Spark based N-gram model for Sentiment Analysis on Twitter Dataset

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

Aryan Srivastava (191322)

Under the supervision of

Dr. Hari Singh

to

Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology
Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Spark based N-gram model for Sentiment Analysis on Twitter Dataset"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2023 to May 2023, under the supervision of Dr. Hari Singh (Assistant Professor (SG)).

I also authenticate that I have carried out the above-mentioned project work under the proficiency stream **Data Science.**
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Aryan Srivastava (191322)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Hari Singh
Assistant Professor (SG)
Computer Science & Engineering Department
Date

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ……………………….

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: ___________________________ __Department: _________________ Enrolment No _________

Contact No. ______________________________E-mail. ______________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ________________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

– Total No. of Pages =
– Total No. of Preliminary pages =
– Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ……………….(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

We owe our profound gratitude and indebtedness to our project supervisor **Dr. Hari Singh**, who took keen interest and guided us all along in our project work titled **—" Spark based N-gram model for Sentiment Analysis on Twitter Dataset"**, till the completion of our project by providing all the necessary information for developing the project. The project development helped us in research, and we got to know a lot of new things in our domain. We are thankful to him.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

S. No	Title	Page No.
1	Natural Language Processing [NLP]	4

LIST OF FIGURES

<u>**LIST OF GRAPHS**</u>

LIST OF TABLES

ABSTRACT

Sentiment Analysis has grown to be a fascinating area in both academic and commercial settings. The word sentiment describes how a person feels or thinks about a certain problem. It's seen as a direct application of opinion mining as well. Twitter is a rich source of textual data and one of the most important data volumes due to the enormous volume of tweets written down each day; as a result, this data has various goals depending on the amount of data needed and the processing that will be required, such as business, industrial, or social goals. Actually, the enormous volume of data, known as big data, is growing quickly every second, necessitating the use of advanced processing methods and powerful computers to carry out the necessary mining activities.

This project aims to perform sentiment analysis on a Twitter dataset using classification algorithms on Apache Spark. The dataset consists of 1.4 million tweets on general topics or events, and the sentiment analysis will determine whether each tweet is positive or negative. The Apache Spark framework will be used to process and analyse the large volume of data efficiently. The classification algorithms to be used include logistic regression, random forest, decision tree, gradient descent, multinomial naive bayes and support vector machines. The project will involve pre-processing the data, feature extraction, and model training and evaluation. The performance of each algorithm will be compared using metrics such as accuracy, precision, and recall. The results of this project will demonstrate the effectiveness of Apache Spark for processing and analysing large volumes of Twitter data and provide insights into the sentiment of Twitter users on a particular topic or event.

CHAPTER 1

1.1 INTRODUCTION

In today's world social media plays a completely critical function in our life. What we are able to think, what we do, all of us express our emotions on social media platforms. Social media is a big, interactive medium for dialogue of numerous troubles associated with society in addition to vital for the growing unfold of facts, specifically throughout instances of herbal disasters, calamities, and mass emergencies. Also, on social media humans speak about the goods which might be released day with the aid of using day. Many groups and business enterprises use those forms of facts (associated with their merchandise) to recognize what the humans reflect on consideration on their product. They can examine these facts and the usage of Social Network Analysis. Interactions via social media systems are now no longer centralised to a selected location, time quarter etc. Social media affords a short and effective manner to unfold facts now no longer counting whether it's miles correct or inaccurate, unfolding of the both types is favourable. However social community typically favoured extra correct and legitimate facts to unfold than fake facts and rumoured facts. Interaction happens in actual time so this affords applicable unfold of facts according to applicable facts.

We use sentiment evaluation to recognize the conduct of humans closer to a selected product. Also, the way it is modified from product to product day with the aid of using day. Sentiment evaluation is the identity and class of feelings whether it's miles positive, bad or impartial the usage of textual content evaluation technique. Sentiment evaluation lets in groups and corporations to become aware of purchaser sentiments closer to their merchandise, brands, offerings in on line communication and via feedback. Sentiment evaluation makes use of the texts that are written with the aid of using a selected character on social media to investigate whether or not it's miles positive, bad or impartial.

Social media platforms like Twitter provide a wealth of information that can be used to gain insights into the opinions and emotions of users on various topics or events. Sentiment analysis is a popular technique for analysing social media data to determine the sentiment expressed in the messages. In recent years, Apache Spark has emerged as a powerful framework for processing and analysing large volumes of data efficiently. This project aims to perform sentiment analysis on a Twitter dataset using classification algorithms on Apache Spark.

The goal of this project is to develop a sentiment analysis model that can accurately classify tweets as positive, negative, or neutral based on the sentiment expressed in the text. The project will involve preprocessing the Twitter data to remove noise, such as stop words and special characters, and performing feature extraction to identify relevant features in the text that can be used for sentiment classification. The classification algorithms to be used in this project include logistic regression, random forests, and support vector machines.
Apache Spark's ability to process data in parallel across a distributed computing cluster makes it an ideal choice for handling large volumes of Twitter data. The project will leverage the scalability and performance of Apache Spark to process and analyse the Twitter dataset efficiently.

The results of this project will provide insights into the sentiment of Twitter users on a particular topic or event and demonstrate the effectiveness of Apache Spark for processing and analysing large volumes of social media data. The sentiment analysis model developed in this project can be useful in various applications, such as brand monitoring, reputation management, and market research.

Importance of Sentiment Analysis

Nearly 80% of the world's advanced information is unshaped, & information got off the internet grounded life sources are no impunity. Luckily, on account of the advancements in Machine Learning & NLP, it is presently conceivable toward making models that gain off the models & are employed to reuse & sort out happy information.
Twitter sentiment analysis fabrics permit sorting huge arrangements of tweets & fete extremity off every advertisement naturally. Likewise, the stylish part, it is quick and introductory, sparing groups significant hours & permitting them to concentrate on errands where they can have a lesser effect.

Listed below are some of the major graces of twitter sentiment analysis

- Scalability - Suppose you have to anatomize numerous tweets representing a brand. While you could do that physically, it would take a long stretch of time of homemade preparation and would wind up being disagreeable and delicate to gauge. By using this algorithm, you can robotize this errand and get smart and bring about an extremely brief timeframe.

- Real-Time Analysis - This algorithm is introductory to see abrupt moves in customer countries of mind, identify if pundits and grumblings are expanding and make a move before the issue raises. You can be checking your image continuously and get important bits of knowledge that permit you to make changes or upgrades when needed.

1.2 Problem Statement

The main thing of the design is to perform sentiment analysis on the tweets of a particular stoner and i.e., determine whether the sentiments/passions associated with a particular tweet are positive, negative, or neutral.
Also, to perform colourful kinds of graphical analysis in the data i.e., subjectivity, no, of likes, retweets etc.

Large amount of data is available over these social media platforms. This data could be put to good use by assaying it and prognosticating results and inferring farther opinions. It could be enough handy in following trades:

- Science and Technology - How target followership is replying to the technology could be prognosticated before the tech is out.
- Stock request - The prices of shares are always shifting and it's nearly delicate to prognosticate the success of an investment on a particular stock.
- Politics - There's always a lot of hassle on twitter during election season. By assaying those tweets one can get a clear idea of the fashion ability of a party/ seeker and prognosticate the results and hence decide a more effective crusade strategy.

1.3 Objectives

The major goals of this project are capturing the feelings associated with various tweets and performing a comprehensive analysis of the text data using spark language processing tools and eventually to create a machine learning model that can be used to classify actual tweets and text into positive or negative sentiment class.

In this project we also want to compare different classification algorithms such as K-Nearest Neighbour(KNN), Decision Tree, Random Forest(RF), Support Vector Classifier(SVC), Gradient Descent(GD), Naive Bayes(NB) and Logistic Regression(LR) in order to evaluate the credibility of our classifier model and determine which algorithm is the best in terms of accuracy for the testing data.

And we eventually want to deploy a working web solution to put the best working model to use and act as ML based tweet analyzer.

1.4 Methodology

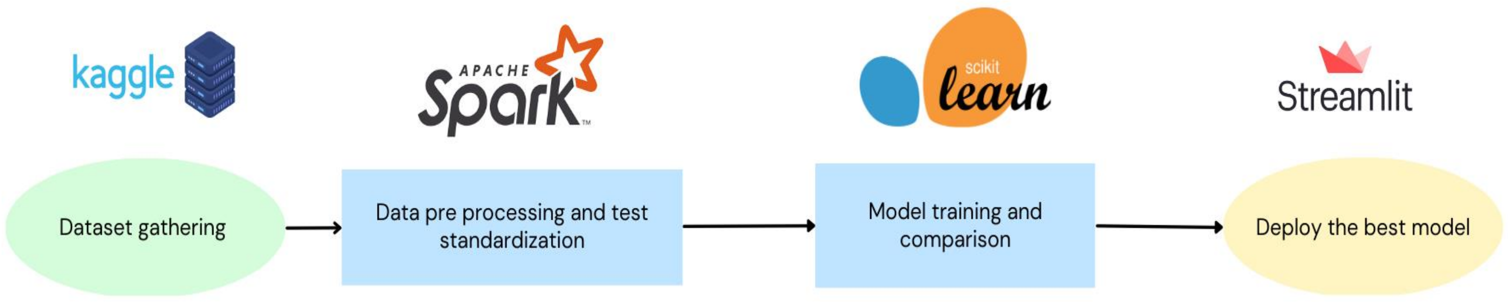


Fig 1. Project methodology

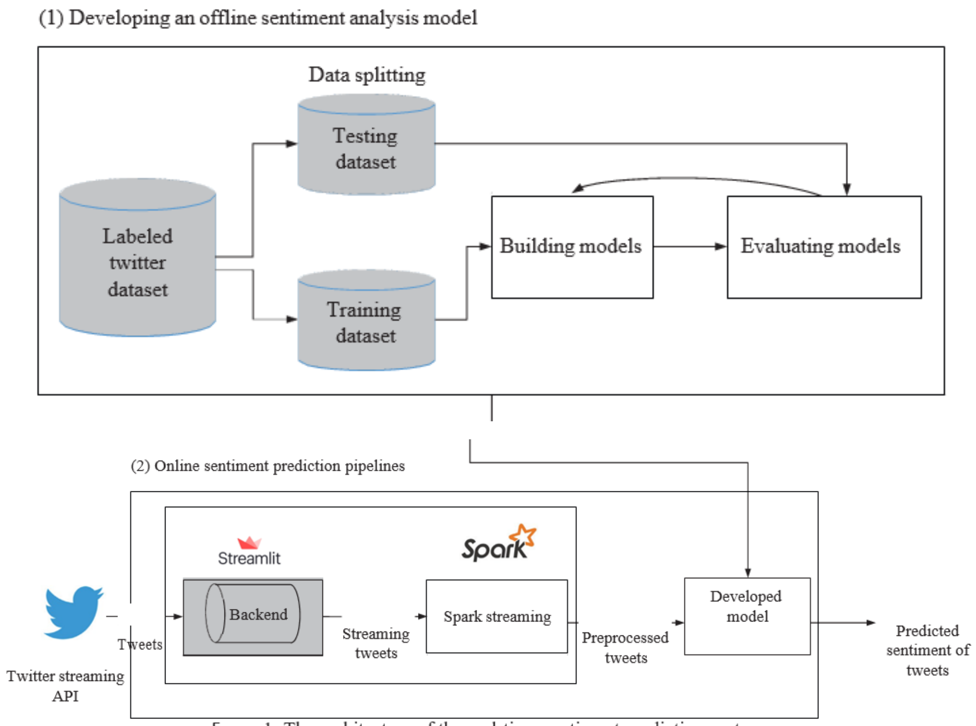


Figure 1: The architecture of the real-time sentiment prediction system.

