# MSc in Data Analytics

Author: Claire O’Brien

e-mail: sbs24031@student.cct.ie

Student ID: sbs24031

GitHub links:

https://github.com/CCT-Dublin/integrated-ca2-claireobrien00.git

https://github.com/claireobrien00/Sem-2-CA2-Dashboard.git

## Abstract

*This report investigated datasets composed of temporal data about companies’ stock prices for 2020. An additional dataset containing Twitter data about the same companies was analysed. LSTM and Autoregressive analyses were carried out on each chosen company. A minimum LSTM loss of 0.0002 was recorded. The lowest mean squared error in the autoregressive section was 0.05. This work aimed to predict a company’s closing stock price and to examine how Big Data storage and processing technologies could be used to optimise the analysis. MySQL and Apache Cassandra were used for data storage. A comparative analysis using YCSB between MySQL and MongoDB highlighted the superior performance of MongoDB.*

## Introduction

This work aimed to predict the closing price of various companies’ stock value at the end of each day. Data from the stock market across 2020 along with tweets pertaining to the various companies was supplied and used for the analysis. This is a difficult problem as there is a high degree of variability within the stock market for numerous external reasons. This makes it extremely difficult to model. The literature was analysed and research was conducted on the optimal ways to store data and predict the closing price (Vijh et al., 2020).

## Advanced-Data Analytics

### EDA

#### Temporal Properties

Time series data requires some of the same and some unique exploratory techniques. The CSV files were loaded into a Jupyter Notebook, and EDA was performed. Various Python libraries contain useful resources for analysing time series data. It was checked that there were no null values in any of the datasets. Each company’s stock prices were visualised to understand how they performed across the year. It was clear immediately that the close, open, high and low prices followed the same pattern across the year. Boxplots showed the spread of the data across each month of the year. Taking AAPL’s data as an example,

the stock price peaked in September and hit its bottom in March/April. The largest price variance within a month occurred in August (see lines 58 and 59).

The stock market closes on Saturdays and Sundays. Time series data must be recorded at regular time intervals, daily in this case, and so a technique was employed to fill in the missing weekend values (Box, 2016). Multiple methods were tested and the KNN Mean method was chosen for all the datasets. This function took the 8 nearest values of the NaN row and replaced NaN with the average of the 8 nearest neighbours. This worked well for all the datasets, the actual and KNN Mean datasets followed a similar path. It is a better representation of the missing values than a simple forward or backfill as it combines multiple nearby prices, 8 in this case.

