**Group ID - MSc in Data Analytics**

Author: Rakesh Kumar Muraleedharan

e-mail: [2022589@student.cct.ie](mailto:2022589@student.cct.ie)

Student ID: 2022589

**Application of Big Data Analytics for Tweet Sentiment Analysis and Sentiment Forecast**

**Goal:** To use Big Data technologies to store the Tweet data and then build appropriate machine learning models sentiment analysis and sentiment forecast.

***ABSTRACT:***

*In this paper, we will work on Analysis tweets of a market influential personality. For our analysis we have used Elon Musk’s tweets from 2015 through 2020. We will analyse their tweets over the years. We will analyse the sentiments in their tweets. We will then do a forecast of the tweets for each week, month, quarter. The results are then presented via dashboards detailing how the predictions are compared to the actual data used for analysis. The results show that the future tweets are varying in accordance with the previous tweets, we could see the sentiments were mostly negative on few months, but they did not impact the overall predictions.*

**Keywords:**

Data Analytics, Data Preparation, Machine Learning, PYTHON, Classification, Sentiment Analysis, Elon Musk Tweets, Visualisation, Data Pre-processing.

**Version Control**: GitHub (Code, Raw data available in the link :<>)

Table of Contents

[1. Introduction 3](#_Toc135690387)

[2. Database Comparison using YCSB 4](#_Toc135690388)

[3. Methodology 8](#_Toc135690389)

[Sentiment Analysis: 17](#_Toc135690390)

[Hyper Parameter Tuning: 20](#_Toc135690391)

[Boosting Algorithm: 21](#_Toc135690392)

[Time Series Forecasting: 22](#_Toc135690393)

1. Introduction

The topic of this research is Tweet Analysis using Big Data Technologies and applying machine learning algorithms. Tweet dump from Elon Musk across the years from 2015 through 2020 is used for the analysis. The tweets from business personalities like Elon Musk influences the stock markets, specially their negative and positive sentiments, might have some influence on the stock index markets.

We will make use of Big Data Analytics to store the data dump and make some preliminary data analysis, do data cleansing, apply sentiment analysis techniques, tokenise the sentiments, analyse the positive, negative oe, and neutral sentiments. We will then build a machine learning model to forecast the feature tweets. The final appropriate models are plotted against the train, test and forecasted data. There is an interactive dashboard to allow user to select the plots they want to view dynamically.

1. Database Comparison using YCSB

Before starting with the tweet analysis using Bigdata, I will first compare two prominent databases. There are several comparison tools available in the market for quantitative analysis of the no-sql database performances like YCSB (Yahoo Cloud Serving Benchmark) , NoSQLbench.

We will use YCSB here for benchmarking. The databases we will use to compare here are MongoDB and MySQL.

YCSB is a frequently used open-source benchmarking tool. YCSB has five pre-defined workloads (A to E).

A-Update Heavy Workload

B-Read Heavy Workload

C-Read Only

D-Read Latest

E-Short Ranges

Different databases are designed to perform better in any of the operations. In YCSB the key chosen to identify a record comes from Zipfian distribution, which means that some keys are very likely to occur whiles are not. One of the advantages of using YCSB is that we can do a bench on any of the above workloads and for the records in operation.Also, we do-not need to manually create any insert/update/delete to check the performance, but YCSB does it for us based on the workload selected.

Read and Update are the most common operations that the data analysts perform. Hence we will run workloads A and B , and then compare MongoDb v/s MySQL.

A screenshot of a computer

Description automatically generated

Fig1: Comparing Workload A and Workload B

As seen in the above screenshot, Workload A has Read/Update ratio of 50/50, which in database terms means heavy update load

Workload B has Read/Update as 95/5 which is Read intense operation.

A screenshot of a computer

Description automatically generated

Fig2: Workload B showing Zipfian distribution and other parameters like the record count.

Let us view how the workloads A and B perform for MongoDB and MySQL databases.