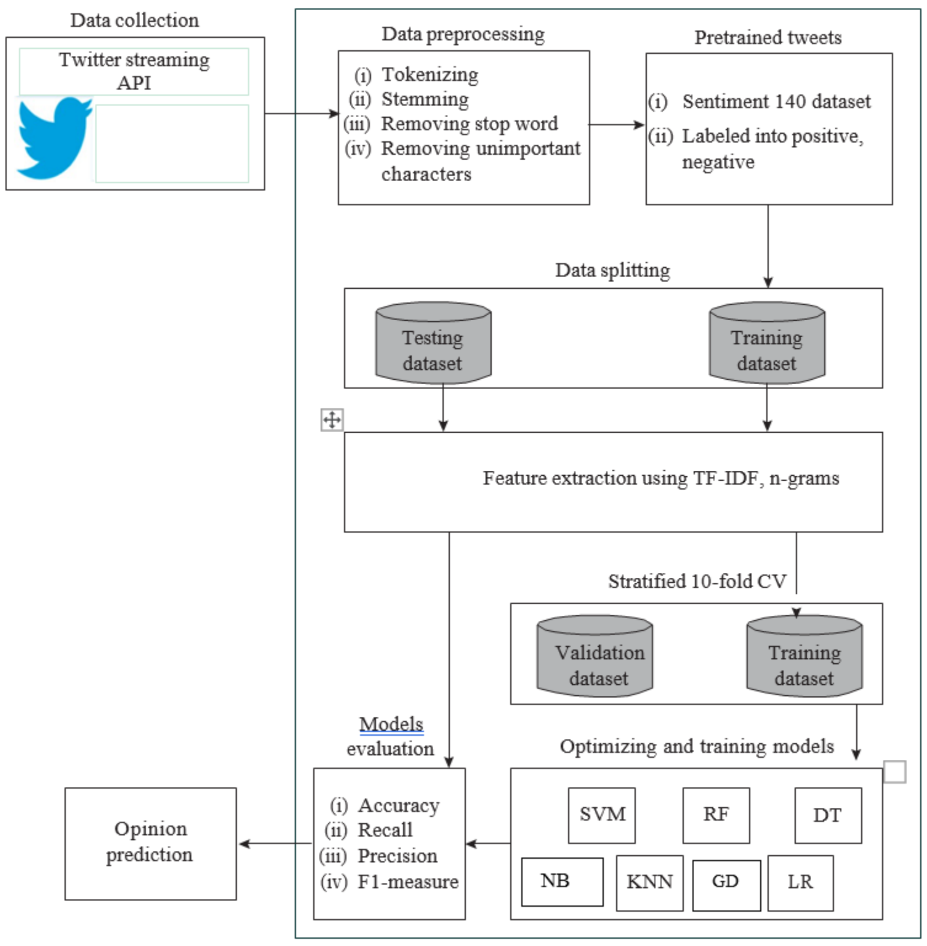


Fig 2. The offline model of the real-time sentiment prediction system.

We went with a very intuitive and straight forward approach when dealing with this project. We can articulate the methodology followed to complete this project can be expressed as follows :

- **Dataset gathering** - Dataset gathering is a crucial step in building a sentiment classification model. The dataset is essentially a collection of text data that has been labelled with a sentiment, such as positive, negative. The goal of gathering a dataset is to train a machine learning model to accurately classify the sentiment of new, unseen text data.

When gathering a dataset for sentiment classification analysis, it is important to ensure that the data is diverse and representative of the domain or topic that the model will be applied to. This means that the data should come from a variety of sources, such as social media, customer reviews, news articles, and other online sources.
The dataset should also be properly labelled with the correct sentiment. This can be done manually or through the use of automated tools, but for this project we went for a pre-labeled dataset with positive or negative sentiment associated with each tweet. Another consideration when gathering a dataset for sentiment classification analysis is the size of the dataset. A larger dataset can improve the accuracy of the model, but it also requires more computational resources and time to train the model. As a general rule, a dataset of at least several thousand examples is necessary for training a sentiment classification model.
Overall, gathering a high-quality dataset is critical for the success of a sentiment classification model. By ensuring that the dataset is diverse, properly labelled, and of sufficient size, machine learning models can be trained to accurately classify the sentiment of new text data.

- **Text preprocessing** - Text preprocessing is an essential step in natural language processing (NLP) and is typically used to transform raw text data into a format that is easier to work with for downstream NLP tasks and for the model to train efficiently. Apache Spark is a popular distributed computing framework that can be used for large-scale text preprocessing.

  There are several common text preprocessing techniques that can be implemented using Apache Spark, including tokenization, stop word removal, stemming, and lemmatization. These techniques can help to reduce the size of the text data, remove noise, and standardise the vocabulary.

Tokenization is the process of breaking up text into individual words or tokens. Apache Spark provides a built-in tokenizer that can be used to split text data into words or other units of meaning, such as phrases or sentences.
Noise Removal, at this stage, waste materials are passed through the following steps:
:

(i) Lower Casing: Lowercase is the most effective form of text before, which guarantees impact and different problems in the process.
(ii) Delete URL. In this step, we remove irrelevant links in Twitter messages.
(iii) Remove special characters. In this step we are removing special characters such as punctuation marks.
(iv) Remove completely. Twitter hashtags are used to measure content or content on Twitter, written with the # symbol.
(v) Stop word removal involves removing common words that do not carry much meaning such as a, an ,the are removed.

Stemming and lemmatization are techniques used to normalise words by reducing them to their root form. Stemming involves chopping off the ends of words to reduce them to a common stem, while lemmatization involves using a dictionary or knowledge base to map words to their base form. Apache Spark provides a stemmer and lemmatizer that can be used to implement these techniques.

In addition to these techniques, Apache Spark can also be used for other text preprocessing tasks such as encoding text data into numerical features, vectorizing text data using techniques like N-grams bag of

words or Word2Vec, and handling text data that contains special characters or non-ASCII characters.
Overall, text preprocessing using Apache Spark is a powerful way to transform raw text data into a format that is suitable for downstream NLP tasks, such as sentiment analysis, topic modelling, and text classification.

- **Model building:** Sentiment classification is a popular task in natural language processing (NLP) that involves predicting the sentiment of a given piece of text, such as positive, negative, or neutral. There are several machine learning algorithms that can be used for sentiment classification, including SVM, random forest, gradient descent, logistic regression, KNN, and Naïve Bayes.

  Support Vector Machines (SVM) is a popular algorithm for sentiment classification that works by separating the input data into different classes using a hyperplane. SVMs have been shown to work well for binary classification tasks, and can be extended to multi-class classification as well.

  Random Forest is a decision tree-based algorithm that builds a large number of decision trees and combines their predictions to produce the final output. Random Forest has been shown to be an effective algorithm for sentiment classification, as it is able to capture complex relationships between features in the data.

  Gradient Descent is an optimization algorithm that can be used to train many different machine learning models, including logistic regression and neural networks. In sentiment classification, logistic regression is often used as a baseline model, as it is simple to implement and can work well for binary classification tasks.

K-Nearest Neighbors (KNN) is a simple yet effective algorithm for sentiment classification that works by finding the k-nearest neighbours to a given input data point and using their labels to predict the label of the input data. KNN can work well for datasets with a small number of features, but can become computationally expensive for larger datasets. Naïve Bayes is a probabilistic algorithm that works by calculating the probability of a given input belonging to a particular class based on the frequency of features in the data. Naïve Bayes is a simple and fast algorithm that can work well for sentiment classification, especially when the dataset is small.

Overall, there are several machine learning algorithms that can be used for sentiment classification, each with its own strengths and weaknesses. Choosing the right algorithm for a particular task depends on factors such as the size and complexity of the dataset, the desired accuracy, and the available computational resources. Hence we choose to apply GridSearchCV to optimise the hyperparameters and different vectorisation techniques such as N-gram bag of words to check the consistency of the models

- **Deployment:** Deployment of a sentiment classification model is a critical step in making the model accessible to end-users. Streamlit is a popular Python library that can be used for building interactive web applications and can be used to deploy a sentiment classification model. To deploy a sentiment classification model using Streamlit, first, the model needs to be trained and saved in a format that can be loaded into Python. This can be achieved using the Pickle Library. Once the model is ready, a Streamlit application can be created that allows end-users to input text data and receive the predicted sentiment classification output.

Overall, deploying a sentiment classification model using Streamlit is a powerful way to make the model accessible to end-users and enable real-time sentiment analysis on text data. With its intuitive user interface and interactive features, Streamlit can provide a seamless and engaging user experience for sentiment classification applications.

CHAPTER 2

LITERATURE SURVEY

Many researchers have recently applied sentiment analysis and word frequency techniques to classify people’s attitudes from tweets. Understanding human expressions from textual data has been a focal point of studies and innovations for the past decade. Twitter being a very popular micro-blog platform which has a huge user base is a good starting point for many studies to understand and operate on textual data and deduce meaning from the same.
Several related works have focused on sentiment classification using different techniques and tools. For instance, Elzayady et al. [1] decided to perform text preprocessing on data such as removal of URLs , hashtags etc. and applied TF-IDF and unigram feature extraction with different bags of words. The data was divided into training and testing processes using a ratio of 3:1, that is, 70% of the data is used for the training process and 30% of the data is used for the testing process.

Similarly, Pang and Lee [2] proposed a machine learning approach for sentiment classification using a combination of Naive Bayes, Maximum Entropy and Support Vector Machine algorithms. The study was conducted on movie reviews, and the results showed that the machine learning approach outperformed traditional rule-based methods.

T. Carpenter et al. [3] summarises the measurement plan, separates many tweets. Prototyping was used in this development. The results separated customers' opinions, positive and negative, from the tweets described in the cookie map and html page. However, the program is designed to create a web application, but this method has to be done due to limitations with which Django can run on a Linux server or Light. Popular expressions; tweets, results, opinion mining, online life, word processing.

H. Saif et al. [4] demonstrated the value of using semantic features to categorise positive and negative emotions in tweets. They tried several ready-made solutions and decided to use AlchemyAPI because of its better performance in terms of scope and accuracy. One of the things that affects our results is the level of abstraction of the content retrieved by the site extractor. They were able to obtain 78 % accuracy and 77 % precision in results.

K. Ravi et al. [5] talk about how Apache Spark allows us to write a distributed version of any machine learning algorithm that scales easily to larger datasets on a set of hardware devices. In this paper, they propose a hybrid of sentence vectors with distributed, balanced versions of well-known machine learning techniques for emotional analysis. They use a distributed sampling method of the neural network model to obtain the sentence vectors for a body. We use various classification algorithms available in Apache Spark to classify the sentence vectors obtained in this way. They consider two methods, ie. the word bag-based Document Term Matrix (DTM) and the hash trick-based DTM. They tested it with a 992 MB review video file. Among the 6 classifiers used, MLP was equally defined between GBT and SVM, but better defined than the others, resulting in an area under the curve (AUC) of 95.44%.