The datasets were next plotted using the multiplicative and additive decomposition plots, auto and partial correlation plots, lag plots and the detrended plot. The ADF and KPSS tests were applied to determine if the series’ were stationary. The approximate and stationary entropies were calculated, these values indicated how well the target variable could be predicted using the date. The Granger Causality test quantified the causal relationship between the closing price and the other variables. Table 1 shows a summary of the results of these experiments and the actions taken based on their results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Seasonality | Stationary | Trend | Autocorrelation | Forecastability | Granger Causality |
| AAPL  Action taken:  Detrended and  Made Stationary | Additive Decomposition:  Trend present  No seasonality  Multiplicative Decomposition:  Trend present  No seasonality  Auto/Partial  correlation plot:  No peaks  Conclusion:  Not seasonal | ADF and KPSS Test:  Not stationary  First difference used  Stationary | Trend present:  Positive trend in the price from January to  December  Detrended | ACF and PACF plots:  Linear relationship  Lag Plot:  Positive linear relationship between Closing price and time | Approximate Entropy:  0.22  Sample Entropy: 0.17  Forecastable | Variables with the greatest causality:  High  Low  Compound score |
| AMZN  Action taken:  Detrended and made stationary | Additive Decomposition:  Trend present  No seasonality  Multiplicative Decomposition:  Trend present  Weak pattern in the additive residual plot  Auto/Partial  correlation plot:  No peaks  Conclusion:  Not seasonal | ADF and KPSS Test:  Not stationary  First difference used  Stationary | Trend present:  Positive trend in the price from January to  December  Detrended | ACF and PACF plots:  Linear relationship  Lag Plot:  Positive linear relationship between Closing price and time  As lag increases there is a small amount of separation, this indicates a smaller lag would be optimal | Approximate Entropy:  0.24  Sample Entropy: 0.19  Forecastable, less than AAPL | Open has the greatest causal relationship |
| BA  Action taken:  Detrended and made stationary | Additive Decomposition:  No  trend present  No seasonality  Multiplicative Decomposition:  No Trend present  No seasonality  Auto/Partial  correlation plot:  Some peaks are present  Conclusion:  Seasonal | ADF and KPSS Test:  Not stationary  Second derivative and log used | No Trend: Price varies across the year but not in a linear way | ACF and PACF plots:  Peaks and troughs present, seasonality indicated  Lag Plot:  Dispersion increases at early time steps and high lag values | Approximate Entropy:  0.3  Sample Entropy: 0.21  Forecastable, less than AAPL and AMZN | Open, High, Low and Adj\_Close all have a causal relationship to the Closing price |
| DIS  Action taken:  Deseasonalised and made stationary | Additive Decomposition:  No Trend  Seasonality  Multiplicative Decomposition:  No Trend  Seasonality  Auto/Partial  correlation plot:  Some peaks are present  Conclusion:  Seasonal | ADF and KPSS Test:  Second derivative taken to make stationary | No Trend | ACF and PACF plots:  Peaks and troughs present, seasonality indicated  Dispersion increases at early time steps and high lag values | Approximate Entropy:  0.38  Sample Entropy:  0.29  Forecastable | Open and High have a causal relationship to the Closing price |
| TSLA  Action taken:  Detrended and made stationary | Additive Decomposition:  Trend present  No seasonality  Multiplicative Decomposition:  Trend present  Weak pattern in the additive residual plot  Auto/Partial  correlation plot:  No peaks, not seasonal | ADF and KPSS Test:  Derivative and log taken | Positive trend present, detrended | ACF and PACF plots:  Linear relationship  Lag Plot:  Positive linear relationship between Closing price and time  As lag increases there is a very little separation | Approximate  Entropy:  0.41  Sample Entropy:  0.28  Forecastable | Open and Low have a causal relationship to the Closing price |

#### Table 01: A summary of the characteristics and actions of the temporal dataset taken

#### Language Processing

A tweets dataset was provided. Each row contained a tweet regarding one of the companies, along with a date. This data was an exogenous variable and was implemented in the Machine Learning Algorithms to provide insight into the closing stock prices of the companies.

Based on this dataset, the companies to be analysed were chosen. The five companies with the most tweets were selected.

The Natural Language Tool Kit (NLTK) was employed to perform the text analysis portion. The text data was filtered and saved to separate data frames based on the ticker column. A function, see line 12, was written to remove unnecessary stop words and punctuation and transform all words into lowercase. Chai et al. comment on removing punctuation, which can impact sentiment analysis. Certain punctuation, such as exclamation marks, can heighten emotions such as excitement or anger. It can also change the meaning of a sentence, for example, a comma in a phrase can change the information being conveyed. Due to the nature of tweets, where incorrect or inconsistent punctuation is often used, the decision was made to run the analysis without punctuation initially. It was subsequently run with punctuation, and there was no discernible difference in the overall sentiment so it was removed for simplicity (Chai, 2023).

Stemming and lemmatisation were experimented with, but the sentiment analyser was found to deal well this the unshortened text. Meaning can also be lost during these processes so it was decided not to use them.

An instance of SentimentIntensityAnalyzer() was created. This is a VADER object available in the NLTK library and has been shown to perform well on social media data. It was chosen due to its ease of implementation and its ability to include emojis. An informal review of the Twitter data revealed a high level of emojis present which are a key part of social media data and include valuable insights into the sentiment of a tweet (Bonta et al., 2019).

