Principles of Big Data Management

COMP-SCI 5540 [Spring Semester 2023]

Project Title: Sentiment Analysis Using Big Data

Group Member:

Name	Student ID
Gaurang Solanki	16337161
Mohan Sai Singu	16351594
Supriya Reddy Jagganolla	16326696
Bhavana Gattuboyena	16331306

Abstract:

Sentiment Analysis is a technique that uses machine learning and natural language processing (NLP) to analyze and classify subjective data. This documentation provides an overview of a sentiment analysis system that uses big data to analyze and classify data. The document covers the project goals, objectives, scope, limitations, constraints, feasibility study, work breakdown structure, system requirement specifications, hardware requirements, system design, and architecture.

Introduction:

Sentiment analysis using big data is a technique that enables businesses to gain insights into their customer's opinions and attitudes towards their products, services, or brands. The goal of this project is to design and implement a sentiment analysis system that can process large volumes of data in real time.

Project Goals and Objectives:

The primary goal of this project is to design and implement a sentiment analysis system that can analyze and classify large volumes of data in real time. The objectives of this project are:

- To develop an automated system that can analyze and classify data from Twitter.
- To create an intuitive user interface that allows businesses to visualize and analyze customer sentiment data in real time.
- To implement a machine learning algorithm that can accurately classify customer sentiments.
- To develop a scalable system that can handle large volumes of data in real time.

Project Scope:

The scope of this project includes the development of a sentiment analysis system that can analyze and classify data from Twitter. The system will include a user interface that allows businesses to visualize and analyze customer sentiment data in real time.

Project Limitations and Constraints:

The accuracy of the sentiment analysis system is highly dependent on the quality and quantity of data. The system's accuracy may also be affected by the language and cultural differences in the data. The project may also be limited by hardware constraints, such as the processing power required to analyze large volumes of data. The Constraints of this project include the availability of data.

Feasibility Study:

The feasibility study for sentiment analysis using big data involves assessing data availability and technical feasibility.

Availability of Data: The availability of data is critical to the feasibility of sentiment analysis. There must be a sufficient volume of data for the system to provide accurate results. Data can be sourced from Twitter.

Technical Feasibility: The technical feasibility of the system is assessed based on the hardware and software requirements. Sentiment analysis requires a large processing capacity, and therefore, big data technologies such as Hadoop and Spark are necessary. The system must also be able to integrate with other systems and provide real-time analysis of data.

Work Breakdown Structure:

The work breakdown structure for this project is as follows:

- **Mohan Singu worked on Requirements Gathering**.
- **Supriya Reddy and Bhavana worked on Data Collection**: Tweepy is a package in Python that can connect to Twitter API V2.0 and extract real-time data from Twitter based on a query condition.
- **Supriya Reddy and Bhavana also worked on Data Storage**: The Hadoop Distributed File System is used by companies and applications as a data storage system to store big data. the HDFS library for Python API is used. This library contains functions to establish connections with the Hadoop cluster, push and pull the data into HDFS, etc.
- **Gaurang Solanki and Mohan Singu worked on Data Cleaning and Pre-processing**: Apache Spark is a distributed computing framework that is open source. PySpark is a Python API for Apache Spark with libraries for real-time and large-scale data processing.
- **Gaurang Solanki worked on Data Visualization**: Plotly is an open-source, interactive, and browser-based graphing library for data visualization with various functions using which we can plot different graphs making the data easier to understand.

Hardware Requirements:

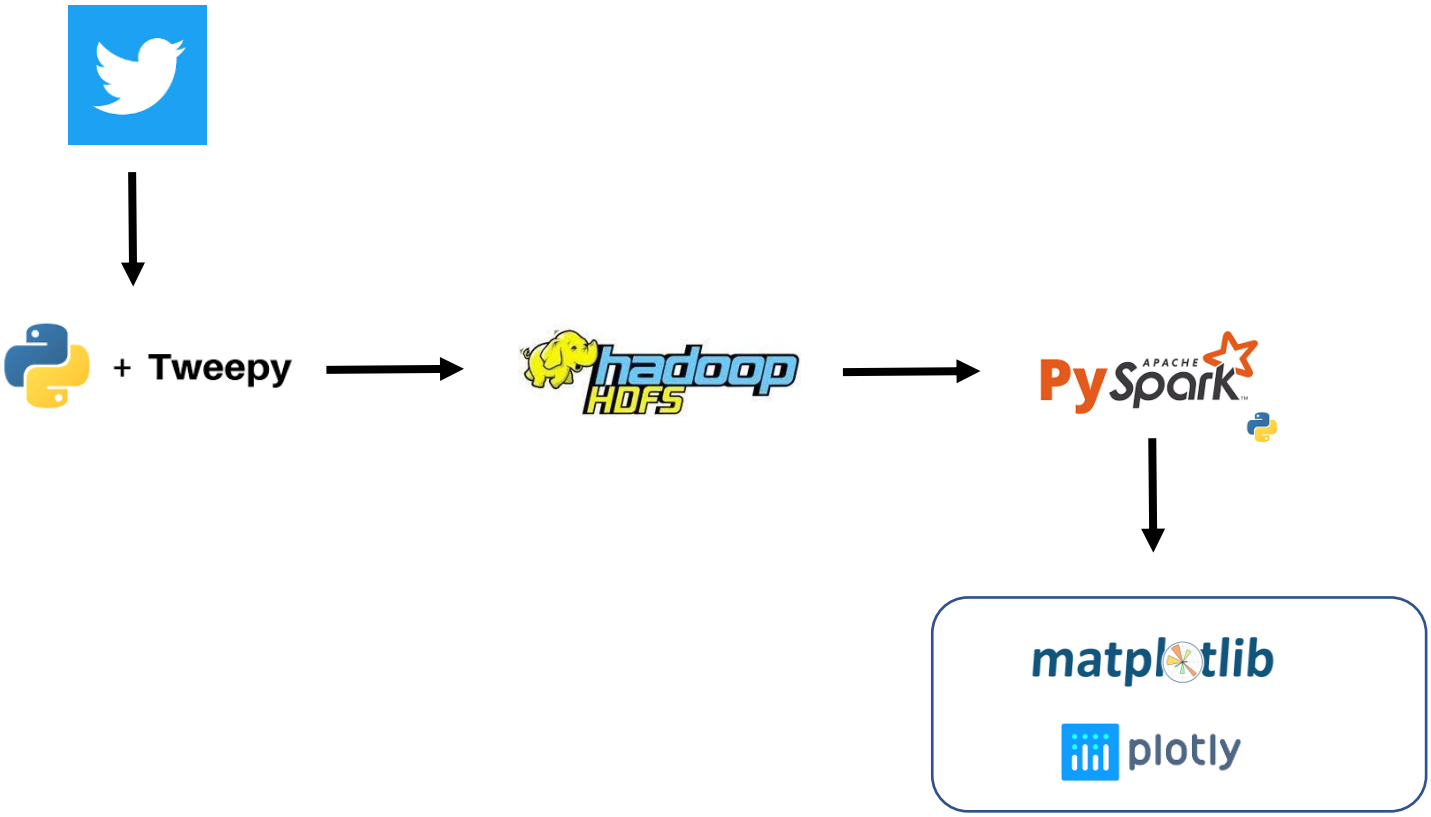
The hardware requirements for the sentiment analysis system are as follows:

- A decent-performance system with a minimum of 4GB RAM.
- Sufficient storage capacity to store large volumes of data.
- High-speed internet connectivity to ensure real-time data processing.

