**MSc in Data Analytics – Integrated Continuous Assessment**

Valentina Martucci SBS23006

[sbs23006@student.cct.ie](mailto:sbs23006@student.cct.ie)

November 10th, 2023

Table of Contents

[**MSc in Data Analytics – Integrated Continuous Assessment** 1](#_Toc150367705)

[1. Introduction 3](#_Toc150367706)

[2. Data processing and storage 3](#_Toc150367707)

[2.1 Methods and tools 3](#_Toc150367708)

[2.2 Data storage and processing 4](#_Toc150367709)

[2.2.1 Database selection 4](#_Toc150367710)

[2.2.2 Data processing 6](#_Toc150367711)

[2.2.3 Data extraction tool evaluation 6](#_Toc150367712)

[3 Time series Forecasting 7](#_Toc150367713)

[3.1 Analysis 7](#_Toc150367714)

[3. 1. 1. Dealing with missing values 8](#_Toc150367715)

[3.2 Modeling 9](#_Toc150367716)

[3.2.1 Model building and selection 10](#_Toc150367717)

[3.3 Dashboard 12](#_Toc150367718)

[4 Conclusion 14](#_Toc150367719)

[References 15](#_Toc150367720)

# Introduction

In the era of digital transformation, the ability to process and analyze large volumes of data, or “Big Data”, has become a critical success factor for businesses across various sectors. Through the application of advanced data processing techniques and forecasting models, this project strives to provide a comprehensive solution for handling Big Data and conducting robust time series analysis.

This document is structured as follows. Section 2 describes the tools used during this project in relation to data management. Section 3 focuses more on the actual data, its analysis and forecasting models. Conclusions for this work are then summarised in Section 4.

# Data processing and storage

## 2.1 Methods and tools

A dataset containing 1.6M observations has been provided for this project. The initial step has been choosing an appropriate database to store the data, process its content, and then store the processed data back in the database for further analysis and modelling. The outcome of the big data processing, along with modelling techniques, has been then summarised in an interactive dashboard. The diagram below represents the data phases, along with the tools used. Additional details on those steps are provided in the next section.

A diagram of a software company

Description automatically generated

Figure - Project Structure

Three Jupyter notebooks have been produced for this study and, as this is a data analysis project, modules like *pandas*, *matplotlib*, *sns* and other have been used for data manipulation and visualization.

Report’s notebooks, along with datasets and report can be found at

<https://github.com/sbs23006/MsC_BigData_AdvAnalytics/>

The report wordcount (including titles and all sections excluding References) is 3011.

## Data storage and processing

The initial step has been the evaluation of one SQL database (MySQL) and one noSQL (HBase). As the number of records is quite high, some benchmark tests have been run against those two databases to evaluate the best performance for this dataset. *Yahoo! Cloud Servicing Benchmark (YCSB)* framework has been used, being a widely recognized tool for evaluating the performances of different database management systems, including both SQL and NoSQL databases (Benchant.com, 2022).

### Database selection

Tests with different workload have been performed on both databases to evaluate their performance sensitivity to data size change. YCSB metrics have been collected for each and compared.

The first metric that has been analysed is runtime, that indicates the total execution time for the test operations. It was observed that MySQL runtime grows significantly as the size of the test data increases, making HBase more suitable for big data storage.

A graph showing a number of numbers

Description automatically generated with medium confidence

Figure - Runtime comparison between MySQL and HBase

The second metric compared has been throughput, that indicates the measure of how many units of information a system can process in a given amount of time. Also in this case, HBase results in being the best choice as it can process more units than MySQL per second.

A graph with a line and a red line

Description automatically generated

Figure - Throughput comparison between HBase and MySQL

Another important metric when looking at databases performances is latency, that is the total amount of time that it will take for the database to receive a request, process the underlying transaction, and return the correct response. Once again, HBase results in being quicker that MyQSL when inserting a bigger amount of data.

A graph showing the difference between latency and num operations

Description automatically generated

Figure - Latency comparison between HBase and MySQL

A NoSQL database is clearly the best choice for this project. Besides HBase, MongoDB has been also considered but, despite the great support from the open-source community, it required more configurations to be able to work with tools like Spark or pymongo to process the data in a Jupyter notebook. HBase has therefore been chosen due to its native integration with Apache Hadoop. Evidence of all those evaluations can be found in the accompanying Jupyter notebook *DB benchmark + Evaluation.ipynb*, as well as within the Big Data section in the file *DataProcessing.ipynb*.