The instance returns a positive, negative, neutral and compound sentiment score. To use as an exogenous variable, a single value was needed. The compound score was extracted and saved. If there were multiple tweets in a day, the average score was found. If there was no tweet for a company on a given day, a score of 0 was returned. This value indicated neutrality.

#### Machine Learning Algorithms

### *Autoregressive Models*

A variety of autoregressive models were applied and tweaked to determine which model best suited each company’s dataset. Converting the datasets to stationary time series was crucial for this part of the analysis. Autoregressive models require the data to be stationary to provide an accurate prediction (Box, 2016).

A standard autoregressive forecaster with a random forest regressor was chosen. Other regressors were experimented with such as Support Vector Machines, KNN and Linear regression but random forest gave the best accuracy overall. This forecaster was run with and without the exogenous variables and compared. Exogenous variables are additional parameters collected at the same time stamps as the target variable and can improve the model’s predictive capabilities.

ARMA, ARIMA, SARIMA and SARIMAX models were all implemented and compared as well. Some of these included the exogenous variables and some just the target variable.

The root mean squared error was used as the performance metric.

### *Neural Network Models*

A neural network was composed for each of the five companies. Long Short-Term Memory (LSTM) models are a specific type of Recurrent NN and are well suited to time series problems; this is due to their inclusion of the previous values in predicting the subsequent one. LSTMs have successfully been applied to financial data and found to outperform more traditional autoregressive techniques (Siami-Namini et al., 2018). Bennett et al. compared an ARIMAX and Neural Network (NN) to predict voltage usage and found the NN performed marginally better (Bennett et al., 2014).

The data was scaled using either a standard scaler, for approximately normally distributed data or a min-max scaler for skewed data. The spread of the data, using a pair plot, was checked before and after the scaling to ensure the shape was maintained. The type of scaling had a big impact on how closely the model predicted the training data.

The exogenous variables were added to the model. As this was a regressive problem, the loss function was the mean squared error. Time steps of 30 were used to represent approximately a month. Three layers of 100 neurons, a dense layer of 100 neurons and an output layer of one neuron made up the architecture of the LSTM. The optimiser Adam was chosen as it provided the lowest loss. Dropout of 20% was added to prevent overfitting and was found to improve the model’s results.

The LSTM model was run using the deseasonalised/detrended/stationary datasets and then the unchanged values. The accuracy metric was found to be higher with the unchanged values. The LSTM was much better at detecting the patterns in the actual closing price.

The number of epochs was chosen so that all the models’ loss functions converged to a stable value, a local minimum, and were not decreasing. The batch size was not experimented with due to time constraints but this could be investigated to improve the results. Relu and Leaky Relu activation functions were tried but found to perform worse than the default LSTM activation function Tanh.

#### Hyperparameter Tuning

A grid search forecaster was used for the Autoregressive models. This type of hyperparameter tuning runs through all the combinations of a defined grid of hyperparameters and returns the combination which produces the optimal metric, the lowest mean squared error in this case. Some of the hyperparameters considered were the lag of the forecaster, the max depth and the number of estimators of the regressor. The optimal combination of hyperparameters was then used in the forecasters.

A gradient booster machine (GBM) was also added. A regressive version was created for this problem. It is a tree-based optimiser which trims the branches that do not improve the model and focuses on the ones that do. Agapitos et al. modelled financial time series data using regularised gradient boosting, a version of GMB, and found it to perform well compared to other machine learning techniques (Agapitos et al., 2017). For this reason, it was explored in this work.

#### Results and Discussion

### *Autoregressive Models*

### The filled, detrended/deseasonalised and stationary datasets were used in the autoregressive models. The biggest improvement for the autoregressive models was observed when the exogenous variables were added. The seasonality parameter values were altered to find the best combination.