MySQL – Workload A

A screenshot of a computer

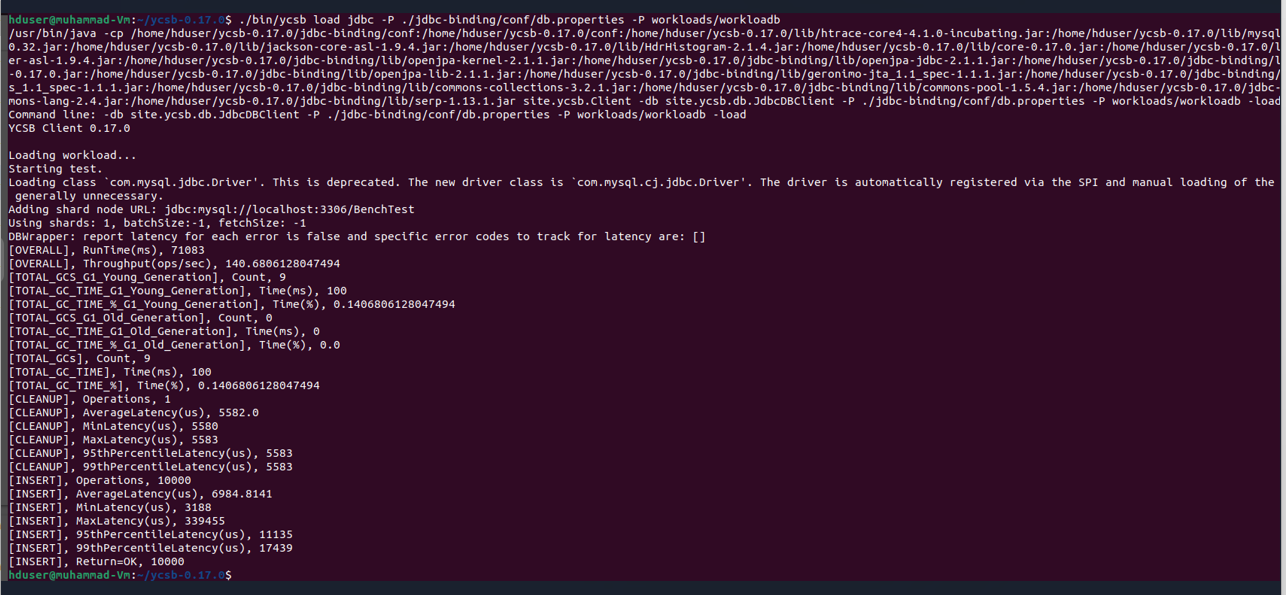
Description automatically generated

MongoDb Workload A:

A screenshot of a computer

Description automatically generated with medium confidence

MySQL Workload B:



MongoDb Workload A:

A picture containing text, screenshot

Description automatically generated

Let us summarise the key performance metrics in a tabular format:

**Workload A: UPDATE Heavy**

|  |  |  |
| --- | --- | --- |
| **Measure** | **MongoDB** | **My SQL** |
| Run Time (ms) | 3675 | 77969 |
| Throughput (ops/sec) | 2721 | 128 |
| Average Latency (us)(insert) | 219 | 7650 |
| Min Latency (us) (insert) | 65 | 3616 |
| Max Latency (us) (insert) | 222719 | 516607 |
| 95th Percentile Latency (us) | 389 | 11759 |
| 99th Percentile Latency (us) | 1910 | 20223 |

The above chart shows that Mongo DB is performing considerably well for Update Heavy Workload. The throughput that is ops/ sec for each operation is very slow in My SQL. The latency measures for insert operations are also very high for my SQL, with average latency is 7500(us) more than MongoDB. Hence based on this analysis, for 10.000 records tested in Workload A, we can say Mongo DB is recommended over MySQL for update intense operations.

Now, looking at Workload B.

**Workload B: READ Intense**

|  |  |  |
| --- | --- | --- |
| **Measure** | **MongoDB** | **My SQL** |
| Run Time (ms) | 3094 | 71083 |
| Throughput (ops/sec) | 3232 | 140 |
| Average Latency (us)(insert) | 228 | 6984 |
| Min Latency (us) (insert) | 71 | 3188 |
| Max Latency (us) (insert) | 129599 | 339455 |
| 95th Percentile Latency (us) | 411 | 11135 |
| 99th Percentile Latency (us) | 2267 | 17349 |

The above chart shows that Mongo DB is performing considerably well for Read Intense Workload. The throughput that is ops/ sec for each operation is very slow in My SQL(140 compared to 3232). The latency measures for insert operations are also very high for my SQL, with average latency is 6500(us) more than MongoDB. Hence based on this analysis, for 10.000 records tested in Workload A, we can say Mongo DB is recommended over MySQL for Read intense operations.