### Data processing

The initial step has been preparing the initial csv file to be loaded into HBase using the built-in ImportTsv utility: this activity mainly consisted of creating a header first, and removing commas from the tweets text to avoid misinterpretation while loading the file into the database. HBase is an open-source, NoSQL, distributed big data store, it integrates seamlessly with Apache Hadoop and the Hadoop ecosystem and runs on top of the Hadoop Distributed File System (HDFS) (Amazon Web Services, Inc.), and for this reason it was very easy to store the csv in the database and then pre-process it with MapReduce jobs.

Due to the size of the data, MapReduce has been used to split the dataset into independent chunks which have then been processed by the map tasks in a completely parallel manner (hadoop.apache.org).

Most of the jobs consist of gaining insights from the data to be then displayed over a dashboard, while others are focused on text manipulation and sentiment extraction. Below is provided a small summary of those jobs, additional details can be found in the *hadoop\_jobs* project folder.

MapReduce jobs:

* Get the distribution of unique flags.
* Calculate how many tweets exist per date.
* Calculate distribution of tweets across moments of the day (morning, afternoon, evening, night)
* Get the number of tweets per user.
* Filter the data by removing rows with empty values and not needed columns, as well as do text manipulation like removing punctuation and stopwords.
* A job has been created to test writing on HBase through Python libraries.
* Calculate the tweets sentiment using *vaderSentiment*

The output of this last job has been then stored in a new table in the database for further work in a Jupyter notebook.



Figure - HBase tables: original and processed tweets + ycsb test

### 2.2.3 Data extraction tool evaluation

The second part of the project consists of using this processed data to analysis and modeling techniques in a Jupyter notebook. Two extraction techniques have been evaluated to load the processed table for future work: python native with *happybase* library and Apache Spark.

Happybase is a developer-friendly Python library to interact with Apache HBase (happybase.readthedocs.io). It provides built-in APIs to read data from a database instance.

PySpark is the Python API for Apache Spark (domino.ai), an open source, distributed computing framework and set of libraries for real-time, large-scale data processing.

Both tools have been evaluated to read data from HBase. While Spark benefits from the HDFS integration to quickly ready the MapReduce output file, happybase resulted in being the best choice in terms of load performance and dataframe readiness (the df is already well formatted with the right headings). The table below summarizes the comparison, evidence of this work can be found in the Big Data section from the accompanying Jupyter notebook *DataProcessing.ipynb*.

|  |  |  |
| --- | --- | --- |
|  | Happybase | Spark |
| Read time | 2min 11s | 16.5s |
| Dataframe load time | 13.5s | 44.1s |
| Output dataframe | Ready to be analyzed | Some processing needed |

Table - Comparison between PySpark and Happybase

# Time series Forecasting

Once data has been pre-processed and sentiment assigned to each tweet, the next step of the project has been analyzing the dataset as a time series and evaluating forecasting models. This section will cover the initial analysis of the time series, the modeling evaluation as well as the final step of summarizing everything in an interactive dashboard.

## 3.1 Analysis

At the core of time-series data is time. Time-series data is a sequence of observations or data points captured in successive order. In the context of a DataFrame, time-series data has an ordered index type DatetimeIndex (Atwan, T.A. 2022).

Before starting any activity, some manipulation has been done on the dataframe loaded from HBase, such as removing unnecessary columns for the forecasting, set the date as index and converting it into a datetime format. Once data has been properly shaped into a time-series, it was observed from an initial plot that there were some missing timestamps.

A graph with blue lines

Description automatically generated

Figure - Sentiment plot of the initial 1.6M tweets

The initial timestamps were including, besides the date, the hour, minutes and seconds. If data is collected at a very high frequency, it might contain a lot of noise; Resampling to a lower frequency can help reduce it, but it can be also useful for handling missing values and and use methods to fill in those values with the lowest noise possible (Brownlee, J. 2016). Because of those reasons, the original time-series has been resampled to a daily frequency. The new time-series is now made of 81 observations with 33 missing values.

Evidence of the work for this section can be found in the accompanying Jupyter notebook *DataProcessing.ipynb*.

A graph with blue lines

Description automatically generated

Figure - Plot of daily sentiment

### 3. 1. 1. Dealing with missing values

Having 40% of missing data is quite significant and could potentially impact the reliability of the analysis. Different imputation techniques have been evaluated, keeping in mind that imputing missing data has implications such as introducing bias and noise, or distort the underlying relationships between variables.

Unfortunately, the original missing data was not provided, so there was no option to use metrics like the root mean squared error to evaluate the best imputation method. The strategy used to select the best filling methods has been mainly looking at the produced timeseries components (trend, seasonality, noise) and choose the best one combining this plus the knowledge of how the filling techniques work.

The trend component represents the long-term movement or direction in the time series; it indicates whether the data is increasing, decreasing, or relatively stable over time. The seasonal component represents the repeating patterns or cycles, it captures the regular ups and downs that occur at fixed intervals. The residual component represents the unexplained variability in the time series that is not attributed to the trend or seasonality. The filled time series for each imputation methods is summarized in Figure 8 below.