A graph with blue lines

Description automatically generatedAll of the companies’ results were visualised in the accompanying notebook. A selection is discussed here. AMZN had a mean squared error (MSE) of 0.048 and BA 15.82 when using the same autoregressive forecaster with a random forest regressor. This shows the variability between two datasets using the same model. The results for these two companies are displayed in Figures 01 and 02.

Figure 01: BA Autoregression Forecaster Testing vs Predictions plot

A graph with blue lines

Description automatically generated

Figure 02: AMZN Autoregression Forecaster Testing vs Predictions plot

A graph showing a line graph

Description automatically generatedFor the SARIMAX models, the DIS dataset had a root MSE of 0.0007. This is comparable to the TSLA RMSE which came in at 0.0002. These low errors indicate that the models’ predictive capacities were very strong. The results of these models can be seen in Figures 03 and 04.

Figure 03: DIS SARIMAX results

A graph showing a number of data

Description automatically generated

Figure 04: TSLA SARIMAX results

### *LSTM*

5 LSTM models were successfully run, one for each company. The BA dataset had the best model when compared to the training data. The close prediction and testing variable can be seen in Figure 04.

A graph showing a line graph

Description automatically generated

Figure 04: BA LSTM Neural Network prediction

A graph showing a line graph

Description automatically generated

Figure 05: AAPL LSTM Neural Network prediction

AAPL’s model also performed well. It had a minimum loss of 0.001. This model could be further improved by adding more LSTM layers.

The cumulated results of the models for each company for days 1, 3 and 7 can be seen in Tables 02-06.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Days | ForecasterAutoreg  (with exogenous variables) | | SARIMAX | | LSTM  Minimum Loss-0.001 | |
|  | Predicted | Actual | Predicted | Actual | Predicted | Actual |
| 1 | -0.89 | -0.88 | -0.52 | -0.51 | 102.65 | 117.5 |
| 3 | -0.94 | -0.95 | 1.41 | 1.43 | 109.68 | 115.04 |
| 7 | 1.29 | 1.3 | -3.04 | -3.14 | 123.11 | 116.6 |

Table 02: Predictions and Actual Data for AAPL.

Note the autoregressive models were not reverted to the original price scale due to time constraints.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Days | ForecasterAutoreg  (with exogenous variables) | | SARIMAX | | LSTM  Minimum Loss-0.0004 | |
|  | Predicted | Actual | Predicted | Actual | Predicted | Actual |
| 1 | 0.96 | 0.92 | -6.89 | -6.89 | 125.03 | 160.85 |
| 3 | -0.37 | -0.41 | -10.82 | -10.8 | 160.67 | 158.82 |
| 7 | 2.76 | 2.73 | -5 | -5 | 164.87 | 164.31 |

Table 03: Predictions and Actual Data for AMZN

Note the autoregressive models were not reverted to the original price scale due to time constraints.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Days | ForecasterAutoreg  (with exogenous variables) | | SARIMAX | | LSTM  Minimum Loss-0.0008 | |
|  | Predicted | Actual | Predicted | Actual | Predicted | Actual |
| 1 | 148.33 | 148.28 | 148.95 | 148.96 | 167.42 | 172.49 |
| 3 | 145.47 | 145.17 | 153.31 | 153.31 | 169.07 | 170.63 |
| 7 | 153.31 | 153.31 | 164.32 | 164.34 | 155.24 | 169.89 |

Table 04: Predictions and Actual Data for BA.

Note the autoregressive models were not reverted to the original price scale due to time constraints.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Days | ForecasterAutoreg  (with exogenous variables) | | SARIMAX | | LSTM  Minimum Loss-0.0004 | |
|  | Predicted | Actual | Predicted | Actual | Predicted | Actual |
| 1 | 124.6 | 124.7 | 121.70 | 121.70 | 124.95 | 125.37 |
| 3 | 127.66 | 127.41 | 123.78 | 123.78 | 127.56 | 125.42 |
| 7 | 123.05 | 123.07 | 131.35 | 131.35 | 123.31 | 124.58 |

Table 05: Predictions and Actual Data for DIS.