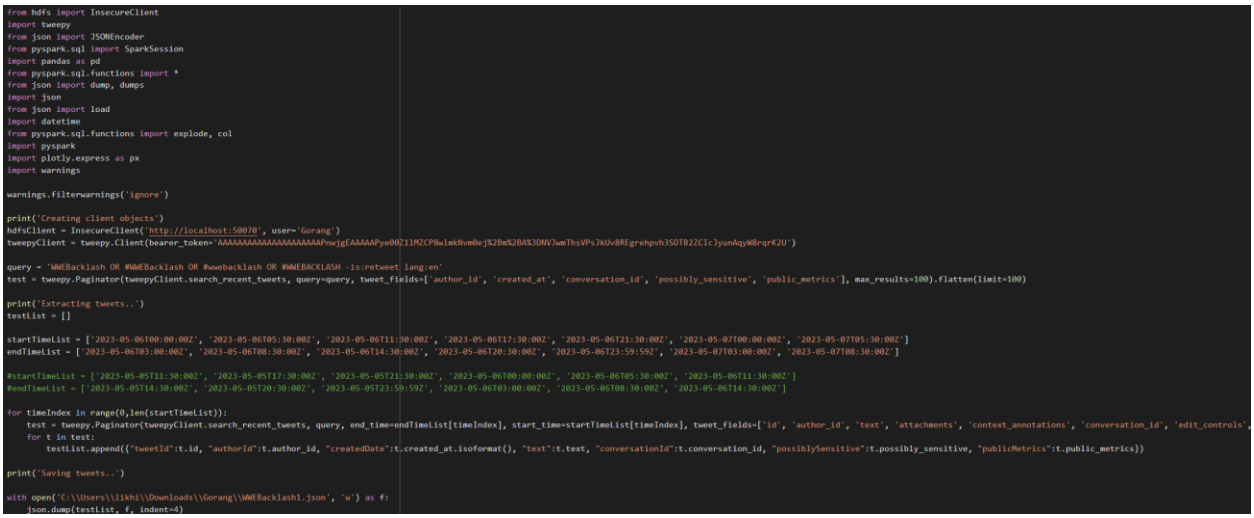
Architectural Diagram:

The system will consist of the following components:

1. Data Ingestion – Tweepy
2. Storing Data - HDFS.
3. Data Processing – PySpark
4. Data Visualization - Plotly
5. Sentiment Analysis – VARER



Code:



```
from hdfs import InsecureClient
import tweepy
from json import JSONEncoder
from pyspark.sql import SparkSession
import pandas as pd
from pyspark.sql.functions import *
from json import dump, dumps
import json
from json import load
import datetime
from pyspark.sql.functions import explode, col
import pyspark
import plotly.express as px
import warnings

warnings.filterwarnings('ignore')

print('Creating client objects')
hdfsClient = InsecureClient('http://localhost:50070', user='Gorang')
tweepyClient = tweepy.Client(bearer_token='AAAAAAAAAAAAAAAAAAAAAPmwjgEAAAAAPye00Z11MZCP8wlmkNvm0ej%2Bm%2BA%3DNVJwmThsVPsJkUv8REgrehpvh3SDTB2ZCIcJyunAqyW8rqrK2U')

query = 'WWEBacklash OR #WWEBacklash OR #wwebacklash OR #WWEBACKLASH -is:retweet lang:en'
test = tweepy.Paginator(tweepyClient.search_recent_tweets, query=query, tweet_fields=['author_id', 'created_at', 'conversation_id', 'possibly_sensitive', 'public_metrics'], max_results=100).flatten(limit=100)

print('Extracting tweets..')
testList = []

startTimeList = ['2023-05-06T00:00:00Z', '2023-05-06T05:30:00Z', '2023-05-06T11:30:00Z', '2023-05-06T17:30:00Z', '2023-05-06T21:30:00Z', '2023-05-07T00:00:00Z', '2023-05-07T05:30:00Z']
endTimeList = ['2023-05-06T03:00:00Z', '2023-05-06T08:30:00Z', '2023-05-06T14:30:00Z', '2023-05-06T20:30:00Z', '2023-05-06T23:59:59Z', '2023-05-07T03:00:00Z', '2023-05-07T08:30:00Z']

#startTimeList = ['2023-05-05T11:30:00Z', '2023-05-05T17:30:00Z', '2023-05-05T21:30:00Z', '2023-05-06T00:00:00Z', '2023-05-06T05:30:00Z', '2023-05-06T11:30:00Z']
#endTimeList = ['2023-05-05T14:30:00Z', '2023-05-05T20:30:00Z', '2023-05-05T23:59:59Z', '2023-05-06T03:00:00Z', '2023-05-06T08:30:00Z', '2023-05-06T14:30:00Z']

for timeIndex in range(0,len(startTimeList)):
    test = tweepy.Paginator(tweepyClient.search_recent_tweets, query, end_time=endTimeList[timeIndex], start_time=startTimeList[timeIndex], tweet_fields=['id', 'author_id', 'text', 'attachments', 'context_annotations', 'conversation_id', 'edit_controls',
    for t in test:
        testList.append({"tweetId":t.id, "authorId":t.author_id, "createdDate":t.created_at.isoformat(), "text":t.text, "conversationId":t.conversation_id, "possiblySensitive":t.possibly_sensitive, "publicMetrics":t.public_metrics})

print('Saving tweets..')

with open('C:\\Users\\likhi\\Downloads\\Gorang\\WWEBacklash1.json', 'w') as f:
    json.dump(testList, f, indent=4)
```