A graph of a graph

Description automatically generated with medium confidence

Figure - Comparison of filled time-series with different methods.

All methods created a time series with similar trend and seasonality, along with an acceptable level of noise. From a graphical perspective, the result is similar for the backward/forward fill technique and linear interpolation. The original time series has most of the missing values at the beginning, while it’s denser towards the end, this means there are enough future observations to make backfill more suited, as it fills the values backwards. On the other hand, the data shows a significant trend over time, so it can introduce bias if the most recent values are not representative of the missing values. Linear interpolation has then been chosen as fitting technique, as it provides a more accurate representation of how the data changes over time, making it suitable for datasets with clear trends or seasonality like in this case. Further evaluation will be done during the modelling phase comparative analysis.

## 3.2 Modeling

Once the candidate time-series has been chosen, some steps have been taken to further prepare the data for modeling:

1. As most models are sensitive to seasonality and trends, those components have been removed using the *seasonal\_decompose* function to extract it and then remove it.

A graph of a red line

Description automatically generated with medium confidence

Figure - Time series before and after removing trend and seasonality.

1. During the evaluation of imputation methods, all time series resulted in being non-stationary, this means that its statistical properties change over time as they manifest trends and seasonality. Non-stationarity can pose challenges for time series analysis and modeling because many statistical methods and models assume stationarity for valid results After step 1, the Augmented Dickey-Fuller (ADF) test has been performed to determine the time series stationarity.
2. Finally, a plot has been produced to determine autocorrelation, where a mix of positive and negative autocorrelation has been observed.

Once trend and seasonality have been removed from the time series data, what is left is the detrended data, which represent the noise or irregular variation in time series. To gain further insights into the behaviour of this residuals, rolling mean and standard variation have been calculated and plotted on a time window of 8 days. The improvement was visually assessed: despite the last part of the time series, both rolling mean and standard deviation were mainly close to zero and stable, indicating most of the time series doesn’t exhibit any systematic pattern. This can be observed in the two Figures below.

A graph showing the value of a stock market

Description automatically generated

Figure - Rolling mean and STD of the initial time series.

A graph showing a number of data

Description automatically generated with medium confidence

Figure - Rolling mean and STD of detrended time series.

### 3.2.1 Model building and selection

When dealing with time series in Data Science and Machine Learning, there are multiple model options. The first models evaluated belong to the family of Autoregressive-Moving-Average, like ARIMA, SARIMA and Exponential Smoothing.

An autoregressive process is a regression of a variable against itself. In a time series, this means that the present value is linearly dependent on its past values (Preixeiro, 2022). Those models’ parameters are the triplet (p, q, and d) where ‘p’ indicates the number of past observations used to predict the current value, ‘q’ is the number of past error terms used for prediction, and ‘d’ is the number of differences required to make the time series stationary.

The first step has been comparing those models’ performance with each imputed time series. The results confirmed that the linear interpolation is best performing method, and the best model for it is ARIMA. Additionally, the results confirmed clearly that the error is reduced quite significantly for stationary time series. Full results are provided in the table below (the metric used is the mean squared error).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Median | ForwardFill | LinearInterp | ForwardFill  (detrended) | LinearInterp  (detrended) |
| SARIMA | 0.146114 | 0.148472 | 0.148469 | 0.044844 | 0.044684 |
| ARIMA | 0.147061 | 0.149021 | 0.149991 | 0.044988 | 0.044682 |
| EXP SMOOTHING | 0.145991 | 0.146996 | 0.147818 | 0.045777 | 0.045341 |

Table - Autoregressive models MSE for different time series.

Once the best model and best parameters have been identified, the time series has been split into train and test to perform the forecast. The result was not satisfactory, as it can be seen in the Figure below, and this is mainly due to the small size of the timeseries.

A graph showing a blue line

Description automatically generated

Figure - Forecast using ARIMA model.

Small time series with very few data points may not contain enough information to reveal more complex patterns or variations. The model will tend to simplify the forecast in such cases. In an attempt to improve the predictions, the rolling forecast technique has been evaluated for this time series. It consists of continuously training a time series model on an initial period of data, forecasting a future period, evaluating the forecasts, updating the model with newly observed data, and repeating the process for multiple forecast horizons (Perera, S. 2018). Unfortunately, the error level did not improve. Evidence of this can be found in the Forecast section of the accompanying Jupyter notebook *DataProcessing.ipynb*.

The other model explored for this forecast is the Random Forest, a popular machine learning algorithm that belongs to the supervised learning technique. It is an ensemble learning method, constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees (Analytics Vidhya, 2021).