Note the autoregressive models were not reverted to the original price scale due to time constraints.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Days | ForecasterAutoreg  (with exogenous variables) | | SARIMAX | | LSTM  Minimum Loss-0.0002 | |
|  | Predicted | Actual | Predicted | Actual | Predicted | Actual |
| 1 | -0.52 | -0.53 | -0.56 | -0.56 | 140.65 | 147.30 |
| 3 | -0.52 | -0.53 | 1.49 | 1.49 | 141.93 | 146.06 |
| 7 | -0.17 | -0.16 | -0.57 | -0.57 | 141.56 | 148.09 |

Table 06: Predictions and Actual Data for TSLA.

Note the autoregressive models were not reverted to the original price scale due to time constraints.

### *Hyperparameter Tuning*

The hyperparameter tuning offered some interesting insights. It produces a plot of the relative importance of each feature. It determines a score of how important each feature is to the target variable, the closing price, see like 417 for an example.

Each company had an optimal regressor tuned. A variety of error measurements were used to quantify the performance of the tuned models.

The results are shown in Table 07. This shows the low MAE values calculated for each company. This indicates this is a useful way to maximise the results of a model.

|  |  |
| --- | --- |
| Company | Tuned models’ Mean Absolute Error |
| AAPL | 0.19 |
| AMZN | 0.21 |
| BA | 4.83 |
| DIS | 2.73 |
| TSLA | 0.1 |

Table 07: MAE for each company’s LightGBM Regression Model

#### Dashboard

A dynamic dashboard was set up using Streamlit. The stock price CSVs were uploaded and visualised according to Tufts principles. Tufts principles offer a guideline for visualising data with integrity and clarity. A graph was created for each company as the prices spanned different magnitudes. Putting all the data on one graph would have been unclear. The graphs accurately represent the available data. A final graph was added for comparison, which contained only the closing price for each company to show a clear and efficient comparison of the companies. The graphs are unobtrusive and aim to maximise data density. There is no unnecessary labelling or additional text. To make it interactive, a button was added to the AAPL plot, allowing you to choose which price you would like to see. A button was added to the AMZN graph, which allowed a month to be chosen, and then the prices for that month were displayed (Tufte, 2001).

The autoregressive forecaster from the SARIMAX model was exported and saved to a .joblib file. It was attempted to add this to the dashboard where the user could input a date and the model would calculate the predicted closing price. An issue arose with the Sklearn library not being available in the virtual environment. Due to time constraints, this issue was not resolved. The code is included in the Sem\_2\_CA2\_Dashboard.py file but is commented out.

## Big Data Storage and Processing

### Data Preparation and Storage

The first task was to deal with the raw data. Using an Ubuntu VM, the selected companies’ CSV files, along with the Twitter CSV file, were added to the HDFS using Hadoop. It was attempted to implement MapReduce on the Twitter data using a PySpark notebook. The attempt is included in the submission. An issue arose with the RDD pipeline and so another approach was considered.

The CSV data was handled using MySQL, a relational database. A new database was created for the report along with a table for each of the five companies. The CSV files were added to the local SQL folder and read into the MySQL database. A new notebook was created and the MySQL connector was deployed. A connection was secured with the database and each of the five stock price tables was uploaded. MySQL was chosen due to its compatibility with CSV files. It works well with structured, finite data such as the stock price data.

Apache Cassandra was used for the tweet data. The CSV was loaded into a table in a custom-made keyspace using the CQLSH shell. Again, a Pyspark notebook was used to perform queries on the file once a connection had been established. Cassandra acted as the storage system for the Twitter data. Apache Cassandra’s architecture is made up of decentralised clusters and in the project, Pyspark acted as the cluster manager. Queries were carried out on the dataset in the PySpark notebook (Salloum et al., 2016). Cassandra is a column-family database which is better equip to deal with the challenges of big data such as flexibility and scalability. The twitter data was less structed, it contained punctuation and emojis between the partitioning commas and Cassandra was able to deal well with this challenge.