From the above bench marking test using YCSB, for 10000 records, based on the results we see that MongoDB performs better than MySQL for both Read Instense and well as update workload operations. Next steps (outside the preview of this document), we can compare MongoDB with other databases like Cassandra and see which databases gives better performance.

1. Methodology

The project-management methodology adopted in this research is a combination of KDD (Knowledge Discovery is Database) and **CRISP-DM** (Cross-Industry Standard Process for Data Mining).

The activities involved are research understanding, data collection, data preparation, Data Analysis, enrichment by collecting additional data, and discussion of analysis results. The method of predictive analysis, based on machine learning was chosen. Training, validation, and testing of machine learning models were performed. A complete set of activities is performed programmatically using the open-source software PYTHON, BIG Data technologies like Apache Spark, Hadoop are also used.

***Below is how the entire analysis is designed.***

* 1. **Research Understanding Phase:**

Understanding the topic of research, Sentiment analysis on Tweet for at least one year and then further forecasting using appropriate machine learning models. This involved detailed study on Big Data, technologies to handle bigdata. Hadoop storage, Apache Spark were the chose technologies for this. Understanding Apache Spark, options available for data manipulation was researched, Advantages and disadvantages of using Spark was researched, simplicity of PySpark SQL was understood. Sentiment analysis including feature engineering, creating sentiments based on tweets, analysing them using the charts, count- vectorization techniques. Times series Forecasting basics, understanding of different Python libraries that does the forecasting, dashboard libraries are also explored.

* 1. **Data Collection:**

At least one year worth of Tweets on a any chosen topic was to be extracted for sentiment analysis. Data Scrapping from Twitter was attempted using the Twitter API. However, because of the policy changes for the twitter developer accounts, data scrapping could not be implemented. Instead, readily available dataset from Kaggle ([Elon Musk Tweets | Kaggle](https://www.kaggle.com/datasets/gpreda/elon-musk-tweets)) are used for research and exploring sentiment analysis and Forecasting. The dataset collected qualify as big data since the data is of high velocity, veracity, volume, variety, and value. Sample analysis done on this data to see if the data is complete and can be used for machine learning.

* 1. **Data Preparation**

Big Data Technologies like Hadoop and Apache Spark are used for data cleaning and preparation. Hadoop is an environment supporting multiple big data databases; hence Hadoop is selected to store the huge dataset. Apache Spark SQL is simple to use and can handle huge data with ease, they have high speed, can do advanced analytics, it is powerful, and has access plethora of PYTHON libraries. Hence Hadoop and Spark are the chosen technologies here.

The first step is to export the file to Hadoop filesystem.

*Command:* ***hadoop fs -put Elonmusktweets.csv /user1***

<Hadoop screenshot>

Libraries used, including the spark libraries.

A screen shot of a computer

Description automatically generated with low confidence

Read the file from the Hadoop File System.

A screenshot of a computer code

Description automatically generated with low confidence

The next step is to view the columns in the file. Then select only the relevant fields, only tweet, id and date are relevant for our analysis and hence rest of the columns are ignored.

A screenshot of a computer code

Description automatically generated with medium confidence

The stopwords are then removed from the tweet.

The below screenshot demonstrates creating a user defined function, its usage. Spark SQL is used where the UDF is called within the select SQL, filter is used to remove blank tweets.

A screenshot of a computer program

Description automatically generated with low confidence

In the next series of steps, the year/month and weekday from the tweet dates are extracted.

Extract Year:

A screenshot of a computer

Description automatically generated with low confidence

Extract Month:

A screenshot of a computer

Description automatically generated with low confidence

Extract Weekday:

A screen shot of a computer

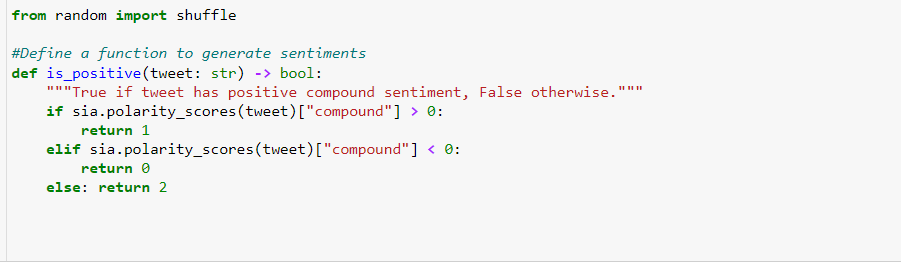
Description automatically generated with low confidence

Confirming the dataset contains one year worth of Tweets:

A screenshot of a computer program

Description automatically generated with medium confidence

The dataset does not contain sentiment column; hence we should analyse the sentiments based on the actual tweets. We will use SentimentIntensityAnalyzer module in nltk. sentiment library. This is very simple to use and doesn’t consume huge RAM resources, multiple functionalities like word frequency, collocations, quick sentiment analysis are available with NLTK. Hence this library is used.

