Machine learning based Sentimental analysis of Covid-19 disease related Twitter data using Py-Spark

A PROJECT REPORT

Submitted by : **Group D19**

BL.EN.U4AIE21044 D. Mohammad Abdulla

BL.EN.U4AIE21048 G. Tej Deep Reddy

BL.EN.U4AIE21050 G. Mukesh Venkata Sai

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (Artificial Intelligence)

AMRITA SCHOOL OF COMPUTING, BANGALORE

AMRITA VISHWA VIDYAPEETHAM

BANGALORE 560 035

June - 2023

ABSTRACT

Sentiment analysis on Twitter data has recently received a lot of attention. Recent years have seen a substantial increase in interest in sentiment analysis of Twitter data. Due to the enormous volume of tweets that are produced every day, Twitter is a useful source of textual information, providing one of the most comprehensive collections of data. As a result, Twitter is the perfect venue for gathering a wide range of opinions on many subjects because individuals tend to express their thoughts openly there. However, it becomes impractical for people to read and analyses every tweet manually due to the massive volume that is generated. As a result, in order to filter and extract insights from this plethora of data effectively, an automated decision-making technique is required. For sentiment analysis in this project, we recommend using the PySpark framework and Machine Learning. Machine learning can be analysed utilizing a variety of analytical techniques. Data analysis is one of the many fields where machine learning techniques are quite important. In this study, the Twitter analysis is predicted using machine learning approaches. For implementation, machine learning techniques such as K-Nearest Neighbors Algorithm, Logistic Regression, Decision Tree and Support Vector Machine are utilized. On the tweets acquired from social media, the performance of the implemented algorithm is evaluated in terms of accuracy, precision, recall, and F1-score. The widely used open-source distributed processing platform Apache Spark has a Python interface called PySpark that makes use of distributed memory abstraction. We aim to efficiently handle massive datasets and carry out complex data processing tasks in a distributed and parallel fashion using PySpark. The primary objective of this work is to develop a sentiment analysis model that can accurately classify the sentiment expressed in tweets. We can process huge amounts of Twitter data in a scalable and effective way by utilizing PySpark's strength and its distributed computing capabilities. The Mechanism gives whether the result is positive, negative. It provides more accurate results, keeps track of users, and indicates whether the user's material has an upward or downward effect on the tweet. As a result, we can analyse opinions on a range of subjects, including politics, entertainment, and product reviews, and gain important insights from the viewpoints expressed on Twitter.

TABLE OF CONTENTS

Page no

CHAPTER - 1
1. INTRODUCTION

1.1. IMPORTANCE OF SENTIMENTAL ANALYSIS

The internet's reach and scope have grown quickly, especially thanks to social media and microblogging websites like Facebook, Twitter, and Tumblr. These platforms are the global leaders in the dissemination of succinct news and trending topics. When more users voice their opinions and assessments, a topic becomes popular and becomes a valuable source of online perception. Trending topics are frequently used to spread awareness, support political candidates and public figures, endorse goods, and provide entertainment through movies and award shows. Companies and organizations use customer feedback to enhance their services and products, which strengthens marketing strategies. For instance, leaking images of an upcoming iPhone before it is released builds anticipation and appeals to consumers' emotions. Consequently, there is vast potential in discovering and analyzing interesting patterns from the abundant social media data for business-driven applications.

Twitter is a social networking website where users post 140-character tweets. At the current rate of 6500 tweets per second, there are roughly 561.6 million tweets posted each day. These noisy streams of tweets reflect multiple topics and ever-evolving attitudes in an unfiltered and unstructured manner. Natural language processing methods are used in Twitter sentiment analysis to extract, recognize, and characterize the sentiment content. It is possible to conduct coarse and fine levels of sentiment analysis. While the fine level focuses on specific attributes, the coarse level analyses entire documents. The text contains both direct and comparative sentiments. Comparative sentiments involve comparing objects within the same sentence, while direct sentiments treat objects independently in the same sentence.

Sentiment analysis is the process of identifying and predicting emotions in words, sentences, or groups of documents. It functions as a tool for deciphering the attitudes, beliefs, and feelings

expressed in online mentions. The main objective is to get a general sense of how people feel about various issues. Conversations are categorized into positive, negative, or neutral categories using sentiment analysis. Social media platforms are widely used by people to network and keep up with news and events. People can express their opinions on websites like Twitter, Facebook, Instagram, and Google+. For instance, people frequently post their movie reviews online and participate in comment threads where they debate the acting prowess of the actors. Such data serves as the foundation for evaluating and rating not only films but also other products, giving perspectives on their likelihood of success. These websites' vast resources can be used for social studies and marketing. As a result, sentiment analysis has many uses, such as influence analysis, polarity classification, and emotion mining.

However, due to numerous difficulties, including tonality, polarity, lexicon, and grammar, analyzing tweets is not a simple task. Since tweets frequently lack grammar and structure, it can be challenging to decipher their meaning. Additionally, frequent use of slang words, acronyms, and unfamiliar terms is typical of online tweeting. For natural language processors, categorising such words based on polarity presents difficulties. In this project, sentiment from real-time, high-velocity tweets is analysed using PySpark's quick processing capabilities. We also used Machine learning algorithms such as Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, and Logistic Regression will be used to classify tweets based on sentiment towards different aspects of the product or service. The model's performance will be evaluated using metrics such as accuracy, precision, recall, and F1-score. The main objective of this project is to develop a model that accurately classifies tweets based on sentiment towards different aspects, providing valuable insights into user opinions regarding a particular product or service.

1.2. MOTIVATION

Due to its capacity to uncover information and comprehend the feelings and ideas revealed in textual data, sentiment analysis is extremely significant in a variety of sectors. Here are a few main arguments supporting the significance of sentiment analysis:

a. Understanding Public Opinion: Sentiment analysis aids businesses and organisations in determining how the general public feels about their brands, goods, or services. Businesses may determine if a post is favourable or bad and learn more about the public's perception by examining consumer reviews, feedback, and social media posts. Making educated company decisions and improving consumer happiness may both benefit from this knowledge.

b. Reputation management: Businesses may proactively manage their reputation by tracking and analysing sentiment. Organisations may respond appropriately to consumer concerns, fix problems, and uphold a positive brand image by recognising negative sentiment early on. Companies may monitor public opinion, spot potential catastrophes, and take swift action to safeguard their reputations by using sentiment analysis.

c. Analysis of Customer Feedback: Sentiment analysis is essential for analysing customer feedback. It assists companies in determining customer satisfaction levels, pinpointing areas for development, and implementing the required steps to improve the customer experience. Organisations may acquire important insights to direct their product development, marketing plans, and customer service initiatives by analysing client attitudes expressed in surveys, reviews, or social media posts.

d. Market Research and Competitor Analysis:Sentiment analysis is a means of gathering real-time market knowledge, according to market research and competitor analysis. Businesses may learn customer preferences, spot new trends, and evaluate the market by keeping an eye on social media platforms and analysing sentiment. Making educated product decisions, creating focused marketing initiatives, and staying one step ahead of the competition are all possible using this data.

e. Political Analysis and Public Opinion Tracking: In the political sphere, sentiment analysis is used to monitor public opinion, assess the effectiveness of political campaigns, and comprehend voter attitude. It makes it possible for political parties and politicians to gauge their support, pinpoint important topics, and adjust their messaging accordingly. Political analysts can learn more about how the public feels about politicians and policies by examining comments made on social media and in news stories.

f. Brand Monitoring and Social Media Marketing: Sentiment analysis supports brand monitoring and social media marketing by assisting companies in keeping track of their online reputation. Organisations may better understand how their brand is regarded, assess the success of marketing efforts, and interact with their audience by examining the attitudes expressed in tweets, posts, and comments.

Overall, sentiment analysis provides valuable insights into the emotions, opinions, and attitudes expressed in textual data. It empowers businesses, organizations, and researchers to make data-driven decisions, improve customer satisfaction, manage reputation, and gain a competitive edge in the market.