Import required libraries such as InsecureClient from hdfs, tweepy, JSONEncoder, SparkSession, pandas, dump, dumps, load, datetime, explode, col, pyspark, plotly.express, and warnings. then create client objects for Hadoop distributed file system (HDFS) and Twitter API using the InsecureClient and tweepy.Client, respectively.

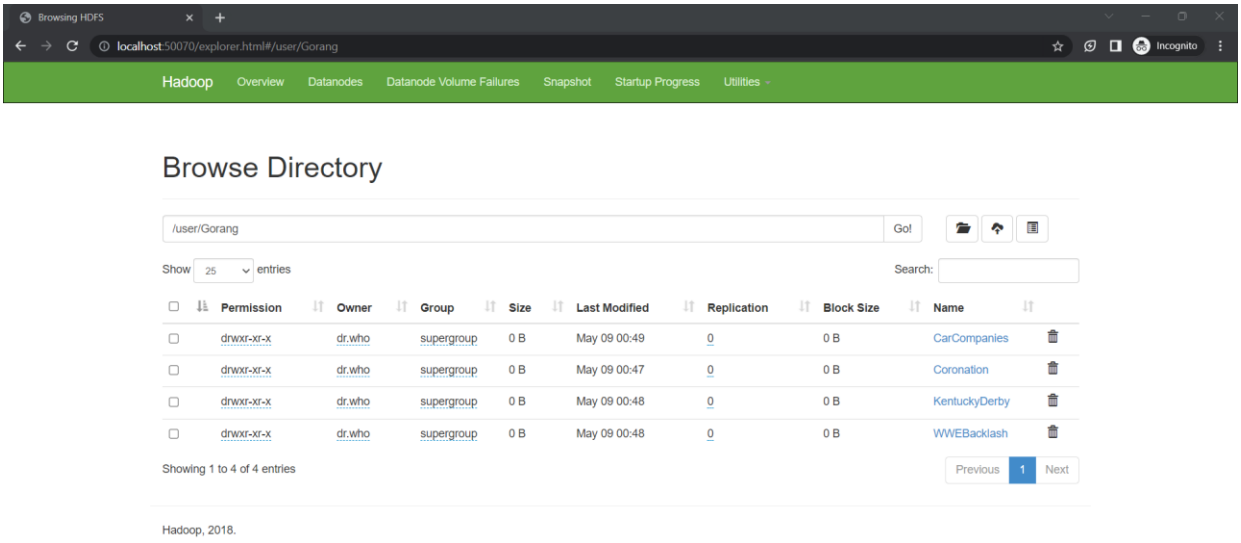
The code then defines a Twitter query for retrieving recent tweets related to **#WWEBacklash or #WWEBacklash or #wwebacklash or #WWEBACKLASH** that are in English and are not retweets. It then retrieves the tweets using the tweepy.Paginator and stores them in a list called testList. The search parameters include tweet_fields such as id, author_id, text, attachments, context_annotations, conversation_id, edit_controls, entities, possibly_sensitive, public_metrics, referenced_tweets, withheld, created_at, and geo, and max_results parameter is set to 100.

The code also defines two lists startTimeList and endTimeList, which define the start and end times for retrieving the tweets. For each time interval defined in these lists, the code retrieves the tweets by calling the tweepy.Paginator again and appends them to the testList. Finally, save the list of tweets in a JSON file named WWEBacklash1.json using the JSON.dump function. The JSON file is saved in the specified file path C:\\Users\\likhi\\Downloads\\Gorang\\.