Whereas Bo Yan et al. [6] proposes a support vector machine (SVM) based method for the sensitivity classification of microblogs. The plan combines microblogging features with preprocessing to ensure information is fit for distribution purposes. After preprocessing, Apache Spark parallel SVM is used for classification. SVM is one of the most popular algorithms for text classification. It is suitable for small scale and non-linear problems.
However, SVM takes a long time when processing large datasets. We use Spark to parallelize SVMs with radial-based function (RBF) cores. Introducing

Apache Spark has great performance in machine learning compared to Hadoop. Experiments show that Spark improves the execution speed of SVM. At the same time, classification accuracy is improved by the information gain (IG) method prioritisation and kernel selection function.

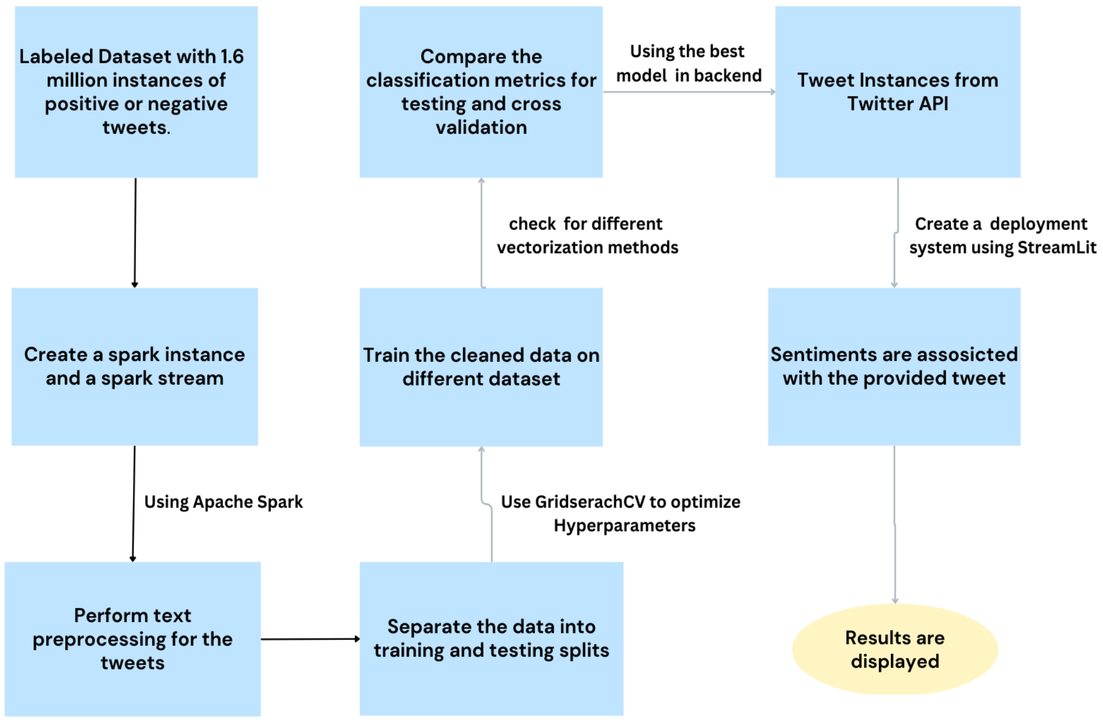
While P. Goncalves et al. [7] and A . Sharma et al [8] talk about how some articles express opinions on events, products and services, political views, and even the emotional and emotional state of the author. Sentiment analysis is used in a variety of applications, including analysis of the impact of social events, analysis of opinions about products and services, and better understanding of communication in online networks (OSNs). There are many ways to assess emotions, including language techniques and supervised machine learning. Although some methods are widely used and popular, it is not clear which method is better at determining polarity (eg., good or bad) because the available data do not provide a comparative model of the current system.

AP Jain et al. [9] showed a framework which is designed to analyse the text of Twitter data using Apache Spark, so it is more flexible, fast and scalable. Naive Bayesian and decision tree machine learning algorithms are used for conceptual analysis.

J. de Godoi Brandão et al. [10] evaluates the performance of the Support Vector Machine (SVM) classifier on tweets opinion mining in five datasets available in the literature. For feature extraction, the N-Gram and TF-IDF, k-folds cross-validation techniques were used in the classifier modelling step. Variations of N-Gram with L-gram, 2-gram, and 3-gram combined with k-folds cross-validation in 10-folds, 15-folds and 20-folds yielded 63.93% to 81.06% accuracy. Satisfactory results were obtained, which can be improved with the application of an optimization technique to adjust the classifier parameters.

CHAPTER 3

System Development



Labeled Dataset with 1.6 million instances of positive or negative tweets.
Create a spark instance and a spark stream
Using Apache Spark
Perform text preprocessing for the tweets
Separate the data into training and testing splits
Use GridserachCV to optimize Hyperparameters
Train the cleaned data on different dataset
check for different vectorization methods
Compare the classification metrics for testing and cross validation
Using the best model in backend
Tweet Instances from Twitter API
Create a deployment system using StreamLit
Sentiments are assosicted with the provided tweet
Results are displayed

Fig 2. Flow of the project

System Requirement:

Hardware:

CPU : i5 10 gen 3.2 GHz

RAM : 8 gb

GPU : None

Software:

- Linux Operating System(Apache Spark)/Windows
- Platform(Anaconda2,Spyder,Jupyter)
- NLTK package,
- Modern Web Browser
- Twitter API
- Sklearn Package

DATA COLLECTION-

Sentiment140

The sentiment140 dataset used, has 1,600,000 tweets that were extracted using Twitter are included. The tweets can be used to determine sentiment because they have been annotated (0 = negative, 4 = positive).

The following 6 fields are included in it:

1) the tweet's polarity (0 = negative, and 4 = positive).
2) ids: The tweet's identifier (2087)
3) Date: The tweet's posting date (Sat May 16 23:58:44 UTC 2009)
4) mark: The question . This value is NO QUERY if there is no query.
5) the person who tweeted)
6) text: the tweet's text (Lyx is cool)

Distribution of Data:

```
]:
    <matplotlib.axes._subplots.AxesSubplot at 0x7fa656e5fbd0>
```

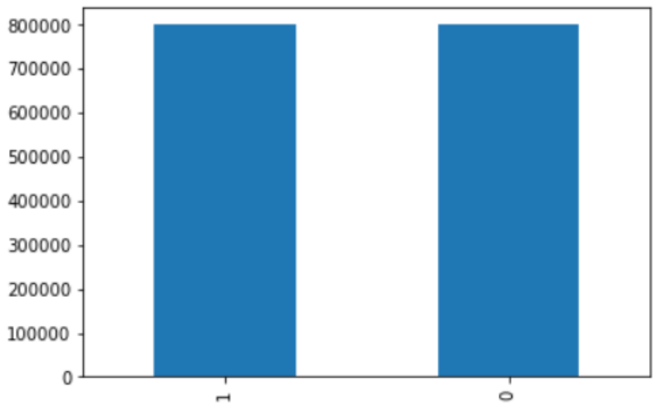
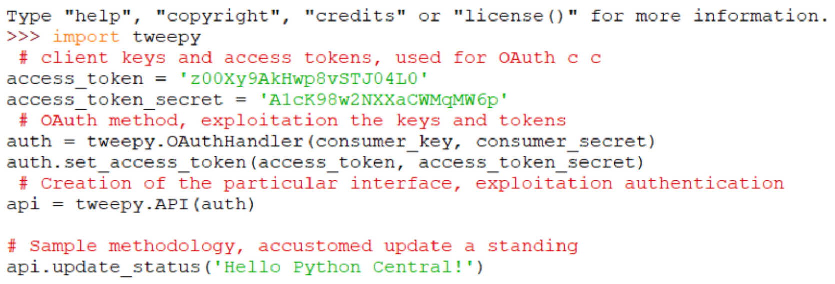


Fig 3: Tweet distribution in dataset

The dataset was used in a paper in Stanford where Alec Go et al. [11] have the main contribution of this paper is the idea of using tweets with emoticons for distant supervised learning.

Data collection for the testing part(Web App):

Twitter API Authentication-



```
Type "help", "copyright", "credits" or "license()" for more information.
>>> import tweepy
 # client keys and access tokens, used for OAuth c c
access_token = 'z00Xy9AkHwp8vSTJ04L0'
access_token_secret = 'A1cK98w2NXXaCWMqMW6p'
 # OAuth method, exploitation the keys and tokens
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
 # Creation of the particular interface, exploitation authentication
api = tweepy.API(auth)

# Sample methodology, accustomed update a standing
api.update_status('Hello Python Central!')
```

Fig 4. API authentication

To increase the completeness of the data collected, we collected the data in small increments over several time points rather than all at once. Collecting data all at once can affect the distribution of Tweets. This is because many tweets have the same sentiment or sentiment as if they were related to the trending topic. This incident was observed when examining a sample of purchased tweets.

Natural Language Processing (NLP) is a very important research area in data science today, and its important application is sentiment analysis. Normal language processing (NLP) is a major research area in informatics today, and its main application is mind testing. From analysing research to creating holistic advertising practices, this has revolutionised the way organisations work. This is an important area to be familiar with. Rather than having multiple people work together to physically complete a survey to estimate the concentration of the population, this method allows this task to be done in seconds.

Following advances are followed so as to finish feeling examination of twitter information:

- Understanding the given explanation of the issue
- Processing and cleaning of the tweets
- Generation of story and perception from the removed tweets
- Extraction of the necessary highlights from cleaned tweets
- Model structure of the assessment investigation

OAuth can be a little more difficult to get started with than basic authentication as it requires a lot of effort, but the benefits OAuth offers are huge.

Tweets are aware of the application used. to get the string that User passwords are not provided for security reasons. Permissions are easier to handle. For example, a collection of tokens and keys is created. This will allow you to read only her courses of events. Ability to compose or send direct messages is limited.

The app does not react to the secret word, so the app works whether the client converts it or not. After logging into the gateway and navigating to the app, you can create another app that can use the Twitter API to provide the information needed for action. This is the screen that contains all the information you need to communicate with the Twitter network. Note that by default, apps do not have access to direct messages. So, you can change your app to give you access to all Twitter features by setting the appropriate options for 's "read, write, and send messages" to dynamically select them. increase.

NLP –

Natural Language process (NLP) is the intersection of applied science, Linguistics and Machine Learning that's attached the interaction between computers and humans in tongue.

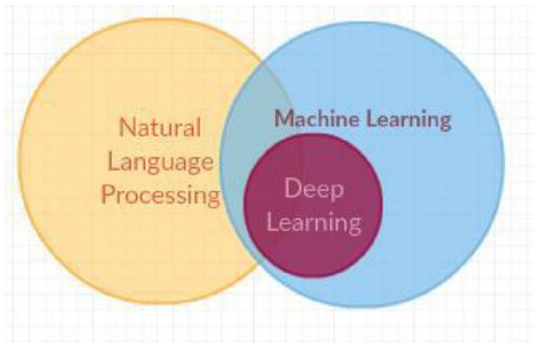


Fig 5: NLP

NLP is far from being able to capture and transmit human speech for a PC. Applications of the Informatics system are in text separation, machine interpretation, and voice agents such as Alexa and Siri. Computer science is one field that benefits from advanced methods of machine adaptation, especially deep learning methods.