### Comparative Analysis using YCSB

YCSB allows database management systems to be compared quantitatively. It does so based on workloads that place differing emphasis on read and write capabilities. Depending on the use case different characteristics are needed. Choosing a database that is well-suited to the application can reduce timely queries and improve overall efficiency, particularly for large data sets for which the cloud data servers are built (Cooper et al., 2010).

YSCB was set up and run on the Ubuntu VM. MySQL, MongoDB and Apache Cassandra were planned to be compared. A user table keyspace was set up in CQLSH but an error arose when running YCSB so MySQL and MongoDB are to be compared. The standard A-F workloads were run. All the workloads kept the record count at 1000.

|  |  |  |  |
| --- | --- | --- | --- |
| Workload | Emphasis | MySQL  Runtime (ms) | MongoDB  Runtime (ms) |
| A | Read 50%  Update 50% | 6387 | 1499 |
| B | Read 95%  Update 0.05% | 3349 | 1684 |
| C | Read 100%  Update 0% | 3667 | 1635 |
| D | Read 95%  Insert 5% | 5486 | 1361 |
| E | Scan 95%  Insert 5% | 3871 | 1166 |
| F | Read 50%  Read, modify, write, proportion 50% (“YCSB/workloads at master · brianfrankcooper/YCSB,” n.d.) | 4368 | 1329 |

Table 2: Results of YCSB comparing MongoDB and MySQL databases

The two databases differ in that MongoDB is a NoSQL database, and MySQL is an SQL database. These two databases store information differently, MySQL is ordered in columns and rows and NoSQL is in a document format. MongoDB has more flexibility and scalability and is better suited to less structured data such as audio or video data. The two systems’ architectures are very different, and so it follows that they should perform differently for the same workloads. Pandey et al. analysed MongoDB to have lower run times over a range of record values for workloads A, C and F. (Pandey, 2020). The results are displayed in Table 2. MongoDB had a lower run time for every workload, which is consistent with what was found in the literature. This comparison was carried out after the bulk of the report had been carried out meaning MySQL had already been implemented successfully. It is recommended, based off this analysis to utilise YCSB prior to choosing a database. Customising a workload is straightforward, estimate the operations, read, insert, update, which is to be most used for a project and estimate roughly the number of records which are in the dataset to be analysed and run YCSB. It is simple and fast to use with easily interpretable results. Kusar et al. found MongoDB to outperform Apache Cassandra, indicating it is a powerful, efficient database storage and processing option (Abu Kausar et al., 2022).

### Big Data Architecture

Multiple Big Data platforms for storage and processing were investigated in this module. The steps carried out for this report are outlined in the flow diagram, Fig. 06.

MapReduce was attempted on PySpark. This was favoured over Hadoop MapReduce as it has been shown in the literature to be faster and easier to implement. However, Hadoop returns better classification accuracies than Spark. There is a trade-off between ease and accuracy. For this project, Spark was chosen as it was better understood (Tekdogan and Cakmak, 2021). The tweets were successfully mapped but an error occurred during the reduce function. MapReduce offers the advantage of processing a large dataset across multiple distributed computing systems and then consolidating the results. This benefits large datasets where processing on a single system could be slow and inefficient.

Apache Cassandra and MySQL were chosen to demonstrate an SQL and NoSQL database. They are both widely used, open-source storage systems that are reliable and scalable, Cassandra more so. One reason for their implementation was both have command line shell tools, mysql and cqlsh, which make creating databases/keyspaces and tables simple.

