

Bernardo R Fonseca | sba21227

**MSc in Data Analytics for Business
(HCI) – Apr**

Dublin 2023

X words

Please refer to the uploaded folder to access 2 Jupyter Notebook for code (**Databricks + Forecast** and **Databases Benchmark**) and screencast of practical demonstration.

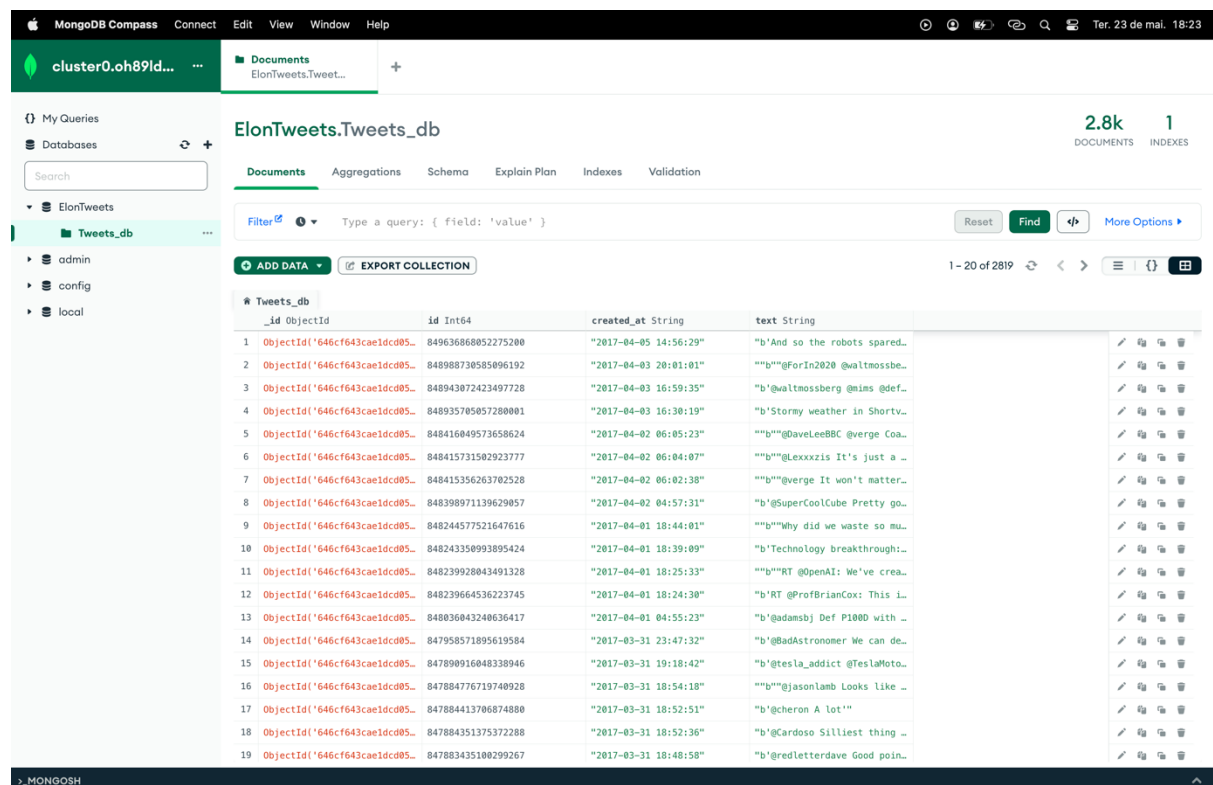
Version Control Repositorium available at: <https://github.com/BernardoRFonseca/Elon-Analysis>

1. Details of the data storage and processing activities carried out, including preparation of the data and processing the data in a MapReduce/ Spark environment

The data storage and processing activities carried out in this project involved multiple steps, including data collection, data storage, data preparation, and data analysis using Spark in a MapReduce environment. Follows a detailed overview of the developed work:

The first step was **data collection**, this process involved gathering tweets from Elon Musk's official Twitter account from 2010 to 2017. The specific methodology used for data collection was "archive" website, this data is available based on a Twitter web-scrape and it is available in an excel format. The file contains close to 3000 tweets.

The second step was **data storage** and **data export**, the collected data was first stored in a MongoDB database. MongoDB is a well-known and widely used NoSQL database that provides flexibility and scalability for storing data and specially unstructured data like tweets. Each tweet was stored as a MongoDB collection, along with the attributed 'created_at', representing the date of publication of the tweet.