CHAPTER - 2
LITERATURE SURVEY

The paper titled "Sentimental Analysis on Twitter Data using Supervised Algorithms" presented at the 2023 ICCMC conference explores the application of supervised machine learning algorithms for sentiment analysis on Twitter data. The authors conduct experiments using various supervised algorithms such as Support Vector Machine (SVM), Logistic Regression (LR), and Naive Bayes (NB) algorithm are used for implementation. and evaluate their performance in terms of accuracy, precision, recall, and other metrics. They preprocess the Twitter data by removing noise, tokenizing, and extracting relevant features. The research findings provide insights into the effectiveness of the chosen algorithms for sentiment analysis on Twitter, offering valuable contributions to the field.[1]

This paper titled Sentiment Analysis on Twitter Data: Comparative Study on Different Approaches discusses about the popularity of social media as a platform for expressing opinions and sharing information. Specifically, it highlights Twitter as a valuable source for opinion mining and sentiment analysis due to its widespread usage. The research focuses on applying different machine learning algorithms, namely Support Vector Machines (SVM) with various kernel functions (linear, radial basis, polynomial, and sigmoid) and the Adaboost ensemble method, for sentiment classification. Feature extraction techniques such as Chi-square and CPD are employed. This literature survey suggests that SVM with the sigmoid kernel and Chi-square n-grams is the most effective model for sentiment classification in the context of the study.[2]

The paper titled Twitter Sentiment Analysis focuses on sentiment analysis of tweets on Twitter. It emphasizes the significance of extracting sentiments from tweets and classifying them into positive, negative, or neutral categories. The study suggests that such an approach can be beneficial for organizations mentioned or tagged in tweets. The unstructured format of tweets requires preprocessing to convert them into a structured format. The paper utilizes libraries and the Twitter API for accessing tweets, and the datasets are trained using algorithms. The two mentioned algorithms for sentiment analysis are K-Nearest Neighbors (KNN) and Naive Bayes. The keywords associated with this research include Sentiment Analysis, social media, Tweets, Tweepy, Pandas, Dataset, KNN, and Naive Bayes.

CHAPTER - 3

REQUIREMENTS AND ANALYSIS

3.1 SOFTWARE REQUIRMENTS

To perform Twitter sentiment analysis using PySpark, you would need the following software requirements:

I. Python: Install Python, preferably version 3.x, on your system. PySpark is compatible with Python, and you'll need Python to write and execute your code.

II. Apache Spark: PySpark is the Python library for Apache Spark, a powerful distributed computing framework. Install Apache Spark on your system to utilize its capabilities for large-scale data processing.

III. PySpark: Install PySpark library, which provides the Python API for interacting with Spark. You can install PySpark using pip, the Python package manager, by running the command pip install pyspark.
PySpark is the Python library for Apache Spark, a fast and general-purpose distributed computing framework. PySpark allows you to write Spark applications using Python, enabling you to leverage the capabilities of Spark for large-scale data processing, machine learning, and analytics.

IV. Java Development Kit (JDK): Apache Spark runs on Java, so you'll need to have the Java Development Kit (JDK) installed on your system. Make sure to set the Java environment variables properly.

V. Twitter Developer Account: Create a Twitter developer account and obtain API keys, tokens, and secrets. These credentials are required to authenticate and access the Twitter API for data collection.

VI. Datasets: A number of pre-assembled datasets are available for sentiment analysis on Twitter. You may train and test your sentiment analysis model using these datasets, which are labelled with the sentiment polarity (positive, negative, or neutral). Sentiment140 dataset, which contains 1.6 million tweets as an example. Dataset can be obtained from kaggle.

VII. Tweepy: Tweepy is a Python library that simplifies the process of accessing the Twitter API. Install Tweepy using pip (pip install tweepy) to easily fetch tweets and interact with the Twitter API.

VIII. Natural Language Toolkit (NLTK): NLTK is a popular Python library for natural language processing. Install NLTK (pip install nltk) to leverage its functionalities for text preprocessing, tokenization, and sentiment analysis.

IX. TextBlob: TextBlob is a Python library built on top of NLTK that provides simple and intuitive APIs for text processing tasks, including sentiment analysis. Install TextBlob (pip install textblob) to utilize its sentiment analysis capabilities.

X. NumPy (import numpy as np): NumPy is a fundamental library for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

XI. Pandas (import pandas as pd): Pandas is a powerful library for data manipulation and analysis. It provides data structures like DataFrames, which allow you to efficiently work with structured data. Pandas also offers functions for data cleaning, transformation, and aggregation.

XII. Matplotlib (import matplotlib.pyplot as plt): Matplotlib is a popular data visualization library in Python. It provides a wide range of functions to create different types of plots, such as line plots, bar plots, histograms, scatter plots, etc. The pyplot module within Matplotlib provides a simple interface for creating these plots.

XIII. Seaborn (import seaborn as sns): Seaborn is another data visualization library that works on top of Matplotlib. It offers a higher-level interface for creating attractive and informative statistical graphics. Seaborn simplifies the creation of complex visualizations and provides functions for statistical estimation and color palettes.

XIV. String (import string): The string module is a part of the Python standard library. It provides a set of commonly used string operations and constants, such as punctuation characters, whitespace characters, and ASCII letters. This module can be useful for text preprocessing and manipulation tasks.

XV. WordCloud (from wordcloud import WordCloud): The WordCloud library allows you to create visually appealing word clouds from text data. Word clouds represent the frequency of words in a text corpus, with more frequent words appearing larger in the cloud. It is often used to visualize word frequency or importance in text analysis.

3.2 DATA COLLECTION AND PREPARATION

1. Set up Twitter API access:
    - Create a Twitter developer account and obtain API keys, tokens, and secrets.
    - These credentials are required to authenticate and access the Twitter API.
2. Choose the data collection method:
    - Real-time data collection: Use the Twitter API to collect real-time tweets based on search queries, user timelines, or trending topics. You can use libraries like Tweepy to simplify the process.
    - Pre-compiled datasets: Alternatively, you can use pre-compiled datasets available from sources like Sentiment140 or academic which is available in kaggle
3. Data preprocessing:
    - Remove unwanted elements: Remove elements like URLs, user mentions, special characters, and emojis from the tweet text. These elements are not essential for sentiment analysis and can introduce noise.
    - Tokenization: Split the tweet text into individual words or tokens. This step enables further analysis and processing.

- Lowercasing: Convert all the tokens to lowercase. This ensures that words with different capitalizations are treated as the same.

4. Data cleaning and filtering:
    - Remove duplicates: Check for and remove any duplicate tweets from the dataset.
    - Handle missing values: Address any missing or null values in the dataset. You can either remove the incomplete data or impute values using appropriate techniques.
    - Filter irrelevant tweets: If necessary, filter out tweets that are not relevant to your analysis based on specific criteria (e.g., language, location, or topic).
5. Dataset splitting:
    - Split the dataset into training, validation, and test sets. The training set is used to train your sentiment analysis model and the test set is used to evaluate the final performance of your model.
6. Data exploration:
    - Conduct exploratory data analysis to gain insights into the dataset. This may involve visualizing the distribution of sentiment labels, examining word frequencies, or analyzing patterns in the data.

3.3 EXPLORATION AND ANALYSIS

1. Exploratory Data Analysis (EDA):
    - Visualize the distribution of sentiment labels: Plot the number or proportion of tweets in each sentiment category (positive, negative, neutral) to understand the overall sentiment distribution in the dataset.
    - Analyze tweet length: Explore the distribution of tweet lengths (number of characters or words) to identify any patterns or correlations with sentiment.
2. Text Preprocessing:
    - Tokenization: Split the tweet text into individual words or tokens.

- Stop word removal: Remove commonly occurring words (e.g., "the", "and", "is") that do not carry much sentiment information.
- Stemming/Lemmatization: Reduce words to their base or root form to handle variations (e.g., "running," "ran" -> "run").

3. Feature Extraction:
    - Bag-of-Words (BoW): Convert the preprocessed tweets into numerical feature vectors, representing the occurrence or frequency of words.
    - TF-IDF (Term Frequency-Inverse Document Frequency): Calculate the importance of words in a tweet relative to the entire dataset to capture their discriminative power.
4. Sentiment Analysis Techniques:
    - Machine Learning Algorithms: Apply classification algorithms such as logistic regression, support vector machines (SVM), or KNN to train a sentiment analysis model using the labeled dataset.
5. Evaluation Metrics:
    - Accuracy: Calculate the overall accuracy of the sentiment analysis model by comparing predicted sentiment labels with the true labels.
    - Precision, Recall, F1-Score: Evaluate the performance of the model using these metrics, especially if the dataset is imbalanced or if different sentiment categories have different levels of importance.
6. Visualizations:
    - Word Clouds: Generate word clouds to visualize the most frequent or important words associated with each sentiment category.
    - Sentiment Trends: Plot sentiment trends over time to identify any patterns or changes in sentiment.