The usual way to create idioms is to use a distinctive dialect tool cabinet to establish the principle of organising tasks in Python to compute using human dialect information. In many cases, this is made easier by providing an interface to at least corpus and lexicon resources for presentations, semi-sections, and sentences, allowing words to be assembled in characteristic frames. It deals with the underlying language library of arguments and for dynamic discourse.

The NLTK uses a huge instrument space and can make things easier for people with a full version system. It divides the sub-sentences of sections, bisects words, looks at the syntactic segments of those words, shows the basic theme, and by doing this it tells the machine to recognize the bulk of the substance. Helpful.

Platform Used –

Windows 10 –

We're talking about Windows 10 because Microsoft works with the current framework of PCs, tablets, built-in gadgets, and more. Windows 10 released by

Microsoft is the successor to Windows 8 in . The successor to Windows 10 was predicted in the Gregorian calendar month to be resurrected rather than unloaded and framed. When window 10 is selected or received, it is updated by direct inheritance from window 7, 8, or window 10. It does not interfere with the design methodology or activities of Framework .

For maintenance buyers, running Windows 10, they will help swap applications to previous software packages and set them to Windows 10. customers get windows 10 and enter or update. With the help of Window Refresh, the Partner Window Ten has been redesigned and physically begins an associate degree overhaul of Windows. 18 Windows 10 is used by IT offices to focus on additional features that enable them to use mobile phones, secure and manage gadgets through on-board (MDM) programming, and run the operating system to support About boarding programming such as the Microsoft Framework Centre Arrangement Chief in examples. Microsoft Windows 10 is used for various validation advances such as good cards and tokens.

Additionally, the Windows Hi will have Windows 10 biometric verification every time the purchaser logs into her with a new fingerprint or facial recognition. This framework is used to embed virtualization-based security tools such as Secluded Shopper Mode, Windows Safeguard Gizmo Watch, and Windows Shielded Qualification Monitor. Windows 10 is deployed to keep explicit data, procedures, and customer authentication highlights separate to resolve issues for each strike. Windows 10 is the new version to covertly write Bit Locker, see data between client devices, relay equipment, message and cloud management Windows 8 came with a new plan that offered touch-controlled, motion-controlled user interfaces like mobile phones and tablets, but there weren't many good interpretations for traditional workplaces and digital computers, especially in business environments. .

<u>Algorithms Used –</u>

Logistic Regression –

Logistic regression is a method used to construct training models in which the dependent variable is binary: eg. dual. Logistic regression is used to describe data and the relationship between one variable and one or more variables. The independent variables can be nominal, ordinal, or continuous.
The name "logistic regression" comes from the concept of the logistic function it uses. The logistic function is also called the sigmoid function. The value of this logistic function is between 0 and 1.
Here is an example of a logistic function that we can use to calculate the probability of a car failing based on how many years have passed since the car was last serviced.

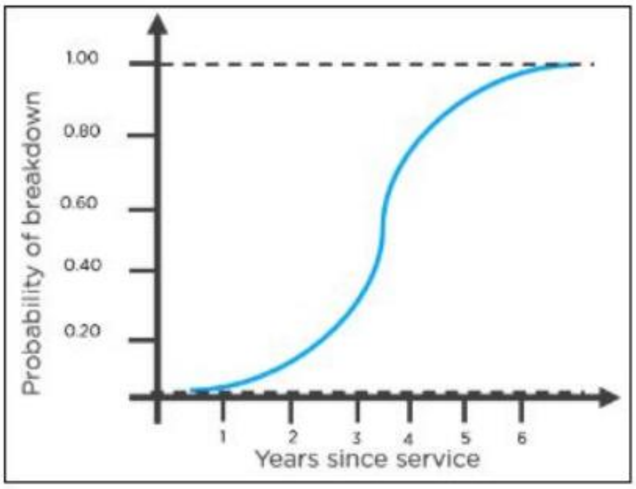
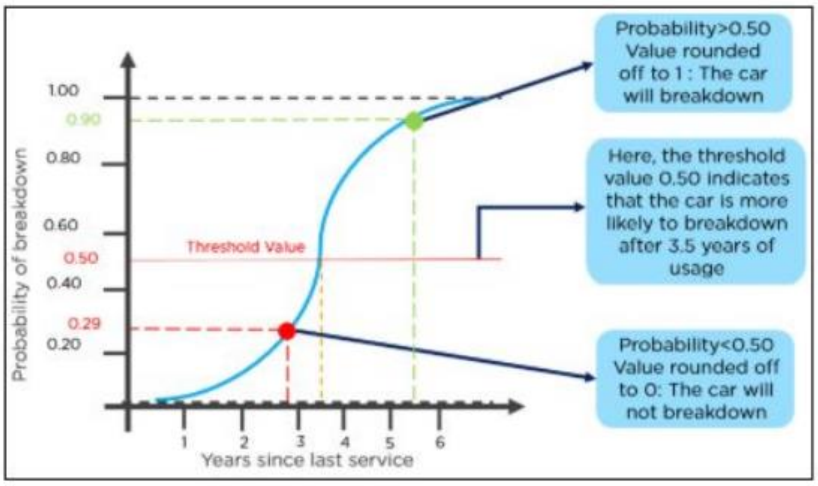


Fig 6: Graph for Logistic Regression

\

Here's how you can interpret the results in the chart to determine whether your car will break down.



Gradient Descent-

Gradient Descent (GD) is a popular optimization algorithm used in machine learning to minimise the cost function of a model. It works by iteratively adjusting the weights or model parameters in the direction of the negative gradient of the cost function until the minimum of the cost function is reached. There are numerous editions of gradient descent that vary inside the manner the step size or gaining knowledge of price is selected and the way the updates are made. here are some famous variants:

1) Batch Gradient Descent: In batch gradient descent, the entire education dataset is used to compute the gradient and update the parameters at each generation. This could be gradual for huge datasets but may also result in a more accurate model.

2) Stochastic Gradient Descent (SGD): In SGD, the best education instance is used to compute the gradient and replace the parameters at each iteration. This may be quicker than batch gradient descent however may lead to extra noise in the updates.

3) Mini-batch Gradient Descent: In mini-batch gradient descent, a small batch of schooling examples is used to compute the gradient and update

the parameters at each new release. this could be a good compromise among batch gradient descent and SGD, as it can be faster than batch gradient descent and much less noisy than SGD.

4) Momentum-based Gradient Descent: In momentum-based totally gradient descent, a momentum time period is delivered to the gradient update to assist boost up convergence and clean out the update process. this could assist in preventing the optimization manner from getting caught in local minima.

5) Conjugate Gradient Descent: Conjugate Gradient Descent (CGD) is a version of GD that may be more green in fixing huge-scale optimization problems with a symmetric and positive precise matrix. CGD calls for less memory and fewer iterations than different GD methods and can converge faster.

Multinomial Naïve Bayes -

Multinomial Naive Bayes is commonly used to assign data to clusters based on the statistics of their elements. It provides an alternative to "heavy" AI-based semantic analysis and simplifies classification of data files.

Classification aims to categorise pieces of text (i.e. documents) into groups by determining the probability that documents are in the category of other documents with the same meaning.
Each article contains a few words (i.e. keywords) that help to understand the content of the article. A class is a tag for one or more documents that refer to the same topic.

Labelling a document with one of the available groups is done by performing a statistical analysis that tests the hypothesis that a document's statement already appears in other documents in a particular category.

This increases the risk that the data will be of the same class as the classified data.

ſhe Bayesian probability $\boldsymbol{p}(\boldsymbol{C}_k \mid \boldsymbol{W})$ is computed as follows:

$$p(C_k \mid \boldsymbol{W}) = \frac{p(C_k) \times p(\boldsymbol{W} \mid C_k)}{p(\boldsymbol{W})}$$

Bayesian Probability Formulae

Random Forest

Random forest is a popular system for gaining knowledge of algorithms that belongs to the supervised mastering approach. it may be used for both type and Regression troubles in ML. it's far primarily based on the concept of ensemble studying, that's a technique of mixing multiple classifiers to solve a complicated hassle and to enhance the overall performance of the model.

Random forest is a classifier that carries a number of decision bushes on various subsets of the given dataset and takes the common to improve the predictive accuracy of that dataset." in place of counting on one selection tree, the random forest takes the prediction from every tree and based totally on most people votes of predictions, and it predicts the final output.

The greater variety of trees in the forest leads to better accuracy and prevents the hassle of overfitting.

The below diagram explains the operating of the Random forest set of rules:

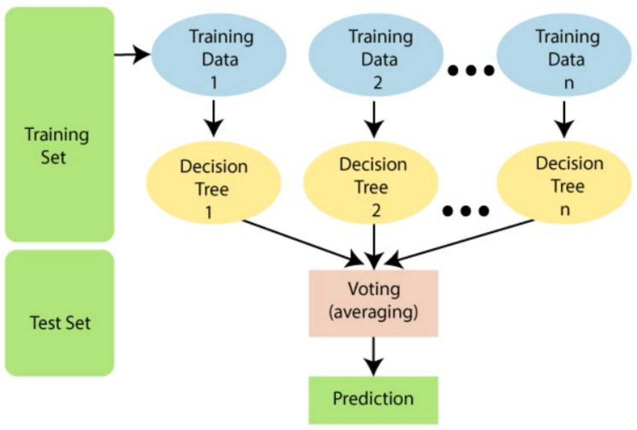


Fig 7 . Random forest

Decision Tree

Decision Tree is a Supervised Learning technique that can be used for both class and Regression problems, however in general it's far favoured for solving class troubles. it's far from a tree-dependent classifier, wherein inner nodes constitute the functions of a dataset, branches constitute the choice guidelines and every leaf node represents the outcome.

In a decision tree, there are two nodes, which are the choice Node and Leaf Node. Decision nodes are used to make any choice and feature multiple branches, whereas Leaf nodes are the output of those choices and do not contain any further branches.

it's far a graphical illustration for getting all the feasible answers to a hassle/selection based totally on given conditions.

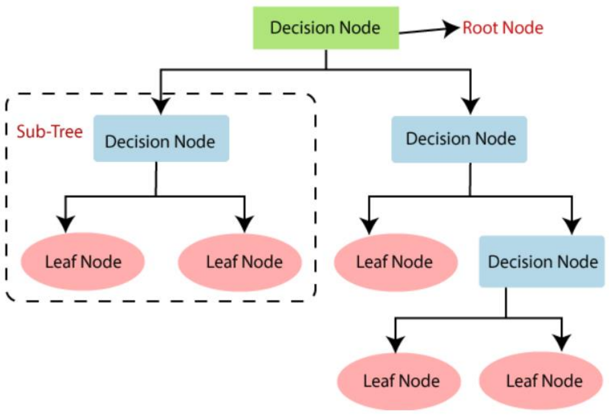


Fig 8. Decision Tree

K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm is a supervised ML algorithm that can be used for classification and estimation problems. However, it is frequently used in classification estimation problems in businesses. KNN is well defined by the following two things -

Lazy Learning Algorithm - KNN is a lazy learning algorithm because it has no higher level training and uses all data for training when dividing.