#	Tweets_db	_id	ObjectId	id	Int64	created_at	String	text	String
1		ObjectId('646cf643cae1dc095...')	849636868052275200			"2017-04-05 14:56:29"		"b'And so the robots spared..	
2		ObjectId('646cf643cae1dc095...')	848988730585096192			"2017-04-03 20:01:01"		"b'""@ForIn2020 @walmossbe..	
3		ObjectId('646cf643cae1dc095...')	848943072423497728			"2017-04-03 16:59:35"		"b'@walmossberg @mins @def..	
4		ObjectId('646cf643cae1dc095...')	84893578057280001			"2017-04-03 16:30:19"		"b'Stormy weather in Shortv..	
5		ObjectId('646cf643cae1dc095...')	848416049573658624			"2017-04-02 06:05:23"		"b'""@DaveLeeBBC @verge Coa..	
6		ObjectId('646cf643cae1dc095...')	848415731502923777			"2017-04-02 06:04:07"		"b'""@Lexxxis It's just a ..	
7		ObjectId('646cf643cae1dc095...')	848415356263702528			"2017-04-02 06:02:38"		"b'""@verge It won't matter..	
8		ObjectId('646cf643cae1dc095...')	848398971139629057			"2017-04-02 04:57:31"		"b'@SuperCoolCube Pretty go..	
9		ObjectId('646cf643cae1dc095...')	848244577521647616			"2017-04-01 18:44:01"		"b'""Why did we waste so mu..	
10		ObjectId('646cf643cae1dc095...')	848243350993895424			"2017-04-01 18:39:09"		"b'Technology breakthroughi..	
11		ObjectId('646cf643cae1dc095...')	848239928043491328			"2017-04-01 18:25:33"		"b'""RT @OpenAI: We've crea..	
12		ObjectId('646cf643cae1dc095...')	848239664536223745			"2017-04-01 18:24:30"		"b'RT @ProfBrianCox: This i..	
13		ObjectId('646cf643cae1dc095...')	848036043240636417			"2017-04-01 04:55:23"		"b'@adamsbj Def P1000 with ..	
14		ObjectId('646cf643cae1dc095...')	847958571895619584			"2017-03-31 23:47:32"		"b'@BadAstronomer We can de..	
15		ObjectId('646cf643cae1dc095...')	847890916048338946			"2017-03-31 19:18:42"		"b'@tesla_addict @TeslaFoto..	
16		ObjectId('646cf643cae1dc095...')	847884776719740928			"2017-03-31 18:54:18"		"b'""@jasonLamb Looks like ..	
17		ObjectId('646cf643cae1dc095...')	847884413786874880			"2017-03-31 18:52:51"		"b'@cheron A lot!"	
18		ObjectId('646cf643cae1dc095...')	847884351375372288			"2017-03-31 18:52:36"		"b'@Cardoso Silliest thing ..	
19		ObjectId('646cf643cae1dc095...')	847883435100299267			"2017-03-31 18:48:58"		"b'@redletterdave Good poin..	

Figure 1 - Screenshot of database present in MongoDB database

After the data being stored in MongoDB, it was exported into a CSV (Comma-Separated Values) format, this facilitates data transfer and compatibility between different systems.

Once the data was exported, the same dataset was imported into a MySQL database. MySQL is a popular relational database management system, it is designed to provide robust storage and

querying capabilities. This step was taken solely with the intention of doing a comparative analysis of at least two databased using a benchmarking tool, for this reason, no other action was taken and data was then exported again.

The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'Elon_db' database selected. The main area shows a query result for the 'Elon_db.mongodb_extract' table. The query is a simple SELECT statement. The result grid shows a list of tweets with their IDs, creation timestamps, and the text content. The bottom of the interface shows the 'Action Output' tab, indicating that the query was completed successfully, returning 2819 rows.

id	created_at	text
849638686052275200	2017-04-05 14:56:29	b"bAnd so the robots spared humanity ... https://t.co/v7JUJQWICV
848988730585096192	2017-04-03 20:01:01	b"b@ForIn2020 @walmossberg @mims @defcon_5 Exactly. Tesla is absurdly overvalued if based on the past"
848943072423497728	2017-04-03 16:59:35	b"b@walmossberg @mims @defcon_5 Ei tu, Wait?"
848935705057280001	2017-04-03 16:30:19	b"bStormy weather in Shortville ...
848418049573658624	2017-04-02 06:05:23	b"b@DaveLeeBBC @verge Coal is dying due to nat gas fracking. It's basically dead.=====
848415731502923777	2017-04-02 06:04:07	b"b@Luxxxis It's just a helicopter in helicopter's clothing"=====
848415356263702528	2017-04-02 06:02:38	b"b@verge It won't matter"=====
848398971138629057	2017-04-02 04:57:31	b"b@SuperCoolCube Pretty good"
848244577521647616	2017-04-01 18:44:01	b"b"bWhy did we waste so much time developing silly rockets? Damn you"
848243350593895424	2017-04-01 18:39:09	b"bTechnology breakthrough: turns out chemtrails are actually a message from time-traveling aliens describing the secret of teleportation"
848239628043491328	2017-04-01 18:25:33	b"b@RT @OpenAI: We've created the world's first Spam-detecting AI trained entirely in simulation and deployed on a physical robot: https://t.co/va2v80xa6...
848239664536223745	2017-04-01 18:24:30	b"bRT @ProfBrianCox: This is extremely important from @elonmusk and @SpaceX - reusable rockets bring us MUCH closer to becoming a spacefaring@ve2...
84803604324036417	2017-04-01 04:55:23	b"b@adamsbj Def P100D with Ludicrous+, although the rocket starts going a lot faster after that"
847958571895619584	2017-03-31 23:47:32	b"b@BasAstronomer We can def bring it back like Dragon. Just a question of how much weight we need to add."
847890916048038946	2017-03-31 19:18:42	b"b@tesla_addict @TeslaKotons Working on it"
847884777819740928	2017-03-31 18:54:18	b"b"b@jasonlamb Looks like it could do 20% more with some structural upgrades to handle higher loads. But that's in fully expendable mode.=====
847884413706874880	2017-03-31 18:52:51	b"b@cheron A lot"
847884351375372288	2017-03-31 18:52:36	b"b@Cardoso Silliest thing we can imagine! Secret payload of 1st Dragon flight was a giant wheel of cheese. Inspired bva2v80xa6 https://t.co/68nMjkPaC
847893435100299267	2017-03-31 18:48:58	b"b@redtentative Good point, odds go from 0% to <g>0% j"
8478829090906341888	2017-03-31 18:46:53	b"bFalcon Heavy test flight currently scheduled for late summer"
847882289581359104	2017-03-31 18:44:25	b"bConsidering trying to bring upper stage back on Falcon Heavy demo flight for full reusability. Odds of success low, but maybe worth a shot."
847881536254955521	2017-03-31 18:41:25	b"b@oxfordreddy Browser is already a little better. Kernel and browser update in prob 6 weeks or so (lots of underlyin@ve2v80xa6 https://t.co/XlgQGwJ21D)
847610890506208257	2017-03-31 00:45:56	b"bRT @SpaceX: More photos from today@ve2v80v99s Falcon 9 launch and first stage landing @ve2v89v92 https://t.co/095WHX44BX https://t.co/c3UYhM...
847608431582098000	2017-03-31 00:38:12	b"b@BasAstronomer @SpaceX Thanks Prof"
847594208219336705	2017-03-30 23:39:41	b"bIncredibly proud of the SpaceX team for achieving this milestone in space! Next goal is refight within 24 hours."
847580067446345728	2017-03-30 22:43:30	b"bRT @SpaceX: Falcon 9 first stage has landed on Of Course I Still Love You @ve2v80v94 world@ve2v80v99s first refight of an orbital class rocket."
847561780532523008	2017-03-30 21:30:50	b"bRT @SpaceX: -40 minutes until launch window for SES-10 opens. All systems and weather are go. Watch here @ve2v89v92 https://t.co/gIC39uBC7z http...
847510437054877698	2017-03-30 18:06:48	b"bRT @SpaceX: Falcon 9 and SES-10 vertical on Kennedy Space Center@ve2v80v99s historic Pad 39A. Launch window opens at 6:27pm EDT, 10:27pm U...
8472594545755820033	2017-03-30 01:29:51	b"bMade today on Tesla sketch pad https://t.co/ZbdfP2NN41 "
847234747747020800	2017-03-29 23:51:19	b"bIf you just downloaded V8.1, tap the the T on center screen three times"