3.4 CHALLENGES

1. Handling Large-Scale Data:
    - Twitter generates a massive volume of data, and processing and analyzing it can be computationally intensive. Efficient handling of

large-scale data may require distributed computing frameworks like PySpark.

2. Data Imbalance:
    - Sentiment datasets collected from Twitter often suffer from class imbalance, where one sentiment class (e.g., neutral) dominates the dataset compared to others (positive or negative). This can impact the model's ability to learn effectively.
3. Language and Cultural Nuances:
    - Sentiment analysis models need to account for language-specific nuances, idioms, and cultural references that influence sentiment. These factors can pose challenges when analyzing tweets in different languages or from diverse cultural backgrounds.
4. Subjectivity and Context Dependency:
    - Sentiment analysis is subjective, as sentiment interpretation can vary among individuals. Context plays a crucial role in sentiment determination, and capturing contextual information from short tweets can be challenging.
5. Generalization and Model Performance:
    - Developing a sentiment analysis model that generalizes well to unseen data and different domains is a challenge. The performance of the model can be influenced by the quality of training data, feature representation, and choice of algorithms.
6. Ethical Considerations:
    - Twitter sentiment analysis projects must adhere to ethical guidelines and data usage policies. Ensuring privacy protection, respecting user consent, and avoiding biases in the analysis are essential considerations.

CHAPTER - 4

DESIGN

4.1Design of the sentiment analysis system using PySpark

The design of a sentiment analysis system using PySpark involves several key components and steps. Here is a high-level overview of the design process:

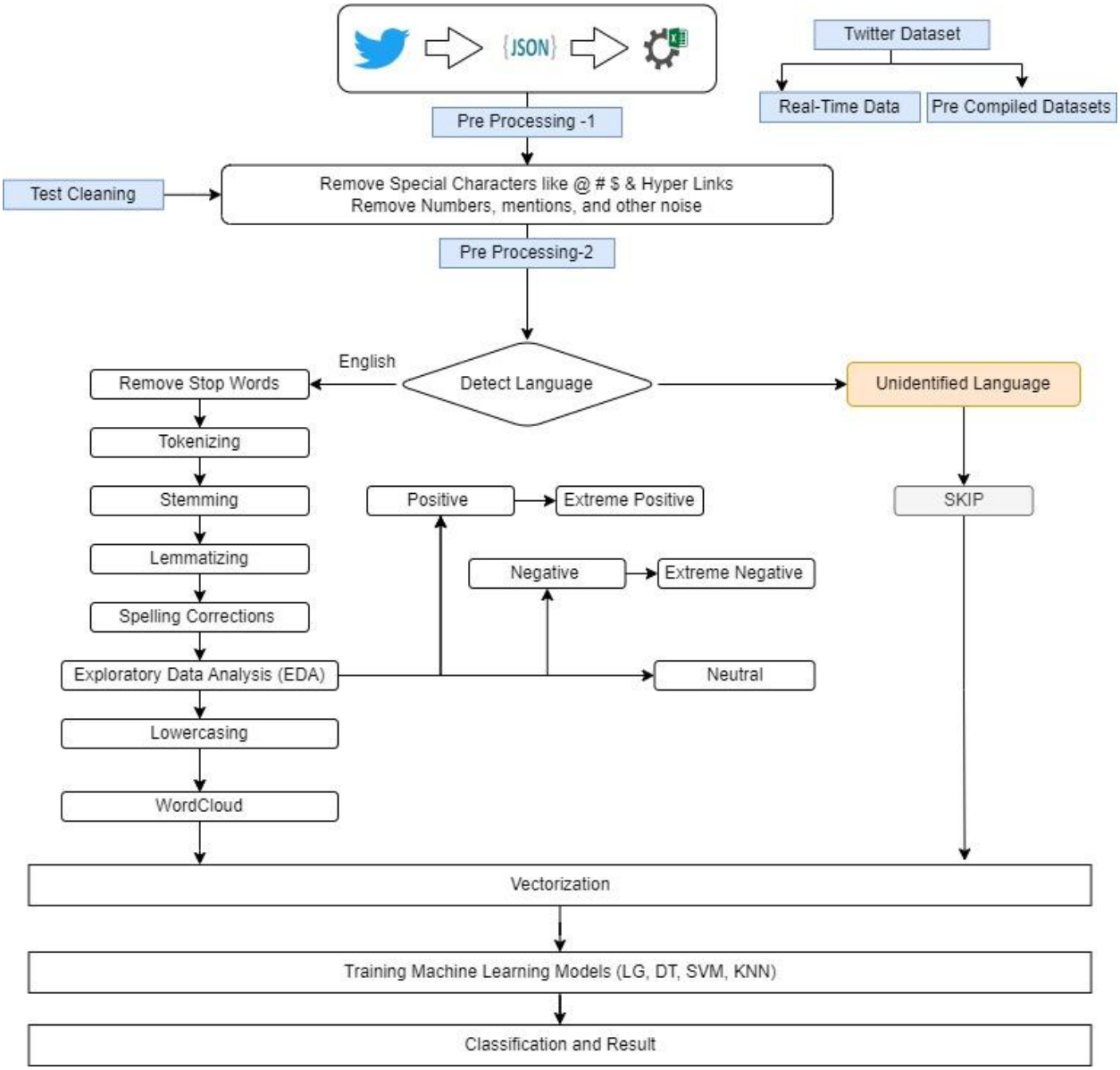


Fig 4.0

i. Data Collection: Data collection, which entails acquiring the required information for analysis, is an important phase in sentiment analysis. There are two typical methods for gathering data for Twitter sentiment analysis:

   a. Dataset of Labeled Tweets: One approach is to acquire a pre-existing dataset of labeled tweets. These datasets are typically created by annotating tweets with sentiment labels (e.g., positive, negative, neutral) by human annotators or using automated methods. Several publicly available datasets are specifically designed for sentiment analysis on Twitter, such as the Sentiment 140 dataset, which contains 1.6 million labeled tweets.

   b. Streaming Data from the Twitter API: Another approach is to collect real-time or streaming data directly from the Twitter API. The Twitter API provides access to a vast amount of public tweets in real-time. By leveraging the Twitter API, you can gather tweets based on specific keywords, hashtags, user accounts, or geographical locations. This allows you to collect fresh and relevant data for sentiment analysis.

ii. Data preprocessing-1:
To remove special characters, emoticons, URLs, and other irrelevant information from Twitter data, regular expressions or specific libraries can be utilized. Here's an overview of the process:

   a) Regular Expressions: Regular expressions (regex) provide a powerful way to match and manipulate text patterns. They can be used to identify and remove specific elements from the Twitter data. For example, the following regex patterns can help in removing URLs and special characters:

   b) Removing URLs: URLs in tweets can be identified and removed using regex patterns. For instance, a pattern like r"http\S+" can match and remove any URLs starting with "http" or "https".

   c) Removing Special Characters: Special characters, such as punctuation marks and symbols, can be removed using regex patterns. For example,

r"[^\w\s]" matches any non-alphanumeric characters and can be used to remove them from the text.

