 4.33. Microsoft (MSFT) had slightly higher errors, with RMSEs of 6.97, 6.63, and 6.16, and MAEs of 5.52, 5.05, and 4.76.

In contrast, Nvidia (NVDA) and Tesla (TSLA) presented significantly higher error values, indicating potential challenges in accurately modelling their price movements. Nvidia's RMSE values were 17.07, 14.58, and 17.92, while its MAE values were 16.18, 13.77, and 17.38. Tesla showed the highest errors, with RMSEs of 19.84, 23.07, and 28.67, and MAEs of 15.53, 18.78, and 24.74.

Overall, the forecasting models performed best for AAPL and AMZN, suggesting stable price behaviour and strong model fit. Conversely, the higher errors for NVDA and TSLA point to increased price volatility or complexity that may require more advanced modelling techniques or the inclusion of additional explanatory variables.

## Tufte’s Principles in the Dashboard

The interactive dashboard created for forecasting adjusted close prices aligns well with Edward Tufte’s principles, ensuring the data is presented clearly and effectively. It keeps the focus on the important information by reducing unnecessary ink and avoiding clutter like excessive gridlines or flashy decorations. Using separate plots for each ticker (small multiples) makes it easy to compare without crowding the visuals, helping users spot trends quickly. Labels, legends, and consistent colours are clear and straightforward, guiding the viewer without confusion. The actual and forecasted data are distinguished by different line styles and colours, making the story easy to follow. The interactive elements allow users to explore the data in detail, putting the emphasis firmly on the numbers rather than distractions. Overall, the dashboard keeps things simple and clear, sticking closely to Tufte’s ideas on good data presentation.