Non-Parametric Learning Algorithms - KNN is a non-parametric learning algorithm as it does not take into account the underlying data.

The K-Nearest Neighbors (KNN) algorithm uses "feature similarity" to estimate the value of the new data point, meaning that new data will be assigned a value based on the matching points in the training.

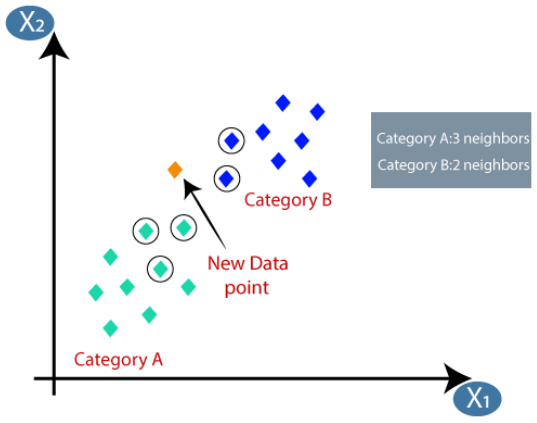


Fig 9. KNN

Support vector machine

Support Vector Machines or SVMs are one of the most popular supervised learning techniques for classification and regression problems. However, it is often used in classification problems in machine learning.
The purpose of the SVM algorithm is to create a good line or decision boundary that can divide the n-dimensional space into classes so that we can easily add new data to the class. Let's work towards the future. This well-defined boundary is called the hyperplane.
The SVM selects high points/vectors that help create an overall plane.

These conditions are called support vectors and hence the algorithm is called a vector machine. Consider the following image where two different classes are separated using a decision boundary or general plane:

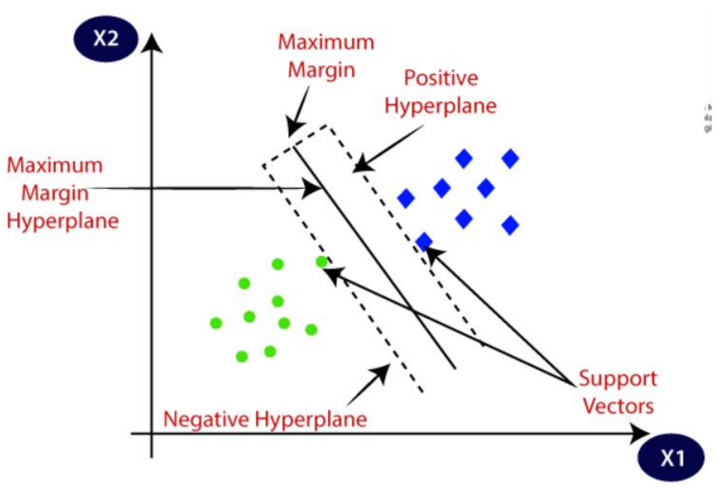


Fig 10. SVM

GridSearchCV

GridSearchCV is a method of performing hyperparameter tuning to determine the best value for a particular model. As mentioned above, the performance of the model depends on the importance of the hyperparameters. Note that there is no way to know in advance the optimal value of the hyperparameter, so ideally we should try everything possible to know the optimal value. Doing this manually can be time and resource intensive, so we use GridSearchCV to update hyperparameters.

GridSearchCV is a function in the Scikit-learn (or SK-learn) model_selection package.
The important point to remember here is that the Scikit-learn library must be installed on our computer. This function helps to loop through hyperparameters and fit your predictor (model) to your topic. Finally, we can select the best parameters from the list of hyperparameters.

Frameworks Used –

Apache Spark –

When used alone or in conjunction with other distributed computing technologies, Apache Spark is a data processing framework that can swiftly conduct operations on very large data sets and distribute operations over several machines. The world of big data and machine learning, which needs vast deployments of computer power to handle large data repositories, depends on these two characteristics.

With its simple-to-use APIs that abstract away most of the laborious work of distributed computing and large data processing, Spark also relieves developers of some of the programming load associated with these activities. Since its 2009 inception in Berkeley, Apache Spark has grown from its modest beginnings at the University of California's AMP Lab to become one of the most significant frameworks for distributed processing of large data worldwide.

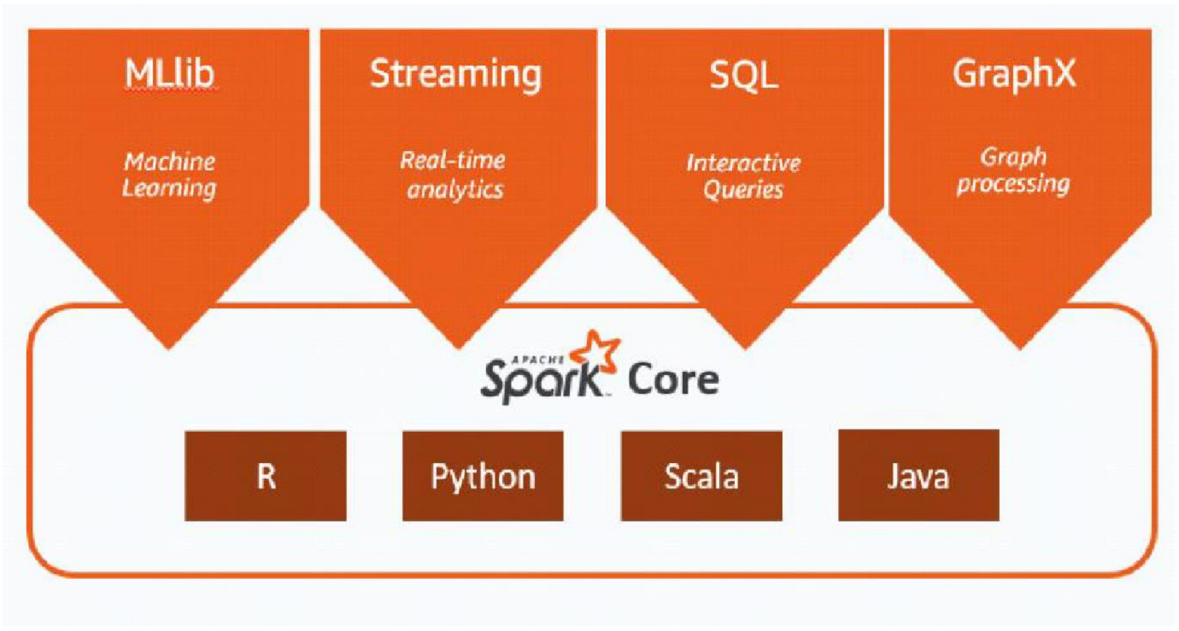


Fig 13: Spark Core

The computer languages Java, Scala, Python, and R provide native bindings for Spark, which also supports SQL, data streaming, machine learning, and graph analysis.
Banks, telecoms firms, gaming firms, governments, and all significant technological firms like Apple, Facebook, IBM, and Microsoft employ it.

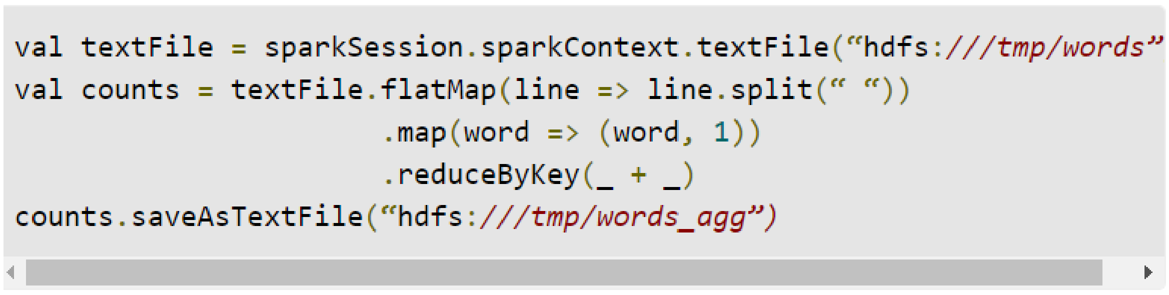
Architecture –

Fundamentally, Apache Spark applications consist of two primary parts: an executor that runs on worker nodes and executes assigned tasks, and a driver that may split user code into numerous tasks and distribute them to those nodes. comprises a number of parts. To act as a mediator between the two, some kind of cluster manager is necessary. By default, Spark may operate in standalone cluster mode. For each computer in the cluster, all that is needed is the Apache Spark framework and his JVM. To manage on-demand worker allocation, however, we advise using a more capable resource or cluster management system. However, Apache Spark can also run on Apache Mesos, Kubernetes, and Docker Swarm. In the enterprise, this typically means running on Hadoop YARN (as is the case for Cloudera and Hortonworks distributions).

Apache Spark is available as part of Amazon EMR, Google Cloud Dataproc, and Microsoft Azure HDInsight if you're seeking for a managed solution. The business where the Apache Spark creators work, Databricks, provides a wide array of managed services, such as Apache Spark clusters, streaming support, integrated web-based notebook creation, and improved cloud I/O performance. The Databricks Unified Analytics Platform is also available from us. Spark distribution using Apache. Your data processing instructions are embedded by Apache Spark into a directed acyclic graph, or DAG. Apache Spark's scheduling layer is called DAG. Identify which processes are carried out on which nodes and in what sequence.

Spark Core -

The Apache Spark API is significantly simpler for developers to use than MapReduce and other Apache Hadoop components because it hides a lot of the complexity of a distributed processing engine behind straightforward method calls. A classic illustration of this is how Apache Spark (seen below in Scala) can condense roughly 50 lines of MapReduce code to count words in a page to just a few lines:



```
val textFile = sparkSession.sparkContext.textFile(“hdfs:///tmp/words”
val counts = textFile.flatMap(line => line.split(“ “))
                     .map(word => (word, 1))
                     .reduceByKey(_ + _)
counts.saveAsTextFile(“hdfs:///tmp/words_agg”)
```

Fig 14: how almost 50 lines of MapReduce code to count words in a document can be reduced to just a few lines of Apache Spark

to create a standard language for data analysis Everyone from application developers to data scientists can benefit from Apache Spark's scalability and speed in an approachable way by offering bindings for Python and R, as well as for enterprise Java and Scala.

Spark RDD –

Resilient Distributed Datasets is the core idea behind Apache Spark (RDDs). An immutable group of things that may be dispersed throughout a computing cluster is represented by an RDD, a programming concept. Fast and scalable parallelism may be achieved by distributing RDD operations across the cluster and even running them in parallel batch processes.

Plain text files, SQL databases, NoSQL stores (like Cassandra and MongoDB), Amazon S3 buckets, and more may all be used to construct RDDs. The RDD concept forms the foundation of many of the Spark Core APIs, enabling not only the standard map and reduce functionality but also built-in support for dataset merging, filtering, sampling, and aggregation.