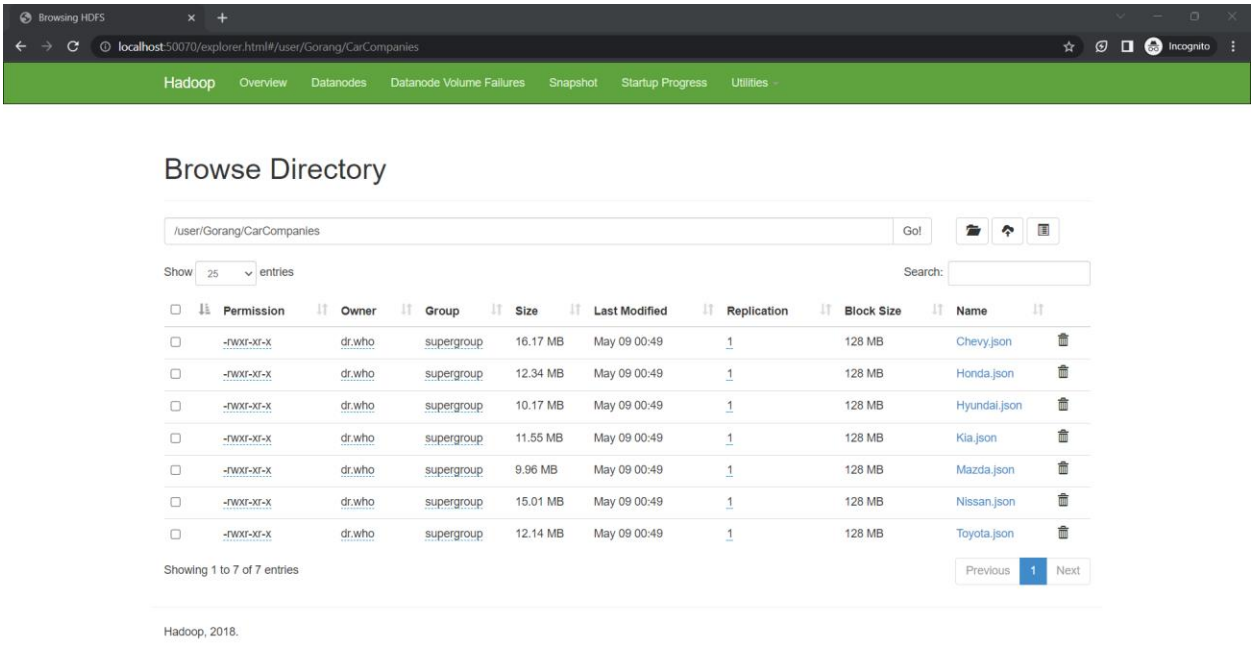
Then, Change the tweet hashtag from #WWEBacklash to #Coronation and #KentuckyDerby as these hashtags are trending at this point.

Also, we did the same thing to extract tweets related to Car Companies.

- Data will stored within respective folders



- Tweets from car companies within respective folders.




```
with hdfsClient.read(filePath, encoding = 'utf-8') as reader:
    model = load(reader)

df = pd.json_normalize(model)
df.rename(columns = {'publicMetrics.retweet_count':'retweetCount', 'publicMetrics.reply_count':'replyCount', 'publicMetrics.like_count':'likeCount', 'publicMetrics.quote_count':'quoteCount'}, inplace = True)
df['text'] = df['text'].astype('string')
df['createdDate'] = pd.to_datetime(df['createdDate'])

print('Building Spark Session')
spark = SparkSession.builder \
                    .master('local[1]') \
                    .appName('SparkByExamples.com') \
                    .getOrCreate()

print('Building Spark Dataframe')
dfsp = spark.createDataFrame(df)

df2 = dfsp.groupBy("possiblySensitive").count()
```