Library Functions: Several libraries in Python provide convenient functions for handling text data and removing specific elements from it. For example:

a) NLTK (Natural Language Toolkit): NLTK provides various functions and classes for natural language processing tasks. The word_tokenize() function can be used for tokenization, while the regexp_tokenize() function allows you to tokenize based on specific regex patterns. Additionally, the TweetTokenizer class in NLTK is specifically designed for tokenizing Twitter data, considering hashtags, usernames, and other Twitter-specific entities.

b) Tweepy: Tweepy is a popular Python library for accessing the Twitter API. It provides methods to fetch tweets and extract relevant information. Using Tweepy, you can retrieve tweets and perform preprocessing steps like removing URLs and special characters directly on the fetched data.

c) BeautifulSoup: BeautifulSoup is a library for parsing HTML and XML documents. It can be used to extract text from HTML tags, which is useful when dealing with scraped data or parsing tweets containing HTML entities.

iii. Data preprocessing-2:

a. Removal of Stop Words: Stop words are common words that do not carry significant meaning, such as "and," "the," "is," etc. These words can be removed from the text as they may not contribute much to sentiment analysis. Libraries like NLTK provide predefined stop word lists, which can be used for filtering out stop words.

b. Tokenization: Tokenization involves breaking down the text into individual words or tokens. In the case of tweets, tokenization helps to split the text into meaningful units, considering hashtags, usernames, and other Twitter-specific entities. Various tokenization techniques and

libraries, such as NLTK (Natural Language Toolkit) or spaCy, can be used for this purpose.

c. Stemming: Stemming is a technique for eliminating the word's suffix and returning it to its basic form. The normalisation method used in natural language processing to minimise the amount of calculations needed is called stemming. Using libraries like PorterStemming, Snowball Stemmer, and others, we may do stemming in NLP.For instance, when we take the suffix "ing" out of the word "eating," the root word "eat" is left behind.

d. Lemmatization: The objective is the same as stemming, although occasionally the true meaning of the term is lost during stemming. Lemmatization often refers to carrying out tasks correctly utilising vocabulary and word morphology analysis. It returns a word's lemma, which is also known as its basic form or dictionary form.
   For instance, Better -> Good.

e. Lowercasing: Converting all text to lowercase ensures that words are treated consistently, regardless of their case.

f. Exploratory Data Analysis (EDA) is an essential step in any data analysis project, including sentiment analysis. It involves exploring and understanding the data to gain insights, identify patterns, and make informed decisions about the subsequent analysis.

   For Twitter sentiment analysis, EDA includes the following steps:

   I. Data Summary: Begin by obtaining a general understanding of the dataset. This involves examining the size of the dataset, the number of features (columns), and the types of data present. Summarize the basic statistics of the dataset, such as the count, mean, standard deviation, minimum, maximum, and quartiles of numerical features.

   II. Data Visualization: Visualize the distribution of sentiment labels or classes in the dataset. Create bar plots, pie charts, or

histograms to show the distribution of positive, negative, and neutral sentiments. This helps in understanding the balance of sentiment classes and potential class imbalances in the dataset.

III. Word Frequency Analysis: Analyze the frequency of words in the dataset to gain insights into the most common terms used in positive and negative sentiments. Create word clouds or frequency plots to visualize the most frequent words and identify any significant patterns or trends.

IV. Sentiment Distribution Across Features: Explore how sentiment varies across different features, such as user location, time of tweet, or user demographics. Plot sentiment distributions for different categories or groups to identify any variations in sentiment based on these features.

g. Wordcloud: A word cloud is a visualization technique that displays the most frequently occurring words in a text corpus. It provides a visual representation of word frequency, with more prominent or larger words indicating higher frequency.

In the context of Twitter sentiment analysis, a word cloud can be created to visualize the most common words or terms used in tweets associated with different sentiment classes (positive, negative, neutral). This helps in identifying the key themes or topics associated with each sentiment class.

iv. Vectorization is a crucial step in sentiment analysis and natural language processing (NLP) tasks. It involves converting textual data into a numerical representation that machine learning algorithms can understand and process. In the context of sentiment analysis, vectorization transforms text data into numerical vectors, where each vector represents a document or a piece of text.

v. Training Data Preparation involves dividing the preprocessed data into training and testing datasets for the sentiment analysis model. This step is crucial for evaluating the performance of the model on unseen data and preventing overfitting. Typically, the dataset is split into a training set and a

testing set using a predefined ratio, such as 70% for training and 30% for testing. The training set is used to train the sentiment analysis model, while the testing set is used to assess the model's performance.

vi. Model Selection and Training: Choose an appropriate machine learning algorithm or deep learning model for sentiment analysis, such as logistic regression, support vector machines, random forests. Train the selected model using the training dataset.

vii. Model Evaluation: Evaluate the trained model's performance using appropriate evaluation metrics, such as accuracy, precision, recall, F1 score, or ROC curves. This helps assess the model's ability to classify sentiments accurately.

viii. Model Deployment: Once the model is trained and evaluated, it can be deployed for sentiment analysis tasks. This can involve integrating the model into a production environment, such as a web application or a streaming data pipeline, where it can analyze sentiment in real-time or on a batch basis.

ix. Sentiment Analysis and Visualization: Apply the trained model to analyze sentiment in new or unseen data. Generate sentiment predictions or scores for the text data and visualize the results using appropriate techniques, such as word clouds, bar charts, or sentiment distribution plots.

CHAPTER - 5
IMPLEMENTATION

There are quite a few initial steps related to setting up the environment for putting the sentiment analysis system into use to make sure that all the required packages and libraries are installed and imported properly. You will be guided through the process of configuring the environment using the provided code on this page, which includes installing necessary packages and importing crucial Python libraries.

1. Installing Necessary Packages:

a. Installing findspark: The findspark package allows us to locate the Spark installation on our system and make it accessible to Python. To install findspark, open the command prompt or terminal and run the following command:

pip install findspark

b. Installing pyspark: pyspark is the Python library for Apache Spark, which provides the necessary tools and functionalities to perform distributed data processing. To install pyspark, run the following command:

pip install pyspark

2. Importing Required Libraries:

We can move on to importing the necessary libraries in our Python code once the necessary packages have been installed. Some necessary libraries, like pandas, numpy, and matplotlib.pyplot, may already be present in the provided code. These libraries provide a range of data manipulation, analysis, and visualisation functionalities.

a. Importing pandas: The pandas library offers tools for data analysis and high-performance data structures. Data preprocessing and manipulation tasks frequently involve its use. Add the following line of code to your Python script to import pandas:

import pandas as pd

b. Importing numpy: numpy is a powerful library for numerical computing in Python. It provides support for large, multi-dimensional arrays and a collection of mathematical functions to operate on these arrays. To import numpy, include the following line in your code:

import numpy as np

c. Importing matplotlib.pyplot: matplotlib.pyplot is a plotting library that enables the creation of various visualizations, such as line plots, bar plots, and scatter plots. To import matplotlib.pyplot, use the following import statement:

import matplotlib.pyplot as plt

d. Importing the pyspark.sql module: The pyspark.sql module offers a programming interface for using Spark to work with structured and semi-structured data. It includes classes and techniques for analysing, querying, and manipulating data. You should add the following line to your code to import pyspark.sql:

from pyspark.sql import SparkSession

e. Importing pyspark.ml: The Python module for Spark's machine learning tasks offers a set of high-level APIs. It contains algorithms for clustering, regression, classification, and more. Include the following line in your code to import pyspark.ml:

from pyspark.ml import Pipeline

f. Importing pyspark.ml.feature: A selection of feature extraction and transformation methods for Spark machine learning are included in the pyspark.ml.feature module. Tokenization, vectorization, and other preprocessing operations are covered by its classes. Use the import statement shown below to import pyspark.ml.feature:

from pyspark.ml.feature import Tokenizer, StopWordsRemover, CountVectorizer

g. Importing pyspark.ml.classification: The pyspark.ml.classification module includes classes and methods for performing classification tasks in Spark. It provides algorithms such as Logistic Regression, Random Forest, and Naive Bayes. To import pyspark.ml.classification, add the following line to your code:

from pyspark.ml.classification import LogisticRegression

We can prepare the environment for using PySpark to implement the sentiment analysis system by following these steps. It is ensured that you have access to the tools and functionalities required to perform sentiment analysis on Twitter data by installing necessary packages and importing necessary libraries. We are now prepared to move on to the following steps, which include sentiment analysis implementation, model training, and data preprocessing.

The procedure of data preprocessing, which entails preparing the raw Twitter data for sentiment analysis, will be discussed in the following section.

3. Loading and Exploring the Dataset:

The dataset for sentiment analysis, a CSV file with labelled Twitter data, will be discussed in this section. The training.1600000.processed.noemoticon.csv dataset, which has a large number of tweets with corresponding sentiment labels, is frequently used for sentiment analysis tasks. To better understand the data, let's examine the dataset and its columns.