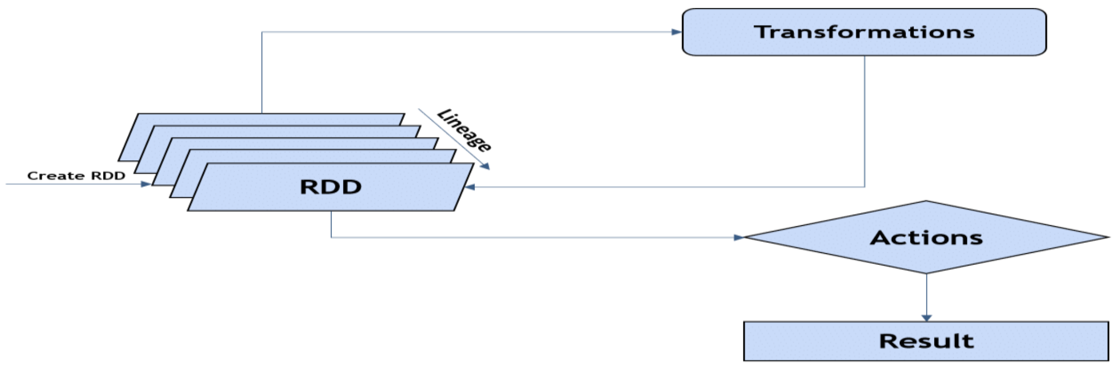


Fig 15: Spark RDD

By segmenting the Spark application into tasks and combining core driver processes that distribute the work over many running processes, Spark runs in a distributed manner. To meet the demands of your application, these executors can be scaled up or down as necessary.

Spark MLib –

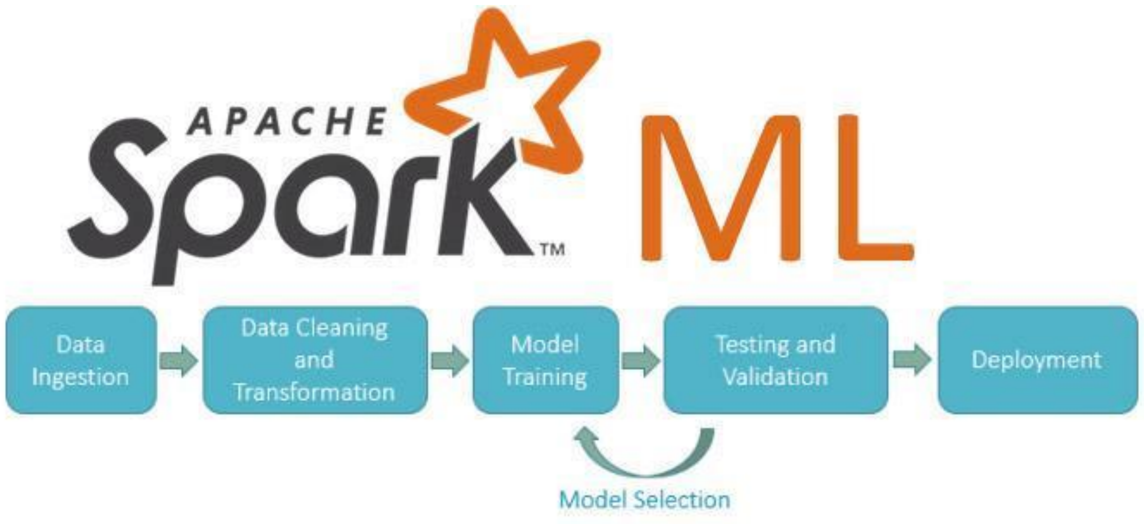


Fig 16: Flow of MLib

Additionally, Apache Spark includes libraries for using machine learning methods and chart analysis on massive amounts of data. The framework for creating machine learning pipelines included in Spark MLlib makes it simple to perform feature extraction, selection, and transformation of structured data sets. The distributed clustering and classification techniques k-means clustering and random forests are also available in MLlib and may be quickly included and removed from bespoke pipelines.

Data scientists may train models on Apache Spark with R or Python, save them using MLlib, and then import them into Java- or Scala-based pipelines for usage in production. Spark Basic machine learning techniques like classification, regression, clustering, and filtering are covered by MLlib, but deep neural network modelling and training are not supported.

Spark Streaming –

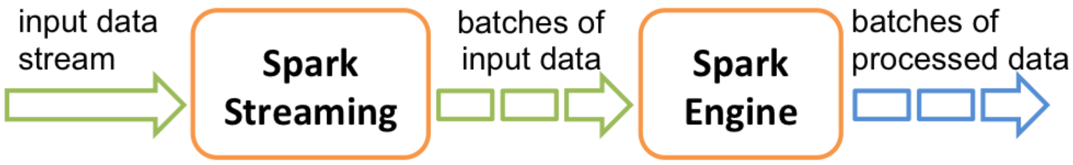


Fig 17: Spark Stream Workflow

Early improvements like Spark Streaming gave Apache Spark a leg up in settings where real-time or almost real-time processing was necessary. Batch processing and stream processing were distinct concepts in the Apache Hadoop universe. If you need batch processing, write MapReduce code; if you need real-time streaming, use Apache Storm or any similar programme. Naturally, this results in many code bases that must be kept up to date with the application domain.

Despite the fact that they are executed using entirely distinct operational features, different resources, and different frameworks. By dividing the stream into a series of consecutive micro-batches that can be controlled using the Apache Spark API, Spark Streaming extends the notion of batching from Apache Spark to streaming.

This minimises overhead for both developers and operators by allowing code for batch and streaming operations to nearly entirely use the same code running in the same framework. Everyone benefits. One criticism of Spark's streaming strategy is that micro-batching cannot compete with the performance of other streaming-enabled frameworks like Apache Storm, Apache Flink, and Apache for scenarios that call for low-latency responses to incoming data. It implies that there is a chance. Switch to the streaming-only approach from micro-batching.

Structured Streaming –

As Spark SQL was to the Spark core API, Structured Streaming (introduced in Spark 2.x) is to Spark Streaming what Spark SQL was to application development: a high-level API. This abstraction is easier. High-level APIs effectively let programmers generate limitless streaming datasets and data frames for Structure Streaming. Additionally, it resolves some very genuine issues users had with earlier frameworks. This is particularly valid when addressing event-time-related aggregation and delayed message delivery. SQL queries can be executed interactively and against live streaming data thanks to the Catalyst Query Optimizer, which processes all queries against structured streams.

Spark Streaming's micro-batch method was initially used by Structured Streaming to handle streaming data. However, the Apache Spark team added a low-latency continuous processing option to Structured Streaming in Spark 2.3, which is really amazing because it enables replies to be handled with latency as low as 1 ms. Continuous processing is still regarded as experimental as of Spark 2.4. While Continuous Streaming only supports a small number of queries, Structured Streaming is built on top of the Spark SQL engine. The future of platform-based streaming apps is structured streaming. As a result, structured streaming should be used while developing new streaming apps.

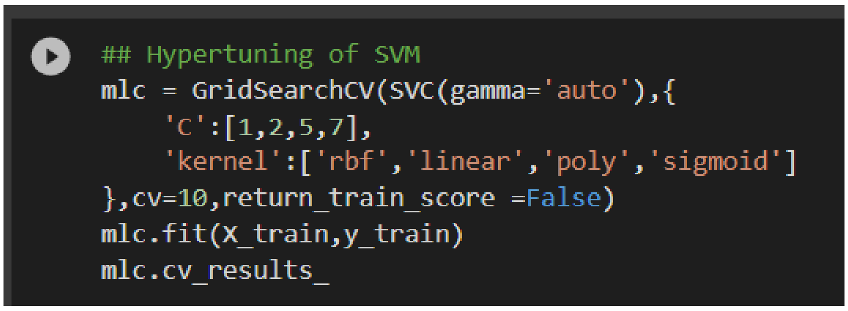
PySpark –

Fig 18: PySpark

Apache Spark has a Python interface called PySpark. It offers a PySpark shell for interactive data analysis in a distributed environment in addition to allowing you to create Spark apps using the Python API. The majority of Spark capabilities, including Spark SQL, DataFrame, Streaming, MLlib (machine learning), and Spark Core, are supported by PySpark. In the programming language Scala, Apache Spark was created. The Apache Spark community introduced a tool called PySpark to support Python in Spark. Working with RDDs in Python is also possible thanks to PySpark. This is possible because of a library called Py4j. PySpark offers a PySpark shell that initialises the Spark context and connects the Python API to the Spark core.

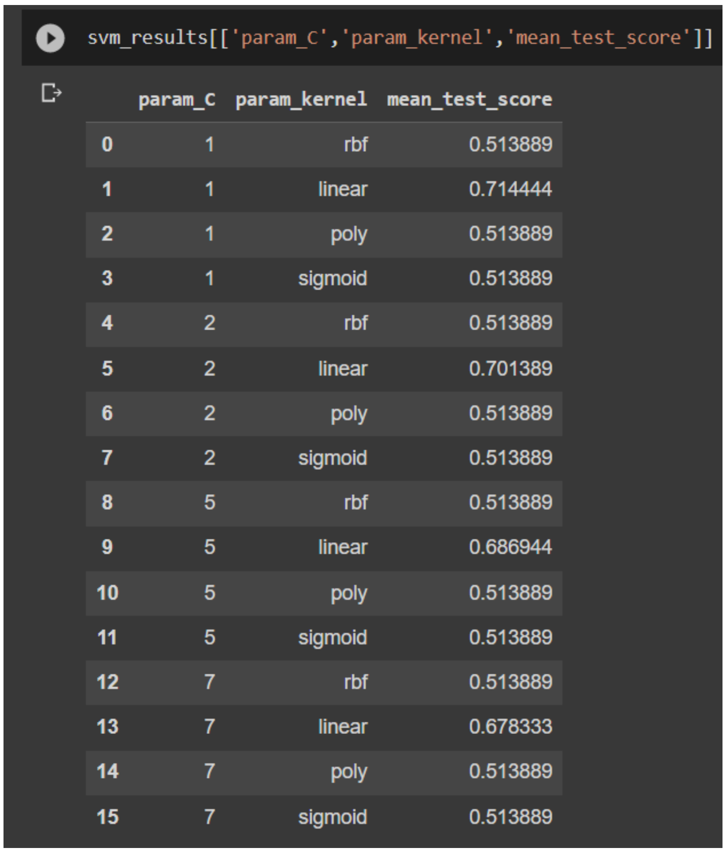
Model selection:

Firstly we use GridsearchCV to find the optimal hyperparameters by comparing the mean accuracy score for 10 fold cross validation.



```
## Hypertuning of SVM
mlc = GridSearchCV(SVC(gamma='auto'),{
    'C':[1,2,5,7],
    'kernel':['rbf','linear','poly','sigmoid']
},cv=10,return_train_score =False)
mlc.fit(X_train,y_train)
mlc.cv_results_
```