The above code is loading or pulling tweets from HDFS (Hadoop Distributed File System), then normalizing and transforming the data before building a Spark session and dataframe. The dataframe is then used to group the data by the 'possiblySensitive' column and count the number of occurrences of each value. Finally, the resulting dataframe is converted to a Pandas dataframe

The specific steps in the code are as follows:

- Load or pull tweets from HDFS using the hdfsClient.read() method and the specified file path.
- Normalize the loaded data into a Pandas dataframe using the pd.json_normalize() method.
- Rename columns of the Pandas dataframe using the df.rename() method to make them more readable.
- Convert the 'text' column to string data type using the astype() method and convert the 'createdDate' column to datetime format using the pd.to_datetime() method.
- Build a Spark session using the SparkSession.builder() method with the specified configurations.
- Create a Spark dataframe from the Pandas dataframe using the spark.createDataFrame() method.
- Group the Spark dataframe by the 'possiblySensitive' column and count the number of occurrences of each value using the dfsp.groupBy().count() method.
- Convert the resulting Spark dataframe to a Pandas dataframe using the toPandas() method.

Then, we used that Data frame for Analysis and visualization.

Analysis and Visualization:

- Most Liked Coronation Tweet:

tweetId	authorId	createdDate	text	likeCount
1654492609652445200	36042554	2023-05-05T09:25:51	So pleased to see so many of you looking forward to the Coronation weekend! 👋 https://t.co/4u8SL3AEfK	21506

- Most Liked WWE Backlash tweet:

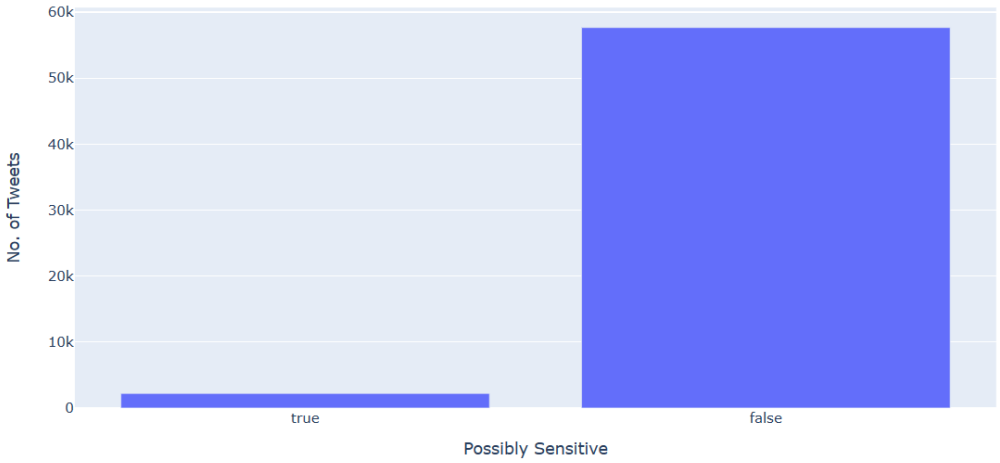
tweetId	authorId	createdDate	text	likeCount
1654996610517016600	1327630782035341300	2023-05-06T18:48:34	WADE?! #WWEBacklash https://t.co/fGJlIrhTMS	18242

- Most Liked Kentucky Derby tweet:

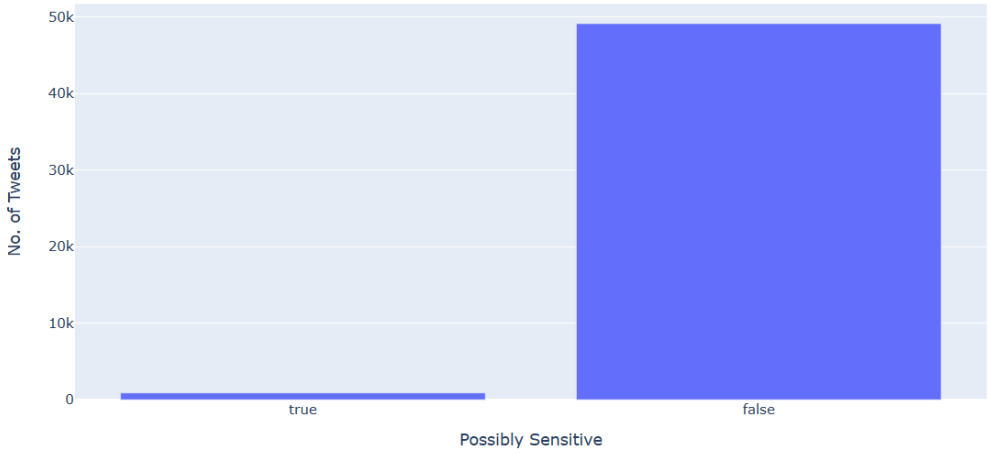
tweetId	authorId	createdDate	text	likeCount
1654989894261919700	25742569	2023-05-06T18:21:53	MAGE pulls off a magical finish to win the 149th Kentucky Derby. ⏳ https://t.co/wIjVzaMccs	8077