## Figure 06: Flow diagram of the Big Data technologies used

## Conclusions

This work successfully applied MySQL and Apache Cassandra and uploaded it to PySpark using the suitable connector on an Ubuntu VM. From there, the datasets were added to a Jupyter notebook. The datasets were explored and adjusted as needed. A separate Twitter dataset was analysed using VADER Sentiment analysis. This sentiment score along with the opening, high, low, adjusted close prices and volume of various stocks were used to predict the closing price using = autoregressive models and LSTM RNNs. Light GBM and GridSearchCV hyperparameter tuning was used on each company to optimise the models.

YCSB was run to compare MongoDB and MySQL, highlighting MongoDB’a, and NoSQL in general, superior performance.

## References

Abu Kausar, M., Nasar, M., Soosaimanickam, A., 2022. INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY A Study of Performance and Comparison of NoSQL Databases: MongoDB, Cassandra, and Redis Using YCSB. Indian Journal of Science and Technology 15, 1532–1540. https://doi.org/10.17485/IJST/v15i31.1352

Agapitos, A., Brabazon, A., O’Neill, M., 2017. Regularised gradient boosting for financial time-series modelling. Comput Manag Sci 14, 367–391. https://doi.org/10.1007/s10287-017-0280-y

Bennett, C., Stewart, R., Lu, J., 2014. Autoregressive with Exogenous Variables and Neural Network Short-Term Load Forecast Models for Residential Low Voltage Distribution Networks. Energies 7, 2938–2960. https://doi.org/10.3390/en7052938

Bonta, V., Kumaresh, N., Janardhan, N., 2019. A Comprehensive Study on Lexicon Based Approaches for Sentiment Analysis. Asian Journal of Computer Science and Technology 8, 1–6. https://doi.org/10.51983/ajcst-2019.8.S2.2037

Box, G.E.P., 2016. Time Series Analysis. Wiley.

Chai, C.P., 2023. Comparison of text preprocessing methods. Nat. Lang. Eng. 29, 509–553. https://doi.org/10.1017/S1351324922000213

Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R., 2010. Benchmarking cloud serving systems with YCSB, in: Proceedings of the 1st ACM Symposium on Cloud Computing. Presented at the SOCC ’10: ACM SIGMOD-SIGOPS Workshop on Cloud Computing in conjunction with SIGMOD 2010, ACM, Indianapolis Indiana USA, pp. 143–154. https://doi.org/10.1145/1807128.1807152

Pandey, R., 2020. Performance Benchmarking and Comparison of Cloud-Based Databases MongoDB (NoSQL) Vs MySQL (Relational) using YCSB. https://doi.org/10.13140/RG.2.2.10789.32484

Salloum, S., Dautov, R., Chen, X., Peng, P.X., Huang, J.Z., 2016. Big data analytics on Apache Spark. Int J Data Sci Anal 1, 145–164. https://doi.org/10.1007/s41060-016-0027-9

Siami-Namini, S., Tavakoli, N., Siami Namin, A., 2018. A Comparison of ARIMA and LSTM in Forecasting Time Series, in: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). Presented at the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, Orlando, FL, pp. 1394–1401. https://doi.org/10.1109/ICMLA.2018.00227

Tekdogan, T., Cakmak, A., 2021. Benchmarking Apache Spark and Hadoop MapReduce on Big Data Classification, in: 2021 5th International Conference on Cloud and Big Data Computing (ICCBDC). Presented at the ICCBDC 2021: 2021 5th International Conference on Cloud and Big Data Computing, ACM, Liverpool United Kingdom, pp. 15–20. https://doi.org/10.1145/3481646.3481649

Tufte, E.R., 2001. The visual display of quantitative information. Graphics Press.

Vijh, M., Chandola, D., Tikkiwal, V.A., Kumar, A., 2020. Stock Closing Price Prediction using Machine Learning Techniques. Procedia Computer Science 167, 599–606. https://doi.org/10.1016/j.procs.2020.03.326

YCSB/workloads at master · brianfrankcooper/YCSB [WWW Document], n.d. URL https://github.com/brianfrankcooper/YCSB/tree/master/workloads (accessed 11.7.24).