Figure 2 - Screenshot of database present in MySQL database

The third step represented an upload to DBFS (Databricks File System) by Databricks. DBFS is a distributed file system provided by Databricks. This tool allows users to store and access data in their Databricks workspace, this is a way of analysing big data in a non-local environment and process it. Uploading the file to DBFS enabled easy access and processing of the data using Spark.

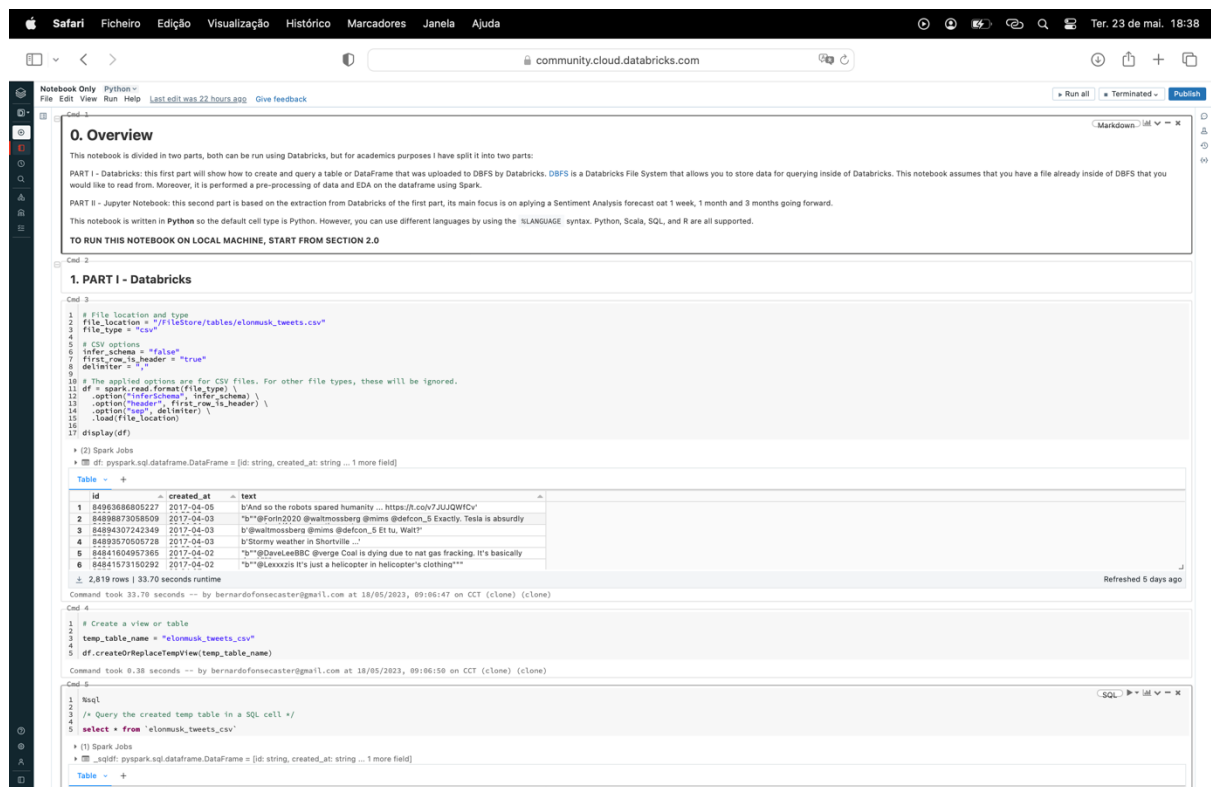


Figure 3 - Database in Databricks

The fourth step was **data preparation and processing** with Spark Databricks, the uploaded CSV file was analysed and processed using Spark, a powerful distributed computing framework. Spark is a tool design, among other things, to provide an interface for distributed data processing and offers various APIs, including Python, for data manipulation and analysis. Spark's SQL DataFrame API used to handle the CSV file. Preliminary data preparation task were performed which included checking and modifying data decription/types, duplicates and unique values. It is important to mention that Spark's distributed processing capabilities enable scalability and performance optimizations. The programming model MapReduce is internally used by Spark to parallelize computations across multiple nodes in a cluster.