The model has been tuned and fitted in two ways: the first approach has been using *ForecasterAutoreg* (joaquinamatrodigo.github.io) as it’s specifically designed for time series forecasting, and after the time series has been treated as a normal regression problem by using the normal *RandomForestRegressor* and manually engineer the lag feature to capture the time dependencies.

The difference was quite significant, in the first case the lowest rmse is 0.0336, while the error was 0.001 for the second test. Having manual control on the lag definition resulted in being a better approach than relying on *ForecasterAutoreg* built-in functionality.

A graph showing a graph of a training

Description automatically generated with medium confidence

Figure - Forecast using Random Forest

An additional regressor, LGBMRegressor, has been explored, but the prediction was not satisfactory as the Random Forest that could better handle the small amount of data.

## 3.3 Dashboard

The final step of this project consists of summarizing in a dashboard the insights gathered via the MapReduce jobs, as well as making forecast of the tweet sentiment using the tuned Arima and Random Forest method. Models have been saved from the Jupyter notebook and deployed into the dashboard. The data visualization for the dashboard uses Plotly. The dashboard is interactive and allows the user to select the date range for the forecasts.

To run the dashboard, please run the accompanying Jupyter notebook called *Dashboard.ipynb*, navigate to <http://localhost:8050> and ignore any warning that might show up. It advised zooming on the forecast graph to better see the details of the predictions.

While building the dashboard, Tufte’s guiding principles for data visualization (Tufte, 2001) were used to have some guidelines. In most cases an editorial decision was made to include elements which Tufte would recommend against, such as grid lines; they have been included in the plots to provide a clearer understanding of the value represented for each date.

A screenshot of a computer screen

Description automatically generated

Figure - Dashboard tab with tweets insights.

A screenshot of a computer

Description automatically generated

Figure - Dashboard tab with forecasting models.

# Conclusion

The architecture for this project included Hadoop, HBase and forecasting of time series data. The provided data was quite big and not ready for analysis/modeling, Big Data techniques have therefore been evaluated and used to process the data and store in two table both processed and initial tweets.

Forecasting very small time series presents specific challenges due to the limited amount of historical data available for model training and evaluation and, for this specific time series, Random Forest produced the best results possible among the options explored.

Finally, an interactive dashboard has been produced with insights coming from Big Data processing as well as the trained forecasting models.

# References

Benchant.com. (2022). Available at: https://benchant.com/blog/ycsb [Accessed 6 Nov. 2023].

Amazon Web Services, Inc. (n.d.). What is Apache HBase? | AWS. [online] Available at: https://aws.amazon.com/it/big-data/what-is-hbase/ [Accessed 6 Nov. 2023].

‌hadoop.apache.org. (n.d.). MapReduce Tutorial. [online] Available at: https://hadoop.apache.org/docs/r1.2.1/mapred\_tutorial.html#Overview [Accessed 6 Nov. 2023].

domino.ai. (n.d.). What is PySpark? | Domino Data Science Dictionary. [online] Available at: https://domino.ai/data-science-dictionary/pyspark [Accessed 8 Nov. 2023].

‌happybase.readthedocs.io. (n.d.). HappyBase — HappyBase 1.2.0 documentation. [online] Available at: https://happybase.readthedocs.io/en/latest/ [Accessed 6 Nov. 2023].

Atwan, T.A. (2022). TIME SERIES ANALYSIS WITH PYTHON COOKBOOK practical recipes for exploratory data analysis, data preparation, forecasting, and model evaluation. [S.l.]: PACKT PUBLISHING LIMITED.

‌

Brownlee, J. (2016). How To Resample and Interpolate Your Time Series Data With Python. [online] MachineLearningMastery.com. Available at: https://machinelearningmastery.com/resample-interpolate-time-series-data-python/ [Accessed 6 Nov. 2023].

Peixeiro, M. (2022). Time Series Forecasting in Python. Simon and Schuster.

‌ Perera, S. (2018). Rolling Window Regression: a Simple Approach for Time Series Next value Predictions. [online] Making Sense of Data. Available at: https://medium.com/making-sense-of-data/time-series-next-value-prediction-using-regression-over-a-rolling-window-228f0acae363.

‌ Analytics Vidhya. (2021). Random Forest for Time Series Forecasting for Data Science. [online] Available at: https://www.analyticsvidhya.com/blog/2021/06/random-forest-for-time-series-forecasting/.

‌ joaquinamatrodrigo.github.io. (n.d.). ForecasterAutoreg - Skforecast Docs. [online] Available at: https://joaquinamatrodrigo.github.io/skforecast/0.10.1/api/ForecasterAutoreg.html [Accessed 8 Nov. 2023].

‌Tufte, E. (2001). The visual display of quantitative information, Cheshire: Graphic Press. - 2001