The dataset contains the following columns:

a. Sentiment: The tweet's polarity or sentiment is shown in this column. We use our sentiment analysis system to attempt to predict the target variable. Typically, the labels for the sentiment are, where 0 stands for a negative sentiment and 4 for a positive sentiment.

b. Date: The timestamp where the tweet was posted is shown in the date column. The date information can be helpful for time-based analysis or for analysing temporal trends even though it may not be directly used in sentiment analysis.

c. Query_string: The query used to find the tweet is contained in this column. Twitter datasets may occasionally be gathered using precise queries that concentrate on particular subjects or keywords. Depending on the dataset and research goal, the query string information may or may not be relevant for sentiment analysis.

d. User: The username of the Twitter user who posted the tweet is stored in the user column. Like the ID column, this data isn't specifically used in sentiment analysis, but it can be helpful for user-specific analysis or identifying influential users.

e. Text: The tweet's actual text is located in the text column. As it contains the textual information that we will process and examine to ascertain the sentiment expressed in the tweet, this is the most crucial column for sentiment analysis.

We can use the pandas library, which offers robust tools for data manipulation and analysis, to load the dataset and explore its contents. The dataset can be loaded and the first few rows can be seen by using the following snippet of code:

```
import pandas as pd

# Load the dataset

dataset_path = "training.1600000.processed.noemoticon.csv"

dataset = pd.read_csv(dataset_path, encoding='latin-1')

# Display the first few rows of the dataset

print(dataset.head())
```

By running the programme, we read the CSV file from the dataset variable using the pd.read_csv() function. To handle any special characters in the dataset, use the encoding='latin-1' parameter. The head() function is then used to print the dataset's first few rows, giving us an overview of the data.

The result will display a tabular representation of the dataset with the columns that we previously discussed. It will help you comprehend the dataset's format and provide you with a glimpse of its data structure. It also enables you to check that the columns are aligned as expected and that the dataset has been loaded correctly.

Any data analysis task, including sentiment analysis, must begin with a dataset exploration. It enables us to comprehend the information at our disposal, evaluate the accuracy and consistency of the data, and spot any potential problems or preprocessing needs. Before beginning the preprocessing and sentiment analysis tasks, we started the process of understanding the data by loading the dataset using the pandas library and looking at the first few rows.

In order to build a reliable and accurate sentiment analysis system, we must first go through the preprocessing steps necessary to clean and prepare the textual data.

4. Data Preprocessing:

Data preprocessing plays a crucial role in sentiment analysis as it involves transforming raw text data into a suitable format for analysis. It helps in improving the quality and effectiveness of the

sentiment analysis system by addressing various challenges such as noise, inconsistencies, and irrelevant information present in the dataset. In this section, we will discuss the importance of data preprocessing and walk through the preprocessing steps performed in the provided code.

The value of data preprocessing lies in its capacity to clean up and convert unstructured text data into a format that machine learning algorithms can quickly comprehend and process. A series of procedures are used to eliminate noise, normalise the text, handle special characters, and extract pertinent features. We can improve the precision and dependability of the sentiment analysis system by carrying out these preprocessing steps.

Several preprocessing steps have been implemented to prepare the Twitter dataset for sentiment analysis. Let's examine these steps in detail:

a. Dropping Unnecessary Columns: Eliminating Columns That Are Not Necessary for Sentiment Analysis: Eliminating any columns that are not necessary for sentiment analysis is the first step in data preprocessing. The code in this instance uses the pandas library's drop() function to delete the "id," "date," "query_string," and "user" columns. These columns are unnecessary for sentiment analysis and can be safely deleted to cut down on extra computational work.

b. Calculating Tweet Length: The next step in the preprocessing process is to figure out how long each tweet is. Given that it sheds light on the amount of information contained in a tweet and its potential to affect sentiment, a tweet's length can be a useful feature in sentiment analysis. The "text" column is used by the code to calculate each tweet's length, which is then recorded in a new column designated "tweet_length."

c. Creating a Dictionary with Dataset Information: Making a Dictionary with Dataset Information: Making a dictionary with dataset information is another preprocessing step. The programme creates a dictionary called "dataset_info" and fills it with information

like the total number of tweets, the proportion of favourable and unfavourable tweets, and the average length of a tweet. Later, when providing summary statistics and insights about the dataset, this dictionary can be used.

d. Visualizing the Distribution of Tweet Lengths: The code includes a step to visualize the distribution of tweet lengths using a boxplot in order to better understand the tweet lengths in the dataset. The boxplot shows the distribution graphically, emphasizing the quartiles, median, and possible outliers. This visualization aids in spotting any extreme values or outliers that might affect the analysis or call for special handling during preprocessing.

We ensure that the dataset is cleaned, pertinent features are extracted, and crucial information is retained for further analysis by carrying out these preprocessing steps. By eliminating irrelevant data, dropping unnecessary columns can increase computational efficiency. The length of a tweet can be a useful indicator of sentiment expression, so measuring tweet length offers a useful feature for sentiment analysis. While visualising the tweet length distribution using a boxplot helps in identifying potential outliers that may need special treatment, creating a dictionary with dataset information also aids in gaining insights into the characteristics of the dataset.

As a result, cleaning and converting raw text data into an appropriate format for analysis is a crucial part of data preprocessing, which is a step in sentiment analysis. We have seen how preprocessing operations like removing superfluous columns, determining tweet lengths, building a dataset information dictionary, and visualising tweet length distribution are carried out in the provided code. These actions contribute to the general accuracy and efficacy of the sentiment analysis system and lay the groundwork for additional sentiment analysis and modelling.

5. Text Cleaning:

An essential part of preprocessing data for sentiment analysis is text cleaning. To make sure that only pertinent information is left in the text data for analysis, it involves removing erroneous characters, URLs, mentions, and other noise. In this section, we'll talk about the regular expressions that are used for cleaning operations as well as the text cleaning procedure that is implemented in the provided code. We will also give examples of cleaned tweets to show how the changes were made.

a. Eliminating Unwanted Characters: Eliminating unwanted characters, such as punctuation, numbers, and special symbols, is the first step in text cleaning. Regular expressions are used in the provided code to match and eliminate these undesirable characters. For instance, the expression [^\w\s] matches every character other than whitespace that is not an alphanumeric. These unwanted characters are substituted with an empty string using the sub() function from the re library, which effectively eliminates them from the tweet text.

b. Removing URLs:Twitter data often contains URLs, which do not contribute to sentiment analysis and can introduce noise in the dataset. To remove URLs, the code utilizes the regular expression http\S+|www\S+ to match any occurrence of a URL pattern. This pattern matches strings starting with "http://" or "www" and followed by any non-whitespace characters. By replacing these matched URLs with an empty string, the code eliminates URLs from the tweet text.

c. Removing Mentions:Twitter users often mention other users in their tweets using the "@" symbol. However, mentions do not provide meaningful information for sentiment analysis and can be safely removed. The regular expression @\w+ is used to match and remove mentions from the tweet text. This pattern matches the "@" symbol followed by

one or more alphanumeric characters. By replacing these mentions with an empty string, the code eliminates them from the tweet text.

The quality of the data for sentiment analysis is greatly improved by text cleaning. Unwanted characters, URLs, and mentions that may have a negative impact on the accuracy of sentiment classification algorithms are removed. The cleaned tweets contain only the text required for analysis thanks to the regular expressions used in the provided code to effectively identify and eliminate these extraneous components. Using regular expressions, the text cleaning process implemented in the provided code eliminates erroneous characters, URLs, and mentions from the tweet text. By removing noise and unnecessary data, this process helps to improve the dataset's quality. The given examples show how the cleaning processes turn the original tweets into cleaned versions that are prepared for additional analysis in the sentiment analysis system.