The fifth step was to conduct a **forecast-oriented preparation and processing** in a Jupyter Notebook in my local machine. In this step, the notebook from Databricks was exported together with the dataset. On local machine, Feature Engineering was performed, this included feature modification such as deleting special characters (@, #, RT and other web/twitter-related language that would not be helpful to the analysis), move all characters to low case and feature creation which included a breakdown of date related features that are helpful for EDA and forecast. The features 'num_words' (number of words per tweet), 'num_characters' (number of characters per tweet) and 'mean_characters' (average number of characters per tweet were as well created.

In summary, the data storage and processing activities involved the collection of tweets from Elon Musk dating 2010-2017, storing them in MongoDB, exporting to a new format, importing into MySQL, exporting to CSV, uploading to DBFS (Databricks), and finally performing data preparation and analysis using Spark in a MapReduce environment which was then completed in local machine in a jupyter notebook environment. This pipeline was draw to meet academic requirements and could nbe simplified. Nonetheless, is was a way to efficient processing of the collected data, enabling various exploratory and analytical tasks to be performed.

2. **A discussion of the rationale and justification for the choices you have made in terms of data processing and storage, programming language choice, machine learning models and algorithms that you have implemented**

When it comes to data processing and storage, programming language choice, machine learning models and algorithms, there are several factors that I had to consider. In this question I will discuss the rational and justification behind the choices made in each of these topics.

For **data processing and storage** I used 4 tools. **MongoDB** was chosen as the initial storage solution for Elon Musk's tweet data. MongoDB is known and reliable NoSQL database that masters the handling of unstructured data, this makes it suitable to store tweets, data know for the particular varying formats and structures. Adding to this, the scalability and flexibility allows for easy updates and additions to the data schema. After the first exporting, data was imported into **MySQL** which is a popular relational database management system, it is designed to provide robust storage and high querying capabilities. The decision to use MySQL might was taken/driven by the need for relational querying and analysis, and as important is the compatibility with other systems and tools that would further be needed. The exported dataset was uploaded using **DBFS (Databricks File System) by Databricks**. DBFS is a distributed file system provided by Databricks. This tool allowed me to store and access data in their Databricks workspace, this is a way of analysing big data in a non-local environment and process it. Uploading the file to DBFS enabled me easy access and processing of the data using Spark. By exporting this to my local machine, I followed the analysis in a **Jupyter Notebook**, the major benefits of using this tool includes the ease of use, the ability to work with every available Python libraries, the ease/flexibility to customize the data processing steps based on specific requirements and the option of accessing offline.

For **programming language** I choose **Python**, extremely versatile, high-level and widely used in the data science and machine learning world. This language offers a complex-rich ecosystem of libraries and frameworks that can be used for various purposes, including data processing, analysis, and modelling. Python is simple, readable and extensively supported by different communities. All of this makes it an amazing choice for tasks the ones performed in this context.

For **machine learning models and algorithms**, I will mention two aspects. For **sentiment analysis** I used **Textblob**, it is an immensely easy-to-use and convenient library for the purpose of performing sentiment analysis. By having pre-trained models, a simple API is capable of calculating sentiment polarity (this means positive, negative or neutral) and subjectivity (how subjective or objective the text is). The choice to use Textblob for sentiment analysis is purely driven by its own simplicity, ease of integration, and reasonable performance for these tasks. For **forecasting**, I used **SARIMAX** (Seasonal Autoregressive Integrated Moving Average with Exogenous Variables), it is a time series forecasting model that does not exclude (takes into account) the aspect of seasonality of data and can handle exogenous variables. As I am analysing tweets over time, this is a model capable for over time sentiment forecasting, as it can capture patterns and trends in the sentiment data, it leverages the autoregressive, integrated, and moving average aspects of data to capture dependencies/patterns. It is as well important to mention that this model can be adjusted to accommodate different forecasting horizons (such as one week, one month or three months).

My rationale and justification was from the starting point structured over three aspects: simplicity/ease-of-use, compatibility/integration and performance. Simplicity/ease-of-use - In data processing, programming language, and models were driven by the simplicity and ease of use they offer. MongoDB and MySQL avail straightforward storage solutions for any given format of data, while

Python and all its libraries, particularly Textblob and SARIMAX, makes the implementation of sentiment analysis and time series forecasting simple! Compatibility/integration – As this project was developed for academic purposes a wide variety of tools were needed to show the grasp of content knowledge, for this to be possible individual compatibility and integration with other tools and systems are must-have-aspects. Performance/scalability – While Textblob is an extremely convenient sentiment analysis solution, in terms of performance it is just reasonable (especially when compared to more advanced NLP models) but this choice is justified by the main goal which is to quickly assess the sentiment of the tweets without extensive model training or even complex feature engineering. The same applies to the model SARIMAX, it is a well-established forecasting model that is well-capable of providing reasonable predictions for sentiment trends over time.

Overall, the choices made in terms of data processing and storage, programming language, and machine learning models were influenced by the needs and requirements set for this project, as mentioned above, all was driven by simplicity/ease-of-use, compatibility/integration and performance and the end-goal of gaining insights into the sentiment of Elon Musk's tweets. While more advanced NLP models and techniques could potentially outcome a more accurate results, the ones showed are extremely satisfactory and meet the specific goals, resources, and trade-offs of the analysis.