- Number of tweets which are sensitive (Controversial)

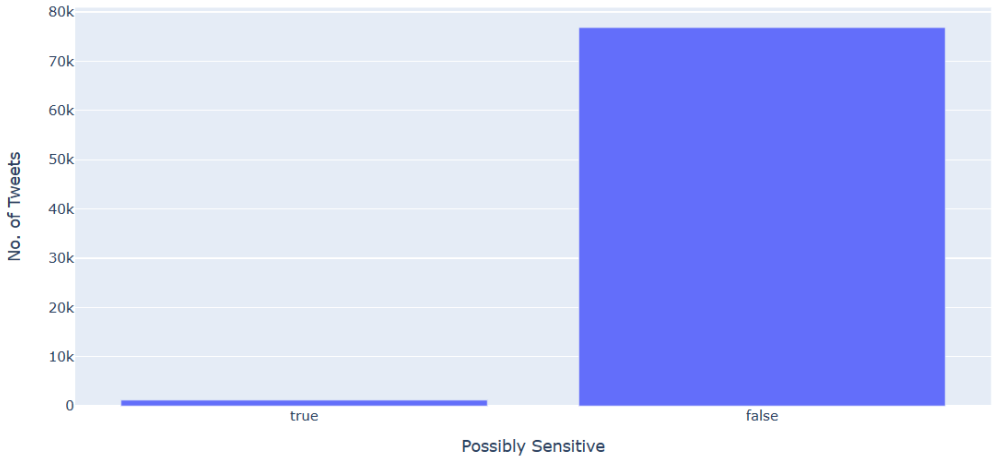
Possibly Sensitive Tweet Count: #WWEBacklash



Possibly Sensitive Tweet Count: #KentuckyDerby



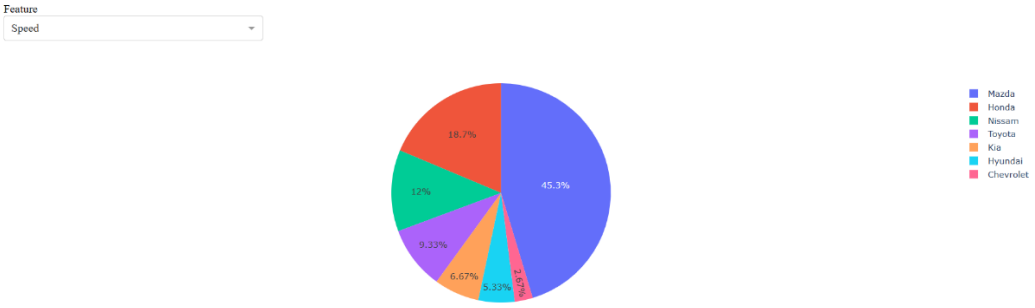
Possibly Sensitive Tweet Count: #Coronation