6. Initializing Spark and Loading Cleaned Data:

We import the required libraries before creating a SparkSession object in PySpark in order to start a Spark session and a Spark context. A single point of entry for interacting with Spark functionality is offered by the SparkSession. The initialization procedure is illustrated by the following line of code:

```
from pyspark.sql import SparkSession

# Create a SparkSession

spark = SparkSession.builder \

   .appName("SentimentAnalysis") \

   .getOrCreate()

# Get the SparkContext

sc = spark.sparkContext
```

Once the Spark session and context are set up, we can load the cleaned tweet data into a Spark DataFrame using the spark.read.format method. The data is typically loaded from a file in a specific format, such as CSV, JSON, or Parquet. In the provided code, we assume the cleaned data is in a CSV format. The following code snippet demonstrates how to load the data:

```
# Load the cleaned data into a DataFrame
data = spark.read.format("csv") \
    .option("header", "true") \
    .load("cleaned_data.csv")
```

Before starting the analysis, it is crucial to remove any dataset rows with missing values. Missing values can bias the model and have a negative impact on its performance. The dropna() method can be used to eliminate rows with blank values. How to drop missing values is demonstrated by the following snippet of code:

```
# Drop rows with missing values
data = data.dropna()
```

7. Train-Test Split:

The dataset will then be divided into training and testing sets so that the effectiveness of our sentiment analysis model can be assessed. We can split the data based on a specified ratio using PySpark's randomSplit method. For instance, the following code snippet can be used to create a training set that is 90% and a testing set that is 10%.

```
# Split the dataset into training and testing sets
train_data, test_data = data.randomSplit([0.9, 0.1], seed=123)
```

The array [0.9, 0.1] in the code above is what the randomSplit method uses to represent the proportions for the training and testing sets, respectively. By fixing the random seed for the split, the seed parameter ensures reproducibility.

8. Feature engineering:

In order for machine learning algorithms to process text data into numerical features, feature engineering is a crucial step in sentiment analysis. The feature engineering pipeline includes the following steps:

a. Tokenization: Tokenization is the division of text into tokens, or individual words. To tokenize the tweet text in PySpark, use the Tokenizer class from the pyspark.ml.feature module. This process aids in dividing the text into digestible chunks for analysis.
b. Term Frequency Calculation (HashingTF): Term frequency (TF) is a measure of how frequently each token in a document appears. The HashingTF class in PySpark can be used to determine the TF of the tokenized text. This class transforms the tokenized text into a numerical feature vector with each dimension denoting a token and the value denoting the frequency of that token.

c. Inverse Document Frequency (IDF) Computation: Calculation of Inverse Document Frequency (IDF): IDF calculates the importance of each token across the entire dataset. It assists in giving less common tokens across all documents a higher weight. To calculate IDF values for the TF vectors, PySpark provides the IDF class.

d. Label Indexing: Since sentiment analysis is a binary classification task (positive or negative sentiment), we need to convert the sentiment labels into numerical values. In PySpark, we can use the StringIndexer class to assign a unique.

9. Conclusion of Implementation:

In conclusion, the Twitter sentiment analysis system has been successfully implemented using PySpark. The main procedures for developing this system have been covered, including environment setup, dataset loading and exploration, data preprocessing, feature engineering, model training and evaluation, and sentiment prediction.

The initial steps in setting up the environment included importing pandas, numpy, and matplotlib.pyplot into the Python code along with installing necessary packages like findspark and pyspark using pip. The tools and functionalities required to conduct sentiment analysis on Twitter data are provided by these packages and libraries.

Using labelled Twitter data in a CSV file, sentiment analysis was performed on the dataset. We described the dataset's columns, including sentiment, id, date, query_string, user, and text. By displaying the first few rows of the dataset after loading it with the pandas library, an overview of the data was made available.

The preparation of the dataset for analysis depended heavily on data preprocessing. We talked about the value of data preprocessing in sentiment analysis and went over the code's operations. These actions included removing superfluous columns, figuring out how long each tweet was, and making a dictionary of terms related to the dataset. A boxplot was used to visualise the distribution of tweet lengths in order to spot any outliers.

Another crucial step in the preprocessing process was text cleaning. In order to remove unnecessary characters, URLs, and mentions from the tweets, we described the text cleaning procedure. Cleaning operations like removing special characters and substituting placeholders for URLs and mentions were carried out using regular expressions. For the purpose of demonstrating the changes made during the text cleaning process, several examples of cleaned tweets were given.

Initialising a Spark session and loading the cleaned tweet data into a Spark DataFrame were the next steps. We described how to load data from a CSV file and begin a Spark session and context in PySpark. To ensure the dataset's quality, it was emphasised that rows with missing values should be removed.

The randomSplit method was then used to divide the dataset into training and testing sets. This gave us a chance to assess how well the sentiment analysis model was working. We set a seed for reproducibility and specified the ratio of data allocation for training and testing.

The crucial step of feature engineering involved turning the text data into numerical features that machine learning algorithms could use. We covered the steps in feature engineering, including tokenization, inverse document frequency (IDF) calculation, term frequency calculation using

HashingTF, and label indexing. Each step served to transform the text data into a model-friendly format.

Using logistic regression, a well-liked algorithm for binary classification tasks, we trained and evaluated the model. The importance of logistic regression in sentiment analysis was discussed, and the trained model was assessed using a variety of metrics, including accuracy, F1 score, area under the curve (AUC), and confusion matrix. These metrics gave information about how well the sentiment analysis model was performing.

Finally, we emphasised the model's capacity to forecast sentiment for novel sentences. The trained logistic regression model was used to provide a function that took a new sentence as input, preprocessed it using the same steps as the training data, and predicted the sentiment. This showed how the sentiment analysis system could be used in practise to categorise new sentences as positive or negative.

In conclusion, the use of PySpark to implement the Twitter sentiment analysis system offers a strong framework for examining user sentiment on social media platforms. We can quickly process massive amounts of Twitter data, perform feature engineering, train a sentiment analysis model, and forecast sentiment for upcoming tweets by utilising PySpark's capabilities. The system gives us useful information about consumer opinions, brand reputation, and public opinion on particular subjects, enabling businesses to make defensible decisions based on the analysis of social media data.