3. Comparative analysis for at least two databases using any benchmarking tool. (For example, ycsb)

Comparative data can be found on Jupyter Notebook "Databases Benchmark"

The results obtained from the comparison between MongoDB and MySQL databases deep and valuable insights into performance and behavior of both. The metrics measured include runtime, throughput, insert operations, average write latency, minimum write latency, maximum write latency, 95th percentile latency and error rate. Below is a screenshot of the combined results:

Combined Load Test Results:			
	Metric	MongoDB	MySQL
0	RunTime(ms)	58452.766000	345.406000
1	Throughput(ops/sec)	0.048227	8.161410
2	Insert Operations	2819.000000	2819.000000
3	Average Write Latency(us)	20.694244	0.120663
4	Min Write Latency(us)	13.715000	0.076000
5	Max Write Latency(us)	225.895000	3.492000
6	95th Percentile Latency(us)	28.909000	0.344000
7	Error Rate	0.000000	0.500000

Figure 4 - Benchmark Results

Starting with runtime, MongoDB shows a significantly higher value of 58,452.8 milliseconds when compared to MySQL's that holds a 345.4 milliseconds. This indicates that the MongoDB load test tooks a much longer amount of time to complete than MySQL load test. This highlights a potential difference in the efficiency of the two databases in handling the given workload.

Moving on to throughput, MongoDB demonstrates a lower value - 0.048227 operations per second, while MySQL reached a significantly higher value of 8.161410 operations per second. This indicates that MySQL processed operations at a much faster/higher rate, this does represent a better performance in terms of handling the workload and executing write operations.

In terms of insert operations, both MongoDB and MySQL performed the same number of operations, recording 2,819 insertions, the reason is that both represent the same dataset records. Examining the

average write latency, MongoDB exhibits a value of 20.69 microseconds, MySQL reached a significantly lower value of 0.120 microseconds. This indicates that MySQL was able to write data in a quickly manner, with lower latency, this means better responsiveness and efficiency in handling write operations. Looking into the minimum and maximum write latencies, MongoDB shows higher values of - 13.715 and 225.895 microseconds. MySQL displayed lower values - 0.076 and 3.492 microseconds. These results reinforce that MySQL outperformed MongoDB. For 95th percentile latency, which represents the latency value below which 95% of the write operations fall, MongoDB demonstrates a higher value of 28.909 microseconds compared to MySQL's 0.344 microseconds.

Lastly, examining the error rate, MongoDB records a 0% error rate, indicating no errors occurred during the load test. On the other hand, MySQL reports a 50% error rate, suggesting that half of the operations encountered potential errors.

In conclusion, the results highlight the performance disparities between MongoDB and MySQL databases in this specific test scenario. MongoDB exhibited a longer runtime, lower throughput, and higher latencies and MySQL reached better performance in terms of throughput, write latency, and error rate. These findings emphasize the importance of considering the specific workload requirements and performance expectations when selecting a database system, these need to be evaluated based on the specific use case.

4. Your analysis of any change sentiment that occurs over the time period that you have selected

On the overviewed time-period of 2010 to 2017, I conducted a sentiment analysis on Elon Musk's tweets and applied a SARIMAX model with the goal of performing a sentiment forecast. The sentiment analysis was performed using Textblob, which provided measures of polarity and subjectivity for each tweet presented in the dataset. To better understand the change of sentiment occurred across time, it is important to understand the meanings of polarity and subjectivity. **Polarity** refers to the sentiment expressed in a tweet, in terms of nominal value it can range from -1 (negative) to 1 (positive), with 0 indicating neutrality. **Subjectivity**, on the other side, bring a measure that looks into extent to which a tweet expresses personal opinions rather than factual information, this can range from 0 to 1.

Upon examining the sentiment analysis results, I started by analysing the 'all-time' score of both measures, and it became evident that tweets lean towards the positive side (46.6%, where neutral resents 41.4% and negative are 12.1%:

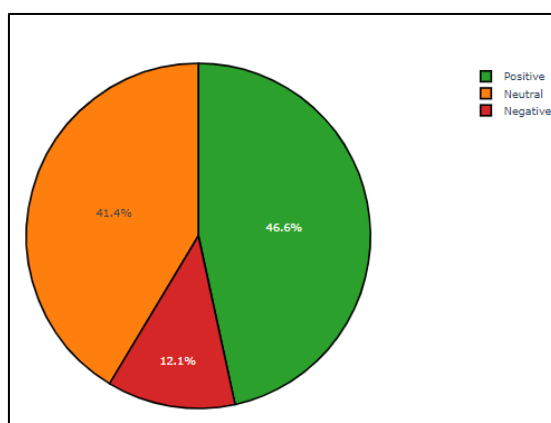


Figure 5 - Pie Chart - Polarity Score

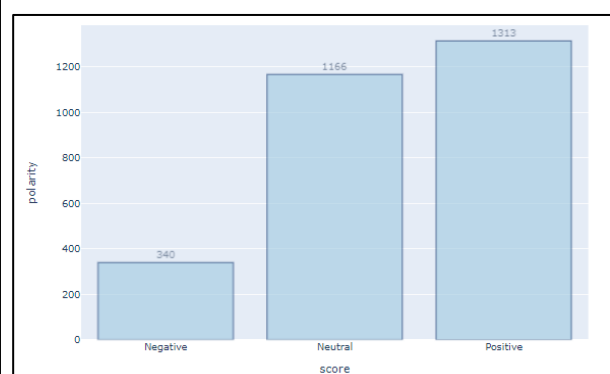


Figure 6 - Bar Chart - Polarity Score

It is as well evident that Elon Musk tweets lean towards the subjective side:

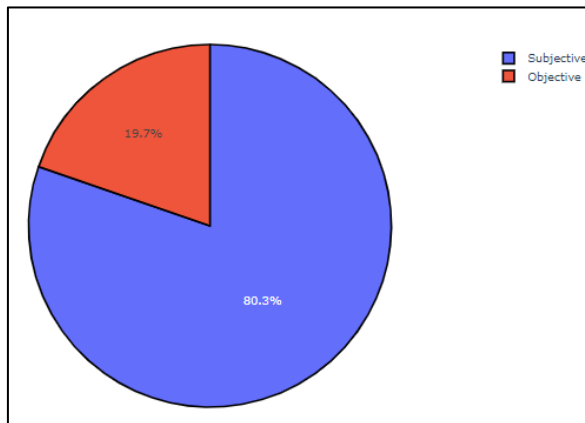


Figure 7 - Pie Chart - Subjectivity Score

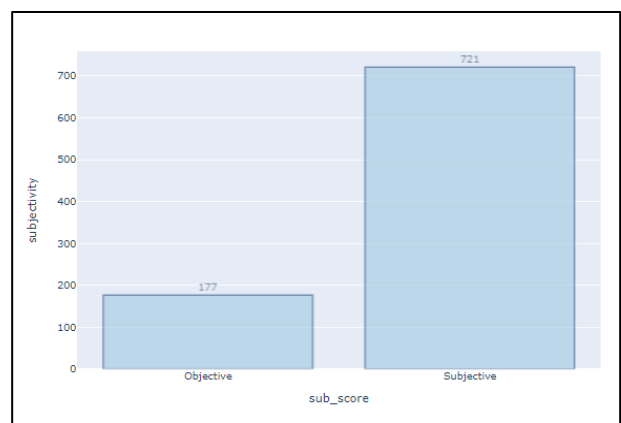


Figure 8 - Bar Chart - Subjectivity Score

After this quick all time-analysis, I grouped the tweets in years and measured the average of these scores for each of the years presented (from 2010 to 2017), this would give a better understanding of sentiments behaviour across time, and possible define a trend, starting by **polarity score**:

	year	polarity
0	2010	0.166667
1	2011	0.115176
2	2012	0.178227
3	2013	0.138702
4	2014	0.097856
5	2015	0.121686
6	2016	0.122195
7	2017	0.124825

Figure 9 - Average Polarity per year

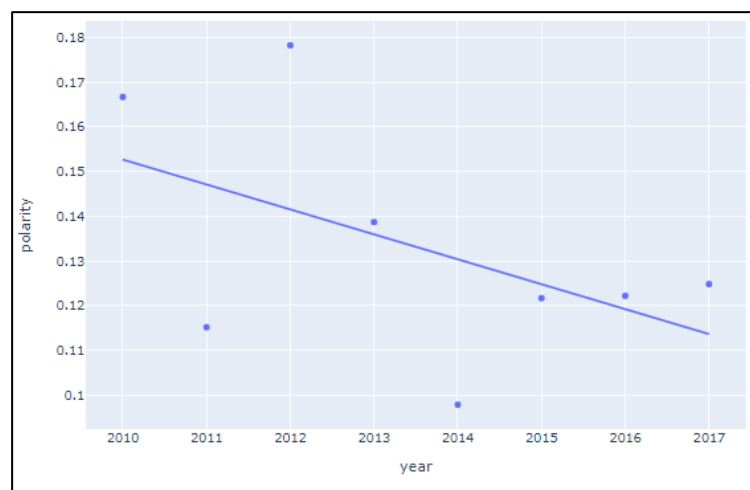


Figure 10 - Average Polarity per year and trend-line

These plots show each year's average polarity score of the tweets. This displayed the change sentiment that occurs over time. The average score in 2010 was 0.166 and went down to 0.125 in 2017. The trend line in the plot is facing downwards meaning that we have a negative trend, we can from this understand that the sentiment is moving towards neutrality/negativity.

For subjectivity, the same analysis was run and the results are displayed below:

	year	subjectivity
0	2010	0.366667
1	2011	0.376722
2	2012	0.400128
3	2013	0.350900
4	2014	0.325523
5	2015	0.332396
6	2016	0.305052
7	2017	0.332885

Figure 11 - Average Subjectivity per year

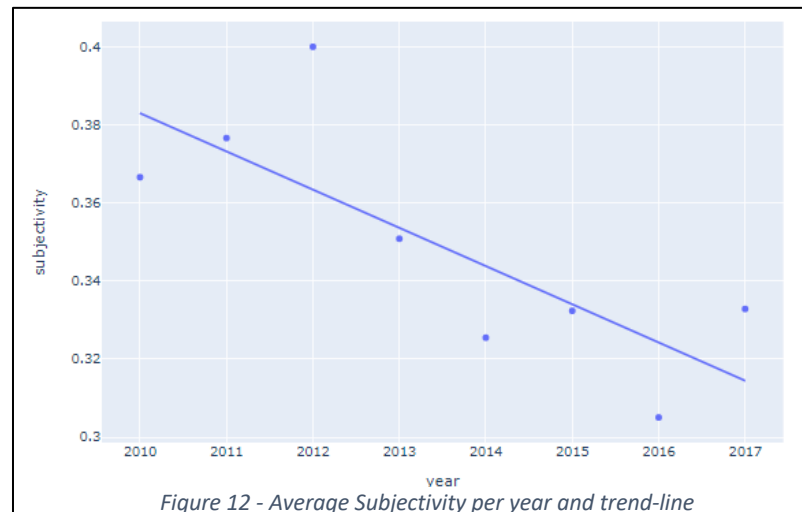


Figure 12 - Average Subjectivity per year and trend-line

These plots show each year's average subjectivity score of the tweets. This displayed the change that occurs over time. The average score in 2010 was 0.366 and went down to 0.332 in 2017. The trend line in the plot is facing downwards meaning that we have a negative trend, we can from this understand that the sentiment is moving towards objectivity rather than subjectivity.

