CCT

MSc In Data Analytics CA2

Contents

[1 Executive Summary 3](#_Toc149852250)

[1.1 Introduction 3](#_Toc149852251)

[1.2 Dataset Description 3](#_Toc149852252)

[2 Data Storage and Cleaning 3](#_Toc149852253)

[2.1 Dataset Storage 3](#_Toc149852254)

[2.2 Data Cleaning 4](#_Toc149852255)

[3 Reference 6](#_Toc149852256)

[Annexture 7](#_Toc149852257)

# Executive Summary

## Introduction

This project is an integrated assessment used for analysing a large dataset derived from the Twitter API named “ProjectTweets.csv”. The dataset comprises od 1,600,00.0 tweets, and the primary objective was to investigate the sentiment shift over the recorded time period. A time series forecast was mandated fir projecting the sentiment across 1 week, 1 month, and 3 months into the future. The forecast results representation in a dynamic dashboard.

## Dataset Description

The Twitter dataset comprises of five key attributes: ”ids” (tweet ID), “date” (tweet date), “flag” (query status), “user” (tweeting user), and “text” (tweet content).

# Data Storage and Cleaning

## Dataset Storage

The dataset “ProjectTweets.csv” was stored into the Hadoop Distributed File System (HDFS), a reliable and scalable storage system designed to span across multiple clusters of commodity hardware (Databricks, 2023). The storage procedure followed these key steps:

* HDFS Initialization: Using the command “start-dfs.sh”, the HDFS was initialized. This command ensured all background processes have been successfully started and are currently active.
* YARN Initialization: The YARN (Yet Another Resource Negotiator) resource management platform was initiated using the “start-yarn.sh” command. YARN is very important for efficient resource allocation and task scheduling within Hadoop ecosystem.
* Dataset storage: The dataset was saved in HDFS using the command “-put” and using “user1” folder.
* Pyspark Integration: To facilitate data processing and analytics, Pyspark was used. Pyspark is an interface for Apache Spark in Python with powerful big data processing capabilities.
* Jupyter Notebook Access: The dataset analysis was executed using Jupyter notebook. By starting Pyspark, a Jupyter notebook was automatically launched and made accessible at the address “http://localhost:8889/notebooks/Downloads/Ca2.ipynb”

A screenshot of a computer

Description automatically generated

Figure 2.1: Jupyter Notebook Access

In summary, the dataset was securely stored in HDFS. YARN together with Pyspark were then used to establish a robust analytical environment through the Jupiter notebook interface.

## Data Cleaning

The initial row of the dataset served as the header, which was preserved and saved under the name "header\_row". Subsequently, it was added back into the dataset with the header names being updated to: "index", "user\_id", "timestamp", "query", "username", and "tweet\_text". In order for the header to be updated the “withColumnRenamed” Pyspark dataframe method was used (Spark, 2023).

The "query" column was removed as it didn’t hold any important information.

### Missing Data

Using pyspark sql functions “when”, “count” and “col” each column was checked for missing values. The output indicated that the dataset doesn’t have missing values (Naveen, 2022).

A screen shot of a computer

Description automatically generated.

Figure 2.2: Missing Data Output

### Tweet Text Cleaning

The tweet\_text column was cleaned using “withColumn” dataframe function (Zach, 2023). The following steps were taken:

* URL removal: All URLs starting with http:// or https:// were removed from the text.
* Hashtags removal: Hashtags, which are words starting with “#” were removed
* Special characters removal: Special characters including &,\*,%,$,~,@ and ! were removed. It was later decided to retain “!” and “?” because of their potential significance in sentiment analysis.

A screenshot of a computer

Description automatically generated

Figure 2.3: Tweet Text Cleaning

### Timestamp Transformation

The dataset contained timestamps in the format "Mon Apr 06 22:19:45 PDT 2009" (StringType). All timestamps were converted into a standard datetime format to support time series sentiment analysis. Due to the changes made in Spark 3.0 datetime parsing bahaviour, the parsing policy was changed to “LEGACY” (Datacamp, 2022) to ensure compatibility. Using PySpark's in-built datetime functions (“unix\_timestamp”, ”from\_unixtime”, “timestamptype”), the timestamps were transformed into the format "2009-04-07 06:19:45".

A white paper with black numbers and black text

Description automatically generated

Figure 2.4: Timestamp transformation

### Column Renaming

The “tweet\_text” column was dropped after creating the “cleaned\_text” column. The latter was renamed to “tweets”. The user\_”id column” was renamed to “userid”.

### Text Normalization

The “tweets” column was converted to lowercase to maintain consistency and reduce redundancy during sentiment analysis, as the same word could have been treated as distinct if written in uppercase or lowercase.

A screenshot of a computer code

Description automatically generated

Figure 2.5: Cleaned Dataset

### Removing Duplicate Rows

The removal of duplicate rows was achieved in Spark using the “distinct()” and the “dropDuplicates()” method. First the dataset was interrogated using the distinct().count() function. The results showed that the dataset has 1,600,000.0 of distinct rows, meaning that no duplicates were identified. A further analysis done specifically on the “username” and “tweets” columns revealed otherwise. After discarding the duplicates, 1,588,174 rows remained. To determine which rows were duplicated, the “subtract” function was utilized. A visualization of these rows was achieved using the “show” function. To validate the accuracy of this method, a selection of rows was randomly cross referenced against the original dataset. All spot checks confirmed that the method used was successful.

## Comparative Analysis Using YCSB Benchmarking Tool

First the cleaned dataset named “distinct\_rows\_based\_on\_colums” was saved in HDFS user 1 folder using the “write.csv” function using the path “/user1/cleaned\_ProjectTweet.csv”.

A screenshot of a computer screen

Description automatically generated

Figure 2.6: HDFS user 1 folder containing cleaned\_ProjectTweets.csv

Comparison quantitatively

I should do the test 3 times and average the results and then compare the results

I can check the time series performance

Casandra is used in Twitter production clusters

# Reference

# Annexture

A screenshot of a computer

Description automatically generated

Note: the total count includes section title, references etc.

Github project location: https://github.com/cuculicu/CA2-Second-semester