Define a function to derive the polarity scores: 

A screenshot of a computer program

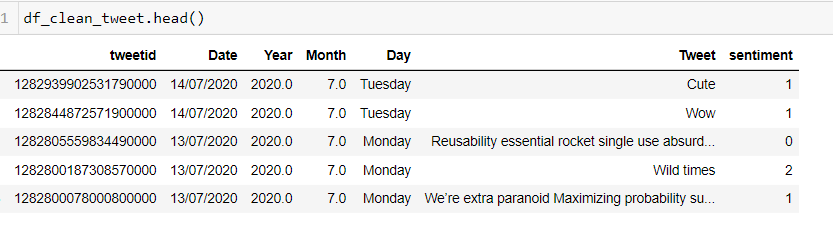
Description automatically generated with low confidence

Using rdd map function to map the data frame columns with the sentiment generated.

A screenshot of a computer

Description automatically generated with low confidence

After clean-up and mapping the sentiment:



**1-> Positive**

**0->Neutral**

**2->Negative**

Next step , export this data frame to a csv file and then save it in GitHub, we will then use it on the local machine. Further data preparation will be done here.

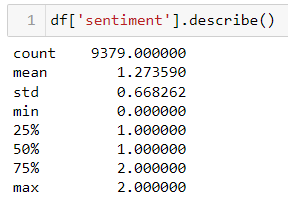
*There were some virtual machine related issues while trying to import the tensor flow for machine learning algorithms, hence exported them and then worked on the local machine.*

Next step, we will download the csv file generated in the previous step from GitHub.

A screenshot of a computer

Description automatically generated with medium confidence

Descriptive Statistics:



Since the sentiment values are classified as 1(positive),0(neutral), 2(negative) , the value in the descriptive statistics that is of interest is mean. It is 1.273590, that is between positive and negative (25% above positive on an average).

A picture containing text, screenshot, diagram, plot

Description automatically generated

A picture containing screenshot, plot, diagram, line

Description automatically generated

As per the above chart, Friday is the day where there are maximum tweets, most of the tweets on Friday is 1(positive); followed by 2(Negative), 0(neutral) is relatively less. Similarly maximum tweets are seen in May with highest being 1(positive); followed by 2(Negative), 0(neutral).

Next step we will create Bag of Words using Count Vectorizer. We need to do this because the machine learning model we apply for the sentiment analysis needs a numerical input as the feature. We will predict sentiments based on the tweets as an independent variable.

The first step, we will identify NULL values. We found 87 values with NULL tweets. We will drop these since the NULL values are across date ranges, so dropping them will not impact the model-based prediction.

Next, we will apply the Count Vectorizer

A screen shot of a computer code

Description automatically generated with low confidence

Next, the train and test data are split and then machine learning models are applied. This will be explained in detail in the next section.

The data preparation is an iterative process, when the accuracy was found to be low, feature engineering, by applying one-hot encoding is applied.

A screenshot of a computer program

Description automatically generated with low confidence

There are three extra fields that will be the target variable going forward.

Next, in data preparation the data needs to be prepared for the time series analysis. The goal is to predict the weekly, monthly, 3 months (quarterly) sentiments. We will hence create three datasets for the same, this will make easy to separate the data frame and the required calculations, hyper parameters etc to be used within that data frame and to suit the model. Also, these separate data frames can be used further at the end while creating dashboards. There will be multiple tweets per day, below screen shot shows randomly one tweet per day , since we will need to create a time series first, and the thumb rule in time series is it should be continuous.

A screenshot of a computer

Description automatically generated with medium confidence

There might be dates where there are no tweets, we will use the asfreq attribute of the data frame and use ffill method to fill the missing dates. ffill method seems to be the best method to use in this scenario, since most of the times, once there is a change in sentiment then for next few tweets the sentiment is unchanged.

Before ffill:

A screenshot of a computer

Description automatically generated with medium confidence

After ffill:

A screenshot of a computer

Description automatically generated with medium confidence

Then we have created separate data frames for weekly, monthly, and quarterly.