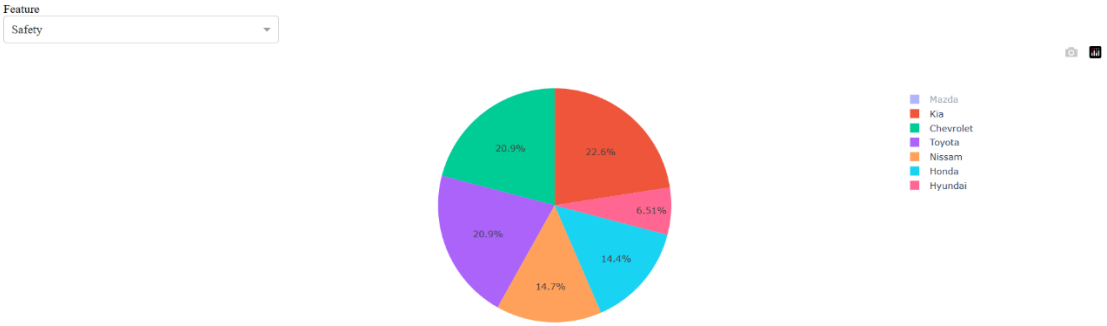
Extracted tweets related to different car companies:

- Number of tweets posted by different car companies

- Speed of cars related tweets by different companies



- Safety of the cars related tweets



Future scope:

Once the data has been pre-processed, it can be used for multiple purposes. One of the most common uses of pre-processed data is for classification problems. In classification, the goal is to assign a label or category to each data point based on its features. Pre-processed data can be fed into machine learning or deep learning models to classify new data based on previously learned patterns.

Pre-processed data can also be used for Natural Language Processing (NLP). NLP is a field of study that focuses on the interactions between humans and computers using natural language. NLP tasks include text classification and language translation. Pre-processed data can be fed into NLP models to perform these tasks more accurately and efficiently.

Reference:

1] K. Shvachko, et al., "The Hadoop Distributed File System," IEEE 26th Symposium on Mass Storage Systems and Technologies, pp. 1-10, 2010.

[2] C. Kaushal and D. Koundal, "Recent Trends in Big Data using Hadoop," International Journal of Informatics and Communication Technology, vol. 8, pp. 39-49, 2019.

[3] M. Wankhede, et al., "Analysis of Social Data Using Hadoop Ecosystem," International Journal of Computer Science and Information Technologies, vol. 7, pp. 2402-2404, 2016.

[4] P. Ganesh, et al., "Performance Evaluation of Cloud service with Hadoop for Twitter Data," Indonesian Journal of Electrical Engineering and Computer Science, vol. 13, pp. 392-404, 2019.

[5] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Communications of the ACM, vol. 51, pp. 107-113, 2008.]

[6] Sheela, L. J. (2016). "A review of sentiment analysis in Twitter data using Hadoop". International Journal of Database Theory and Application, 9(1), 77–86. doi:10.14257/ijdta

[7] D.N. Biju and Y. Arora, Twitter Data Analysis using Hadoop. International Journal of Advance Research and Innovation Ideas in Education, vol. 4(5), 2018.

[8] S. Wilson and S.R, "Twitter data analysis using Hadoop ecosystem and Apache zeppelin", Indonesian Journal of Electrical Engineering an Computer Science, Vol. 16(3), pp. 1490~1498, 2019.

[9] Anisha P. Rodrigues & Niranjan N. Chiplunkar, 2018, "Real-time Twitter data analysis using Hadoop ecosystem", Cogent Engineering, 5:1, 1534519, DOI: 10.1080/23311916.2018.1534519

[10] Tare, M., Gohokar, I., Sable, J., Paratwar, D., & Wajgi, R. (2014). Multi-class tweet categorization using map-reduce paradigm. International Journal of Computer Trends and Technology (IJCTT), 9(2), 78–81. doi:10.14445/22312803/IJCTT-V9P117

[11] Michal Skuza and Andrzej Romanowski., 2015, "Sentiment study of Twitter Data within Big Data Distributed Environment for Stock Prediction," in Computer Science and Information Systems (FedCSIS),

Federated Conference on, pp. 1349–1354.

- https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoopcommon/ClusterSetup.html
- https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoopcommon/SingleCluster.html
- https://docs.tweepy.org/en/stable/
- https://twittercommunity.com/t/problem-with-apache-flume-and-twitterapi/175288
- https://spark.apache.org/docs/latest/api/python/
- https://matplotlib.org/stable/index.html
- https://pypi.org/project/hdfs/
- https://pypi.org/project/vaderSentiment/