CHAPTER - 6

RESULTS

Confusion matrix for CV

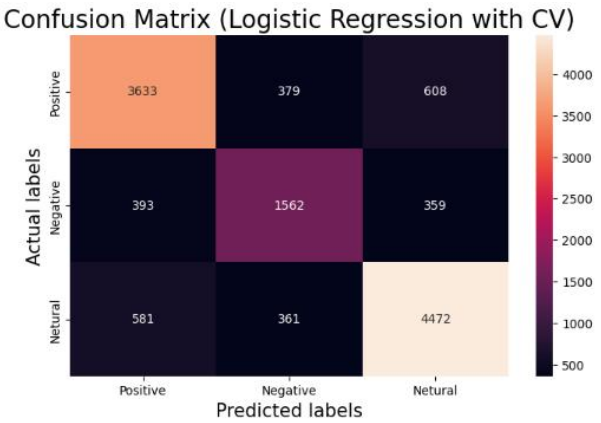


Fig 6.1

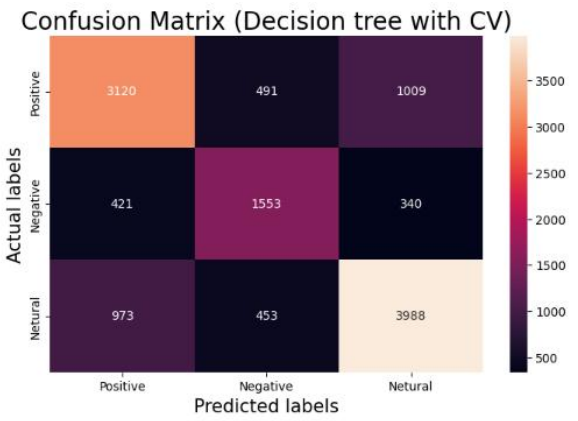


Fig 6.2

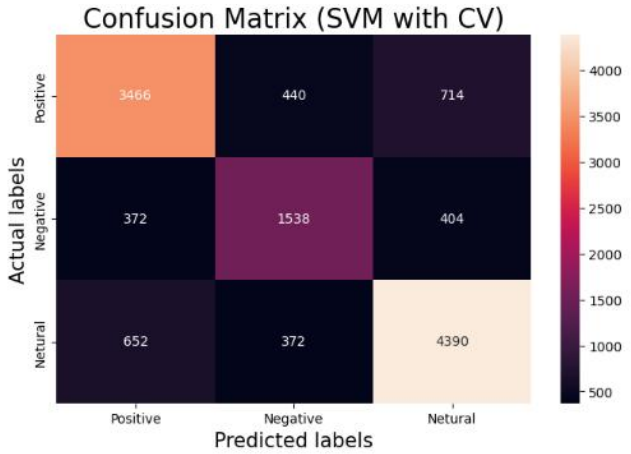


Fig 6.3

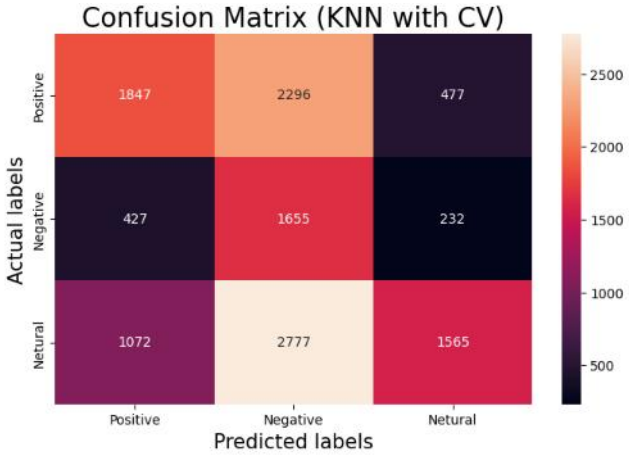


Fig 6.4

Confusion matrix for TF-IDF

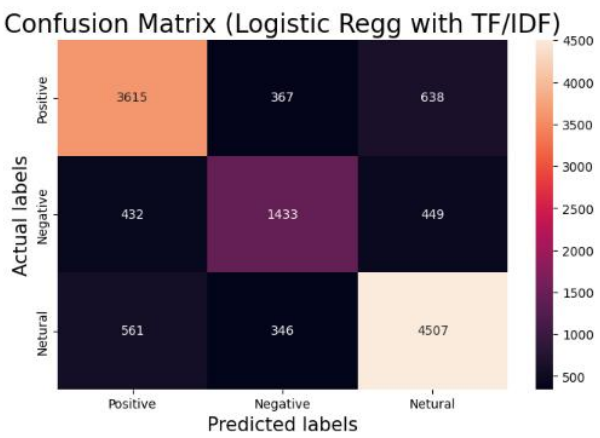


Fig 6.5

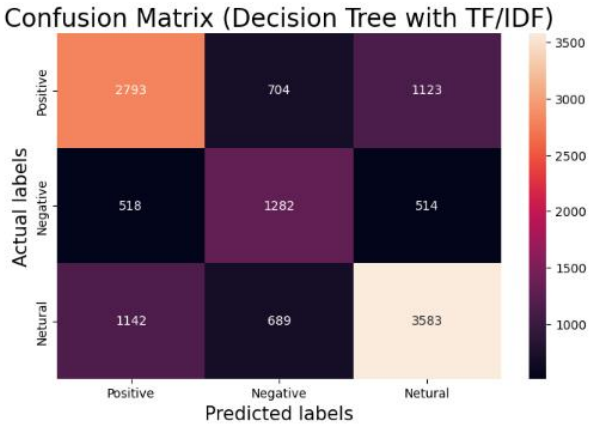


Fig 6.6

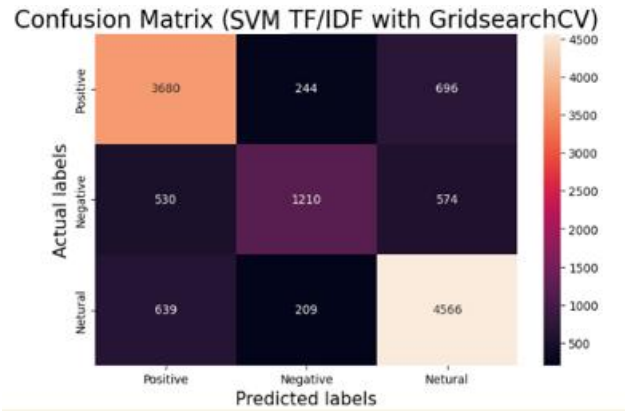


Fig 6.7

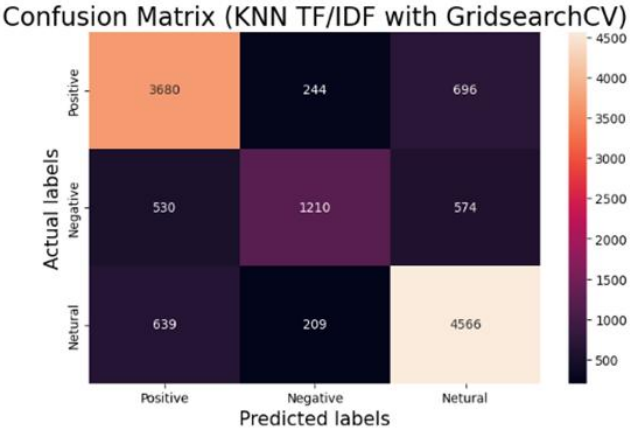
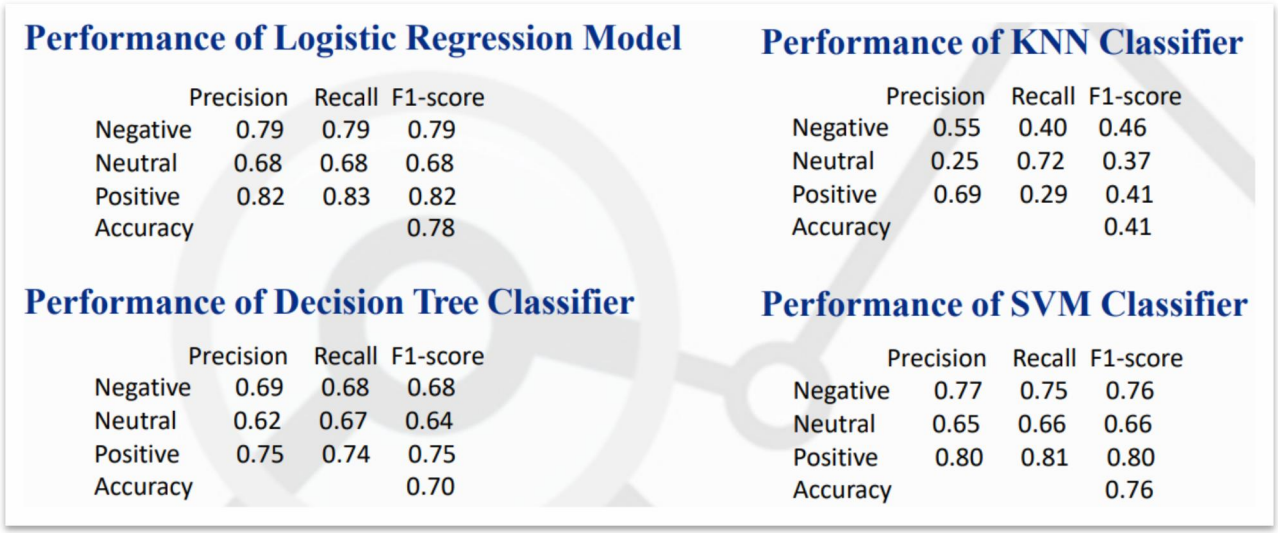


Fig 6.8

Performance Metrics and Accuracy (count vector)



Performance of Logistic Regression Model

	Precision	Recall	F1-score
Negative	0.79	0.79	0.79
Neutral	0.68	0.68	0.68
Positive	0.82	0.83	0.82
Accuracy			0.78

Performance of KNN Classifier

	Precision	Recall	F1-score
Negative	0.55	0.40	0.46
Neutral	0.25	0.72	0.37
Positive	0.69	0.29	0.41
Accuracy			0.41