**Model Building and Evaluation:**

We will divide the model building and evaluation section into two sub-sections sentiment analysis and the second one for Time series prediction of sentiments.

## Sentiment Analysis:

Statistical analysis of the sentiments, like the days of maximum tweets, average number of tweets are available in the above section. In this section we will develop machine learning models for natural language processing (NLP) and then identify the best model to predict the sentiments.

1. Multinomial NB: Multinomial Naïve Bayes algorithm is selected as the first model. This is because it is the basic, powerful, and easy classification model used extensively in NLP. It is a probability-based model.

**P(A|B) = P(A) \* P(B|A)/P(B)**

P(B) = prior probability of B

P(A) = prior probability of class A

P(B|A) = occurrence of predictor B given class A probability.

A screenshot of a computer

Description automatically generated with medium confidence

The accuracy obtained is 81% Train and 71% Test, the model is known to give low accuracy than other models, hence we will test other models.

1. Neural Network for Classification problem:

A screenshot of a computer code

Description automatically generated with low confidence

We will create a three-layer neural network (when counting layers we don’t include the input layer because it does not have any parameters to learn) using Keras’ sequential model. Each layer is “dense” (also called fully connected), meaning that all the units in the previous layer are connected to all the neural in the next layer.

In the first hidden layer we set units=1000, meaning that layer contains 1000 units with ReLU activation functions: activation='relu'. Relu is the activation function to be used for classification model since it is will out the value as 0 or the input value depending on the input passed. The number of units are tuned and undergone iterative testing, best accuracy is obtained with units=1000 in the first and units=500 in the next two hidden layers. The output for a classification problem should use one unit and SoftMax activation. This is the frequently used activation function to normalize the output to a probability distribution.

The accuracy received was less 48%.

Applying one hot encoding to see if the performance improves:

Applying the same set up as above, just that the output units will be set to 3, since there will be three variables after one-hot encoding on the sentiments field.



Accuracy improved drastically with one-hot encoding:

A screenshot of a computer program

Description automatically generated with low confidence

A picture containing text, line, plot, diagram

Description automatically generated

From the above graph, test accuracy is consistent after 2 epoch , train accuracy is almost consistent after 4 epoch, however there seems to be a overfitting, test and train accuracy matches at near epoch 2.

### Hyper Parameter Tuning:

In the next step, we will apply hyper parameter tuning and improve the accuracy.

A screen shot of a computer

Description automatically generated with medium confidence

We will use GridSearchCV in the hyper tuning cross Validation. The train and test data are split with 30% test data. We will use inverse document frequency (Tfid Vectorizer) here, instead of count vectorizer which will take the raw text to create Bag of Words and apply inverse document frequency. Param grid is defined, these are hyper parameters that can be configured. We will use the pipeline module from the sklearn. pipeline, here we configure the tfidf vectorizer to be used in SVC (Simple vector classifier), this is again a powerful classifier algorithm available in sklearn library. We will measure accuracy as the scoring measure.

A picture containing text, screenshot, font, line

Description automatically generated

A screenshot of a computer program

Description automatically generated with low confidence

A screen shot of a computer

Description automatically generated with low confidence

There is an increase in the accuracy to 77%.It also shows the list of stopwords to be used for higher accuracy.

## Boosting Algorithm:

Boosting algorithms are used to reduce overfitting /underfitting/ bias of the model. Using one of the frequently applied models is lightgbm (gradient boosting model). LGBMclassifier method is used here .

A screenshot of a computer program

Description automatically generated with medium confidence

There is an increase in accuracy seen with the booster algorithm.

Summarizing the accuracy seen in all the above models in a table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Model** | **Train Accuracy** | **Test Accuracy** |
| Naïve Bayes Classification | MultinomialNB | 78% | 69% |
| Neural Network | Sequential (3 layers, relu activation for dense, 1 unit softmax activation for output layer) | 48% | 49% |
| Neural Network (with one hot encoding) | Sequential (3 layers, relu activation for dense, 3 units softmax activation for output layer) | 99% | 85% |
| Hyper Parameter Tuning(Grid Search CV) | Simple Vector Method(Classification) | 77.4% | 79% |
| Boosting Algorithm | Light GBM | 85.26% | 80.15% |

*Table 1: Summary table highlighting the model performance in sentiment Analysis*

## Time Series Forecasting:

TBC