In conclusion, the sentiment analysis revealed an overall positive and subjectivity sentiment in Elon Musk's tweets from 2010 to 2017. Throughout time these results trend downwards meaning that we are data is pointing towards both neutrality/negativity (polarity score) and objectivity (subjectivity score). This analysis is vital and the model used in the forecast should be able to grasp these correlations to better predict future sentiments.

5. Your forecast of the sentiment at 1 week, 1 month and 3 months going forward

Based on the sentiment analysis forecast done with the help of **SARIMAX** (Seasonal Autoregressive Integrated Moving Average with eXogenous repressors) model, I generated sentiment predictions based of **polarity** score for the following **1 week, 1 month, and 3 months**. By evaluating the plot diagnosis and comparing the results using two metrics such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), I can provide insights into the forecasted sentiment at each of these time horizons.

MSE (Mean Squared Error) and RMSE (Root Mean Squared Error) are widely used metrics to evaluate the accuracy of a forecast or prediction model. These calculate the differences between predicted values and actual observed values. These metrics provide an end-result that tell how well the model outcome is aligned with the real data. **MSE** calculates the average squared difference between the predicted values and the real- values. **RMSE** is the square root of the MSE, this metric provides a more interpretable value of prediction error, on the same scale as the original data. Both MSE and RMSE are useful for comparing the accuracy of different models, as RMSE is in the same format as the original data, it makes it easier to interpret, a lower RMSE value indicates better accuracy as it shows a closer alignment between prediction and real-data.

With the results of the predictions, I can created a comparative data frame to easily compare the performance:

	1 Week	1 Month	3 Months
MSE	0.111017	0.072385	0.072385
RMSE	0.333192	0.269007	0.269045

Figure 13 - Model performance for 3 time horizons

Looking at the forecasted sentiment for one week, the SARIMAX model reached a MSE of 0.11 and RMSE of 0.33. The plot diagnosis indicates that the model captured the general trend of the sentiment, however, it's important to note short-term predictions might conclude in high fluctuations or uncertainties.

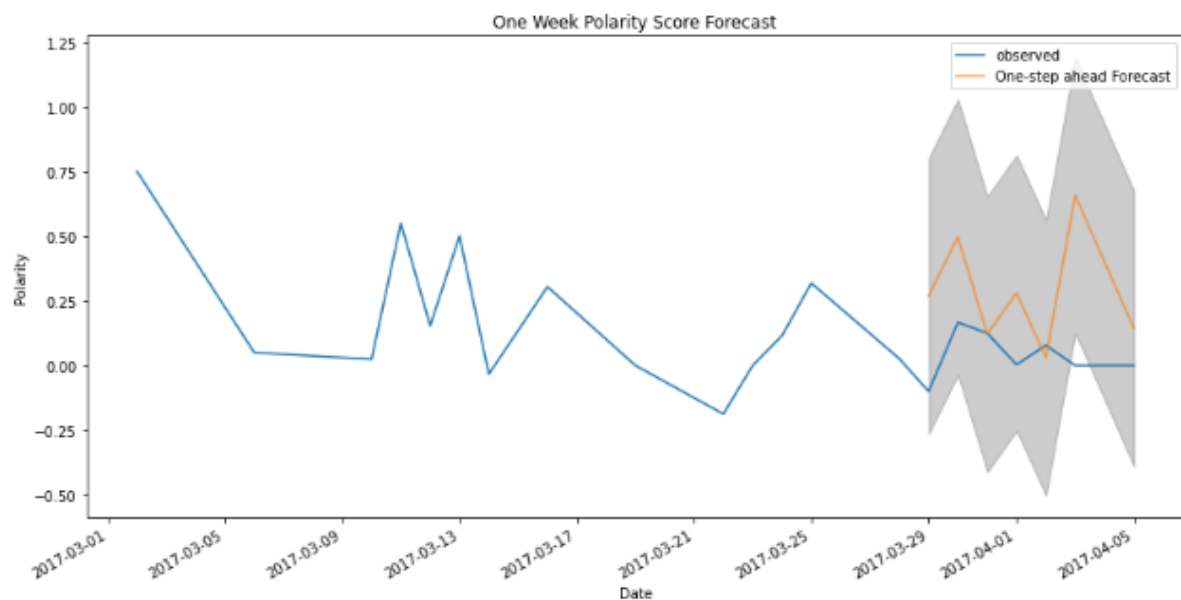


Figure 14 - One Week Forecast

Moving on to the sentiment forecast for one month, the SARIMAX model reached a MSE of 0.07 and a RMSE of 0.269, this result is significantly better than the one performed by the one-week model:

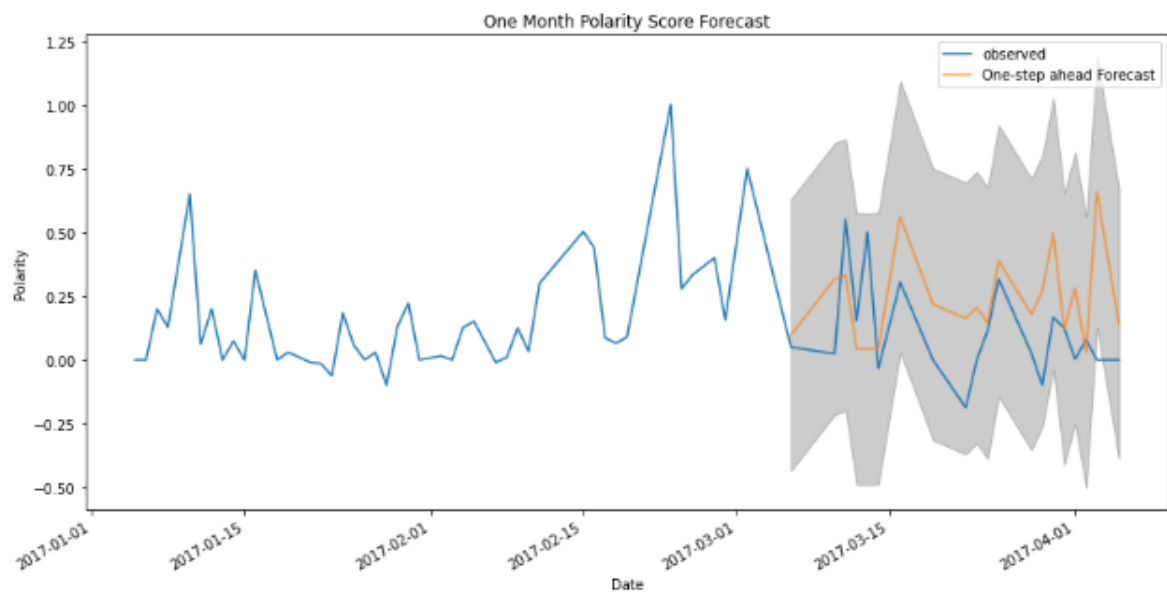


Figure 15 - One Month Forecast

Looking ahead to the sentiment forecast for three months, the SARIMAX model reached a slightly lower result when compared to the one-month, it reached a MSE of 0.07 and RMSE of 0.269. For certain periods of time (07-02-2017 to 07-03-2017) the plot diagnosis displayed a relatively unstable sentiment prediction when compared to the observed/real sentiment, the result was still satisfactory:

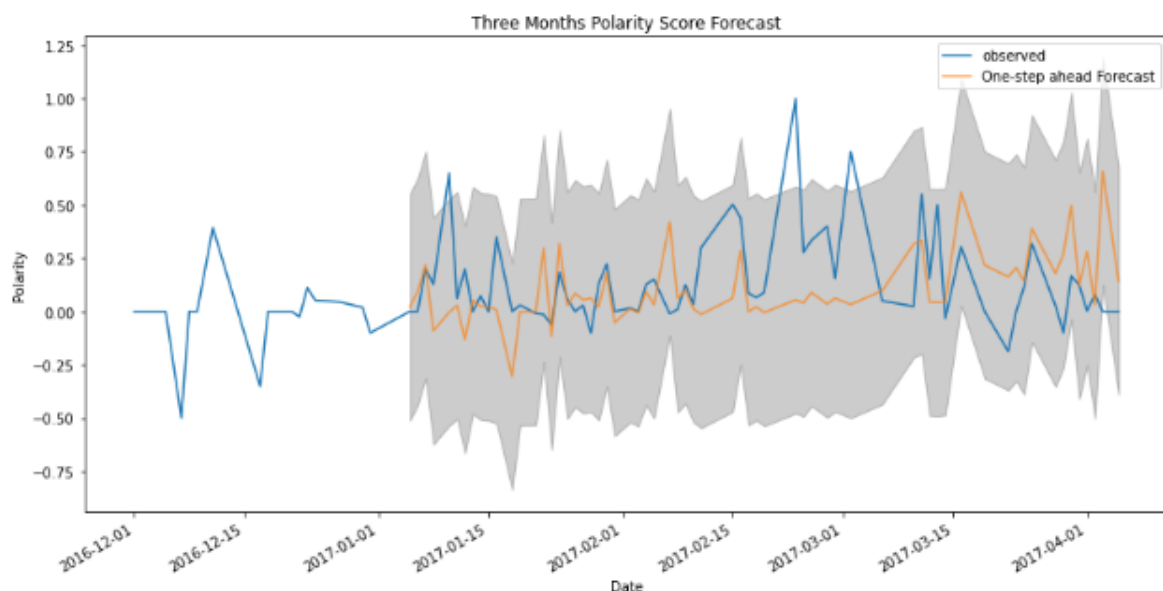


Figure 16 - Three Months Forecast

Upon comparing the forecasted sentiment with the actual sentiment data for each time horizon, the conclusion is that the one-month is the best performer. Overall, the results indicated satisfactory values, suggesting a reasonable level of accuracy in the sentiment forecast. However, it's important to interpret these metrics in conjunction with the nature of sentiment analysis, which can

be in most of the time subjective and influenced by multiple contextual factors, like news, events or even unpredictable factor that might emerge.

6. Presentation of results by making appropriate use of figures along with caption, tables, etc and your dashboard for your forecast.

Throughout the entirety of the Notebook, figures and tables have been displayed to describe data, insights and results. At the end of the Jupyter Notebook, a Dashboard was created to compile all these details. The Dashboard was built using Dash, which is a framework used for building web-apps with interactive visualizations. Below is a screenshot of this dashboard, for an interactive view, please refer to the Jupyter Notebook, point 2.6:

Elon Musk's Tweets - Sentiment Analysis Dashboard



Figure 17 - Dash Dashboard