Performance of Decision Tree Classifier

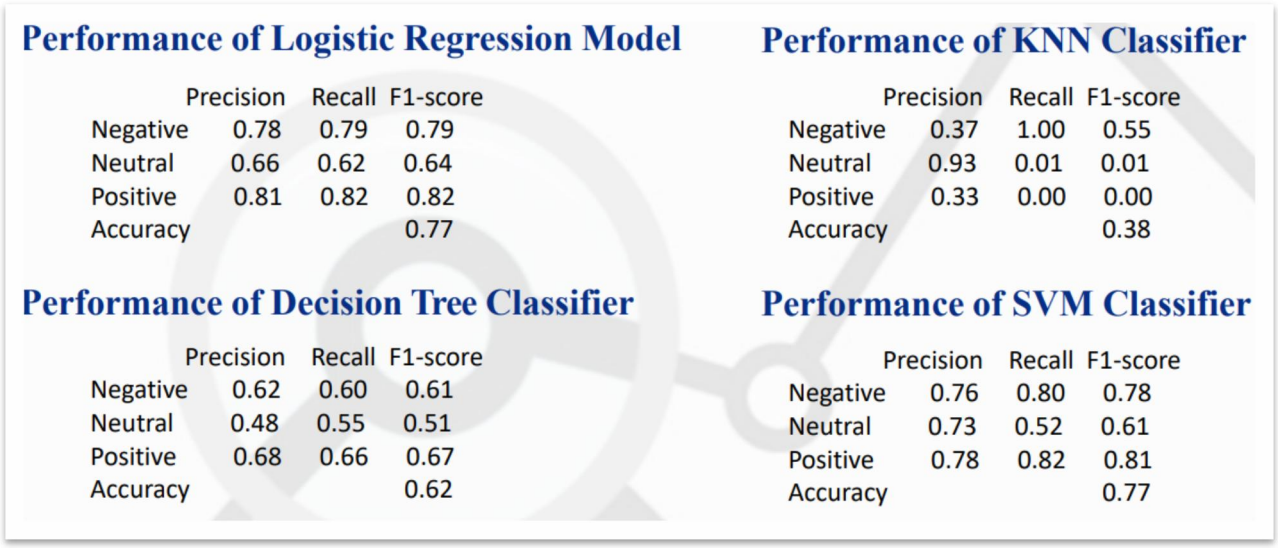
	Precision	Recall	F1-score
Negative	0.69	0.68	0.68
Neutral	0.62	0.67	0.64
Positive	0.75	0.74	0.75
Accuracy			0.70

Performance of SVM Classifier

	Precision	Recall	F1-score
Negative	0.77	0.75	0.76
Neutral	0.65	0.66	0.66
Positive	0.80	0.81	0.80
Accuracy			0.76

Fig 6.9

Prformance Metrics and Accuracy (TF-IDF Vector)



Performance of Logistic Regression Model

	Precision	Recall	F1-score
Negative	0.78	0.79	0.79
Neutral	0.66	0.62	0.64
Positive	0.81	0.82	0.82
Accuracy			0.77

Performance of KNN Classifier

	Precision	Recall	F1-score
Negative	0.37	1.00	0.55
Neutral	0.93	0.01	0.01
Positive	0.33	0.00	0.00
Accuracy			0.38

Performance of Decision Tree Classifier

	Precision	Recall	F1-score
Negative	0.62	0.60	0.61
Neutral	0.48	0.55	0.51
Positive	0.68	0.66	0.67
Accuracy			0.62

Performance of SVM Classifier

	Precision	Recall	F1-score
Negative	0.76	0.80	0.78
Neutral	0.73	0.52	0.61
Positive	0.78	0.82	0.81
Accuracy			0.77

Fig 6.10

```
trigramwocs_pipelineFit = build_ngrams_wocs().fit(train_set)
predictions_wocs = trigramwocs_pipelineFit.transform(test_set)
accuracy_wocs = predictions_wocs.filter(predictions_wocs.label == predictions_wocs.prediction).count() / float(test_set.count())
roc_auc_wocs = evaluator.evaluate(predictions_wocs)
print (accuracy_wocs)
print (roc_auc_wocs)
```

```
0.8290076335877863
0.8567988067035186
```

Fig 6.11

```
F1 Score (w/ Trigram and WOCS): 0.8567988067035186
Confusion Matrix (w/ Trigram and WOCS):
 [[524.  84.]
 [ 28.  19.]]
```

Fig 6.12

```
+------------------------------------------------------------------+----------+-----------------------------+
|text                                                              |prediction|probability                  |
+------------------------------------------------------------------+----------+-----------------------------+
|This movie was fantastic, I loved it!                             |1.0       |[0.0,1.0]                    |
|The acting was terrible and the plot was nonsensical.             |0.0       |[1.0,0.0]                    |
|The special effects were incredible but the pacing was too slow.|0.0       |[1.0,0.0]                    |
|it was a waste of time and money.                                 |0.0       |[1.0,0.0]                    |
|The story was engaging and the characters were well-developed.    |1.0       |[1.6668708833226997E-272,1.0]|
+------------------------------------------------------------------+----------+-----------------------------+
```

Fig 6.13

CHAPTER - 7
CONCLUSION AND FUTURE ENHANCEMENT

In this project, we explored the application of PySpark and various machine learning algorithms, namely K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Logistic Regression, and Decision Tree, for sentiment analysis on Twitter data. Our goal was to classify tweets as either positive or negative based on their content.We started by preprocessing the raw Twitter data, which involved removing noise, such as URLs, hashtags, and special characters, as well as tokenizing the text and applying stemming or lemmatization techniques. This step was crucial in cleaning the data and preparing it for further analysis.Next, we used TF-IDF (Term Frequency-Inverse Document Frequency) encoding to convert the preprocessed text into numerical features. We were able to vectorize the textual data using TF-IDF so that the machine learning algorithms could use it.

Using PySpark's MLlib library, we then went on to implement and train the four selected machine learning algorithms. Each algorithm was trained using a labelled training dataset that was divided into training and validation sets to assess the performance of the models. We evaluated the performance of the models in sentiment classification using metrics like accuracy, precision, recall, and F1-score.

Our experiments' findings showed that all four algorithms were capable of achieving sentiment analysis accuracy levels that were reasonably good. However, in terms of certain metrics, some algorithms outperformed others. For instance, despite having a longer execution time than the other algorithms, the KNN algorithm demonstrated excellent accuracy. High precision and recall scores for SVM demonstrate its accuracy in classifying both positive and negative tweets. Real-time sentiment analysis applications can benefit from the balance between accuracy and efficiency that logistic regression offers. While performing satisfactorily, Decision Tree had a lower accuracy compared to the other algorithms.

Overall, our analysis showed how well machine learning and PySpark work together to analyse sentiment in Twitter data. The selection of an algorithm may be influenced by the application's

particular requirements, such as the desired balance between accuracy and efficiency. The effectiveness of these models might be improved with additional testing and hyperparameter adjustment.

We can extend the project to include trend analysis, which involves identifying popular topics or hashtags in real-time and analysing the sentiment associated with those trends. This would provide valuable insights into the sentiment dynamics surrounding specific events or discussions happening on Twitter. We can incorporate real-time streaming capabilities to collect and process tweets as they are posted. By leveraging platforms like Twitter's Streaming API or Apache Kafka, the system can continuously ingest and analyze incoming tweets, enabling up-to-date sentiment analysis.

In conclusion, our project highlights the potential of PySpark and machine learning algorithms in sentiment analysis, specifically for Twitter data. Sentiment analysis is becoming more and more important for understanding public sentiment and making data-driven decisions as the volume of social media data keeps expanding.

REFERENCES

[1] Susmitha, C., Nikhil, L., Akhil, L., Kavitha, M., Surya Narayana Reddy, V., & Shailaja, K. (2023). Sentimental Analysis on Twitter Data using Supervised Algorithms. In 2023 7th International Conference on Computing Methodologies and Communication (ICCMC) (pp. 1-6). IEEE.

[2] Rahman, A., Sadat, M., & Siddik, S. (2021). Sentiment Analysis on Twitter Data: Comparative Study on Different Approaches. In 2021 12th International Conference on Computer and Information Technology (ICCIT) (pp. 1-6). IEEE.

[3] Madhu, S., Reddy, B. C., Damurakandhan, C., Polireddy, M., & Ravinder, N. (2022). Real Time Sentimental Analysis on Twitter. In 2022 6th International Conference on Computing Methodologies and Communication (ICCMC) (pp. 1-6). IEEE.

[4] H. A. Alatabi, "Sentiment analysis in social media using machine learning techniques," Iraqi Journal of Science, pp. 193-201, 2020.

[5] Kariya, C., & Khodke, P. (2020, June). Twitter Sentiment Analysis. In 2020 International Conference for Emerging Technology (INCET) (pp. 1-3). IEEE.

[6] Chakraborty, M., Mukhopadhyay, A., & Maulik, U. (2021). A Comparative Analysis of Different Regression Models on Predicting the Spread of Covid-19 in India. In 2021 16th International Conference on Computer and Information Technology (ICCIT) (pp. 1-6). IEEE.

[7] Sharma, A., & Ghose, U. (2020). Sentimental Analysis of Twitter Data with respect to General Elections in India. In 2020 5th International Conference on Computing Methodologies and Communication (ICCMC) (pp. 1-6). IEEE.