Fig. 19

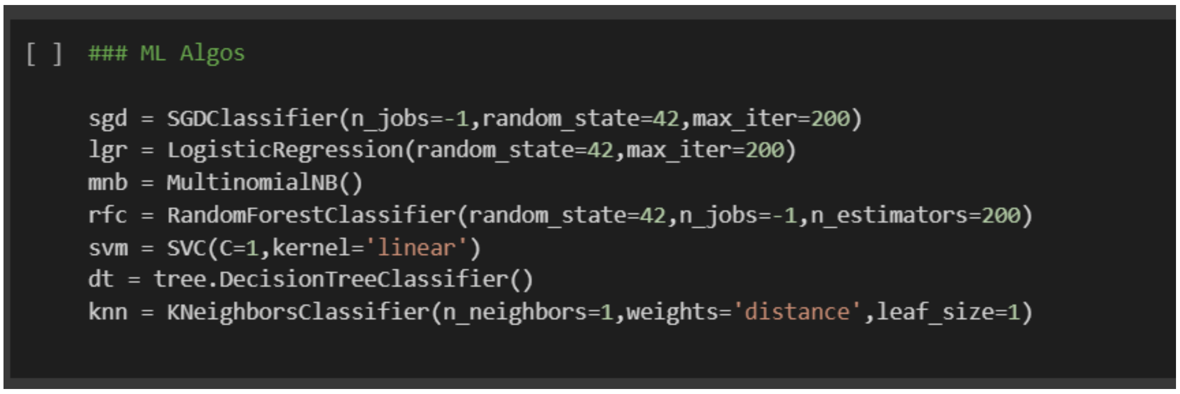


```
svm_results[['param_C','param_kernel','mean_test_score']]
```

```
    param_C  param_kernel  mean_test_score
0         1           rbf         0.513889
1         1        linear         0.714444
2         1          poly         0.513889
3         1       sigmoid         0.513889
4         2           rbf         0.513889
5         2        linear         0.701389
6         2          poly         0.513889
7         2       sigmoid         0.513889
8         5           rbf         0.513889
9         5        linear         0.686944
10        5          poly         0.513889
11        5       sigmoid         0.513889
12        7           rbf         0.513889
13        7        linear         0.678333
14        7          poly         0.513889
15        7       sigmoid         0.513889
```

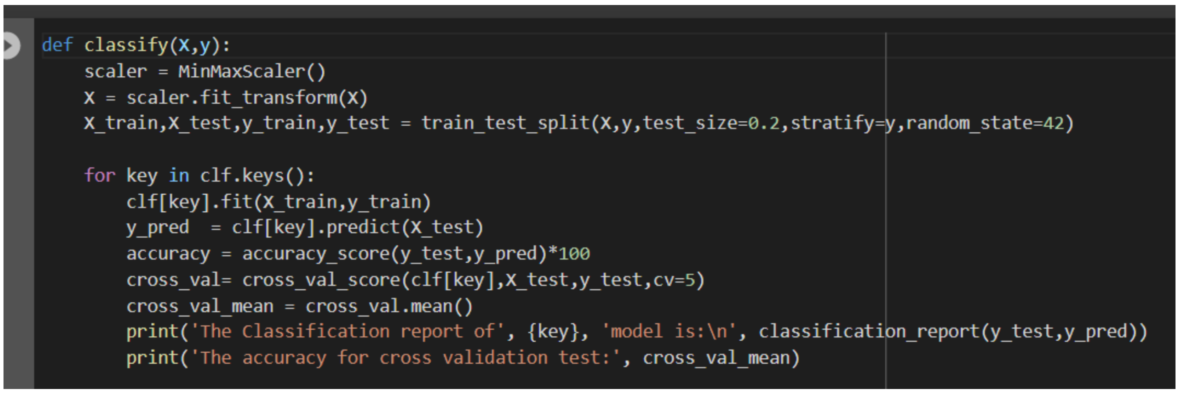
Fig. 20

Then we build the classification models for the selected algorithms



```
### ML Algos

sgd = SGDClassifier(n_jobs=-1,random_state=42,max_iter=200)
lgr = LogisticRegression(random_state=42,max_iter=200)
mnb = MultinomialNB()
rfc = RandomForestClassifier(random_state=42,n_jobs=-1,n_estimators=200)
svm = SVC(C=1,kernel='linear')
dt = tree.DecisionTreeClassifier()
knn = KNeighborsClassifier(n_neighbors=1,weights='distance',leaf_size=1)
```

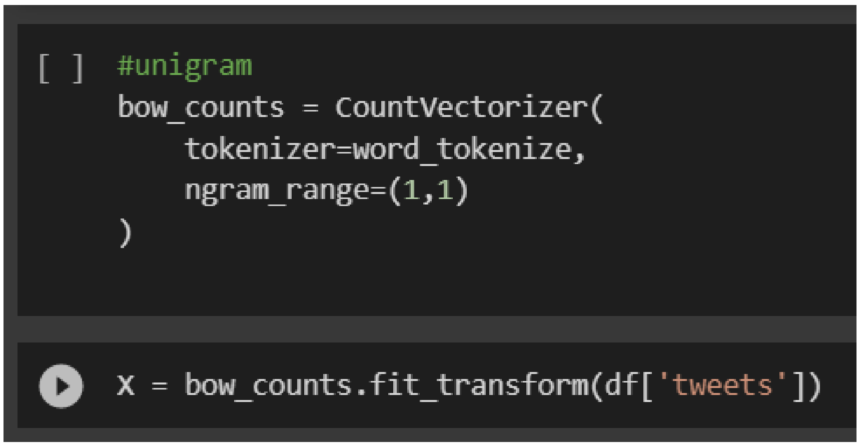
Fig. 21



```
def classify(X,y):
    scaler = MinMaxScaler()
    X = scaler.fit_transform(X)
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,stratify=y,random_state=42)

    for key in clf.keys():
        clf[key].fit(X_train,y_train)
        y_pred  = clf[key].predict(X_test)
        accuracy = accuracy_score(y_test,y_pred)*100
        cross_val= cross_val_score(clf[key],X_test,y_test,cv=5)
        cross_val_mean = cross_val.mean()
        print('The Classification report of', {key}, 'model is:\n', classification_report(y_test,y_pred))
        print('The accuracy for cross validation test:', cross_val_mean)
```

Fig. 22

Then we would test the classification metrics for testing and cross validation data for different vectorisation and bag of words such as n grams.



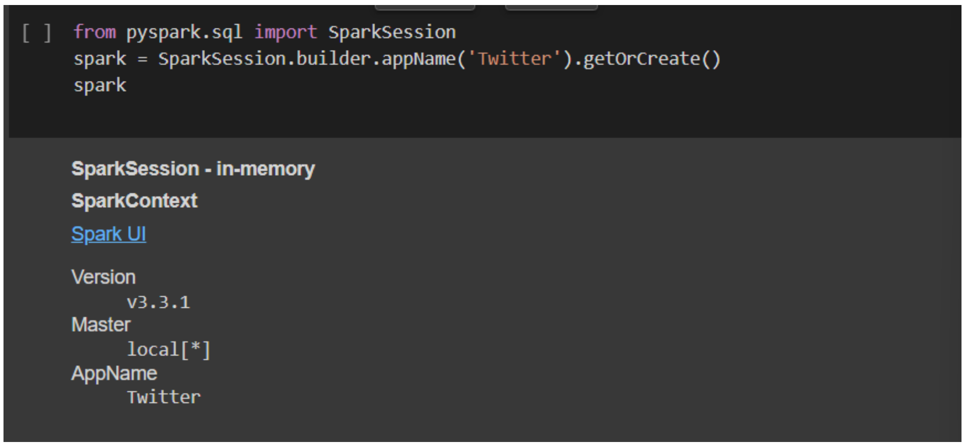
```
[ ] #unigram
    bow_counts = CountVectorizer(
        tokenizer=word_tokenize,
        ngram_range=(1,1)
    )

    X = bow_counts.fit_transform(df['tweets'])
```

Fig. 23

Text analysis using PySpark:

Creating a Spark session :

We create a spark session to load our data and use the spark session to stream the data so that we can do further operations on the same.

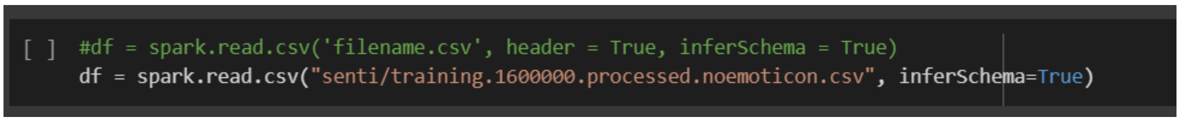


```
[ ]  from pyspark.sql import SparkSession
     spark = SparkSession.builder.appName('Twitter').getOrCreate()
     spark
```

```
SparkSession - in-memory
SparkContext
Spark UI

Version
    v3.3.1
Master
    local[*]
AppName
    Twitter
```

Fig 24

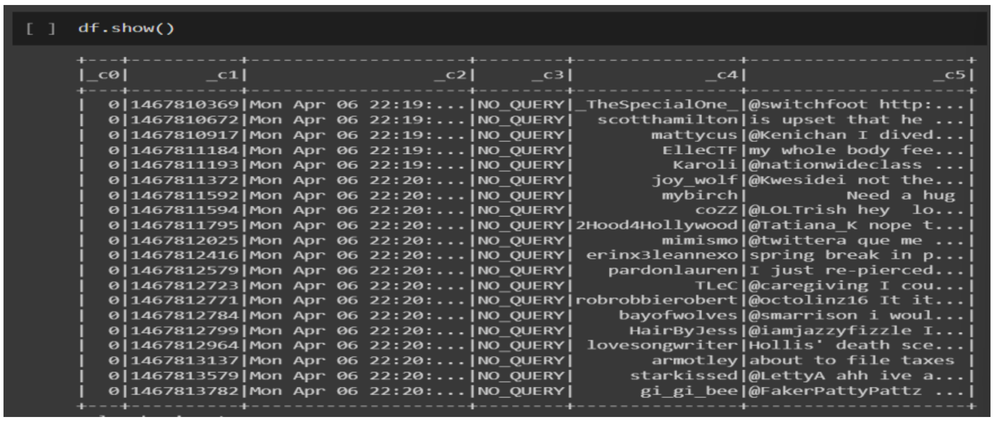
Loading the data in spark data frame:



```
[ ]  #df = spark.read.csv('filename.csv', header = True, inferSchema = True)
     df = spark.read.csv("senti/training.1600000.processed.noemoticon.csv", inferSchema=True)
```

Fig 25

First look at the data in a spark stream:

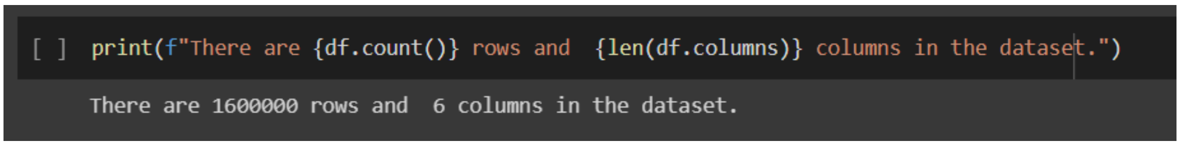


```
[ ]  df.show()
```

```
+---+----------+--------------------+--------+---------------+--------------------+
|_c0|       _c1|                 _c2|     _c3|            _c4|                 _c5|
+---+----------+--------------------+--------+---------------+--------------------+
|  0|1467810369|Mon Apr 06 22:19:...|NO_QUERY|_TheSpecialOne_|@switchfoot http:...|
|  0|1467810672|Mon Apr 06 22:19:...|NO_QUERY|  scotthamilton|is upset that he ...|
|  0|1467810917|Mon Apr 06 22:19:...|NO_QUERY|       mattycus|@Kenichan I dived...|
|  0|1467811184|Mon Apr 06 22:19:...|NO_QUERY|        ElleCTF|my whole body fee...|
|  0|1467811193|Mon Apr 06 22:19:...|NO_QUERY|         Karoli|@nationwideclass ...|
|  0|1467811372|Mon Apr 06 22:20:...|NO_QUERY|       joy_wolf|@Kwesidei not the...|
|  0|1467811592|Mon Apr 06 22:20:...|NO_QUERY|        mybirch|         Need a hug |
|  0|1467811594|Mon Apr 06 22:20:...|NO_QUERY|           coZZ|@LOLTrish hey  lo...|
|  0|1467811795|Mon Apr 06 22:20:...|NO_QUERY|2Hood4Hollywood|@Tatiana_K nope t...|
|  0|1467812025|Mon Apr 06 22:20:...|NO_QUERY|        mimismo|@twittera que me ...|
|  0|1467812416|Mon Apr 06 22:20:...|NO_QUERY| erinx3leannexo|spring break in p...|
|  0|1467812579|Mon Apr 06 22:20:...|NO_QUERY|   pardonlauren|I just re-pierced...|
|  0|1467812723|Mon Apr 06 22:20:...|NO_QUERY|           TLeC|@caregiving I cou...|
|  0|1467812771|Mon Apr 06 22:20:...|NO_QUERY|robrobbierobert|@octolinz16 It it...|
|  0|1467812784|Mon Apr 06 22:20:...|NO_QUERY|    bayofwolves|@smarrison i woul...|
|  0|1467812799|Mon Apr 06 22:20:...|NO_QUERY|     HairByJess|@iamjazzyfizzle I...|
|  0|1467812964|Mon Apr 06 22:20:...|NO_QUERY| lovesongwriter|Hollis' death sce...|
|  0|1467813137|Mon Apr 06 22:20:...|NO_QUERY|       armotley|about to file taxes |
|  0|1467813579|Mon Apr 06 22:20:...|NO_QUERY|     starkissed|@LettyA ahh ive a...|
|  0|1467813782|Mon Apr 06 22:20:...|NO_QUERY|      gi_gi_bee|@FakerPattyPattz ...|
+---+----------+--------------------+--------+---------------+--------------------+
```

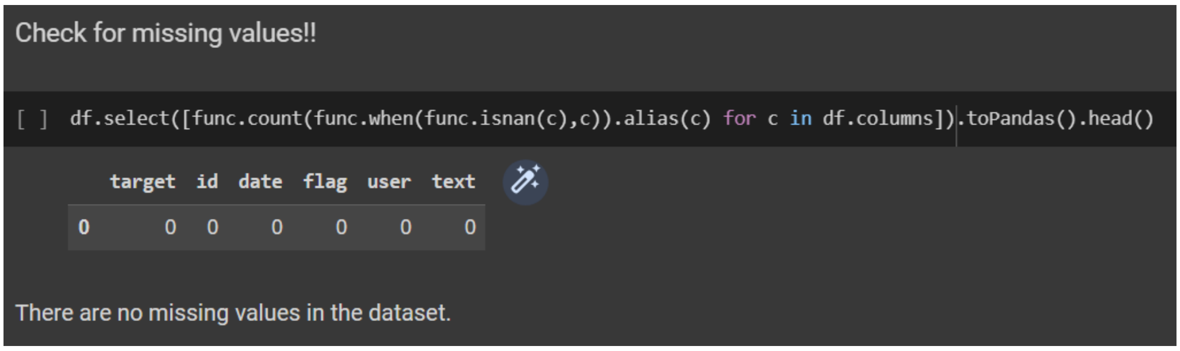
Fig 26

Performing Exploratory Data Analysis:



```
[ ]  print(f"There are {df.count()} rows and  {len(df.columns)} columns in the dataset.")

     There are 1600000 rows and  6 columns in the dataset.
```

Fig 27



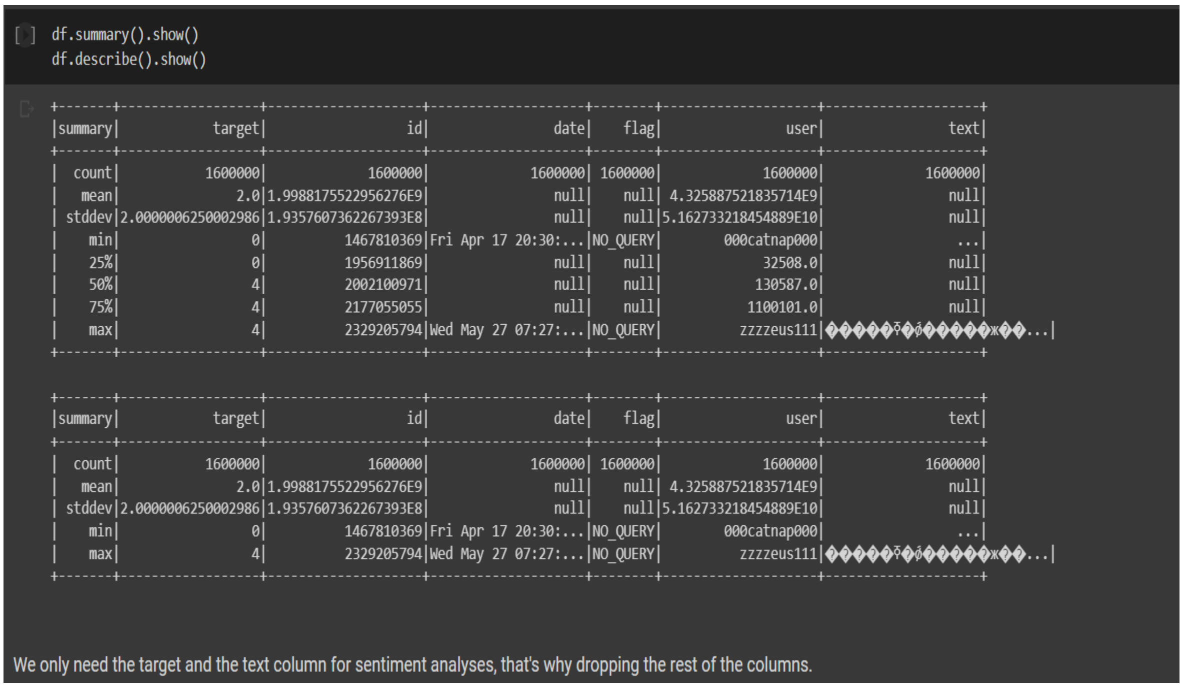
Check for missing values!!

```
[ ]  df.select([func.count(func.when(func.isnan(c),c)).alias(c) for c in df.columns]).toPandas().head()
```

	target	id	date	flag	user	text
0	0	0	0	0	0	0

There are no missing values in the dataset.

Fig 28



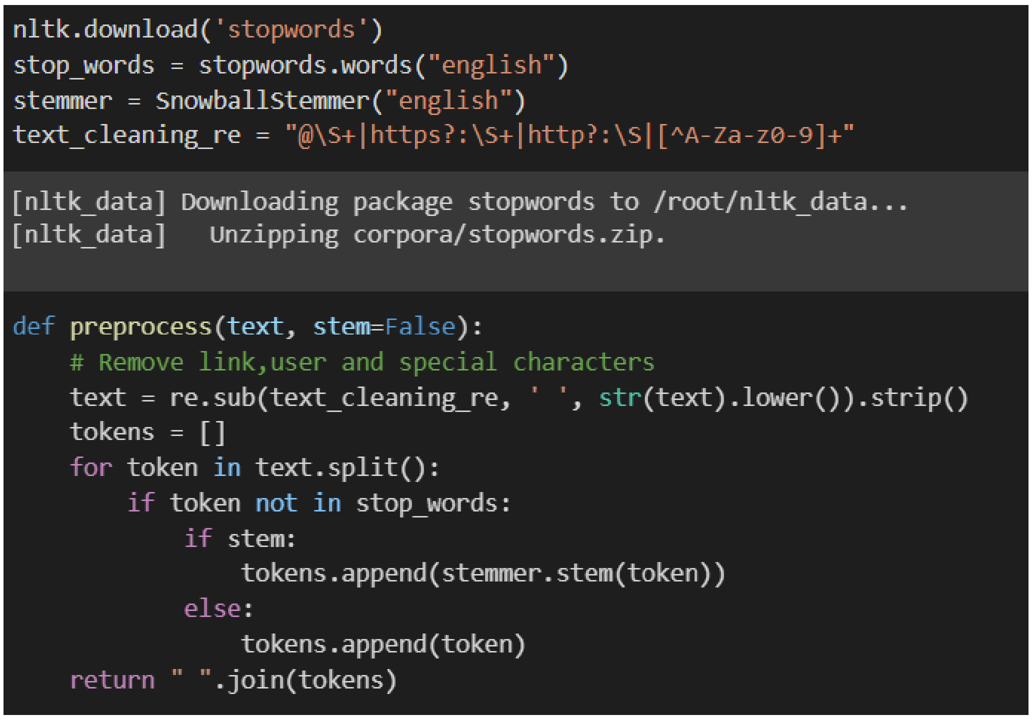
```
[ ]  df.summary().show()
     df.describe().show()
```

```
+-------+------------------+--------------------+--------------------+--------+--------------------+--------------------+
|summary|            target|                  id|                date|    flag|                user|                text|
+-------+------------------+--------------------+--------------------+--------+--------------------+--------------------+
|  count|           1600000|             1600000|             1600000| 1600000|             1600000|             1600000|
|   mean|               2.0|1.9988175522956276E9|                null|    null| 4.325887521835714E9|                null|
| stddev|2.0000006250002986|1.9357607362267393E8|                null|    null|5.162733218454889E10|                null|
|    min|                 0|          1467810369|Fri Apr 17 20:30:...|NO_QUERY|        000catnap000|                 ...|
|    25%|                 0|          1956911869|                null|    null|             32508.0|                null|
|    50%|                 4|          2002100971|                null|    null|            130587.0|                null|
|    75%|                 4|          2177055055|                null|    null|           1100101.0|                null|
|    max|                 4|          2329205794|Wed May 27 07:27:...|NO_QUERY|          zzzzeus111|�����õ�ő������ж��...|
+-------+------------------+--------------------+--------------------+--------+--------------------+--------------------+

+-------+------------------+--------------------+--------------------+--------+--------------------+--------------------+
|summary|            target|                  id|                date|    flag|                user|                text|
+-------+------------------+--------------------+--------------------+--------+--------------------+--------------------+
|  count|           1600000|             1600000|             1600000| 1600000|             1600000|             1600000|
|   mean|               2.0|1.9988175522956276E9|                null|    null| 4.325887521835714E9|                null|
| stddev|2.0000006250002986|1.9357607362267393E8|                null|    null|5.162733218454889E10|                null|
|    min|                 0|          1467810369|Fri Apr 17 20:30:...|NO_QUERY|        000catnap000|                 ...|
|    max|                 4|          2329205794|Wed May 27 07:27:...|NO_QUERY|          zzzzeus111|�����õ�ő������ж��...|
+-------+------------------+--------------------+--------------------+--------+--------------------+--------------------+
```

We only need the target and the text column for sentiment analyses, that's why dropping the rest of the columns.

Fig 29

Text Pre-processing:



```
nltk.download('stopwords')
stop_words = stopwords.words("english")
stemmer = SnowballStemmer("english")
text_cleaning_re = "@\S+|https?:\S+|http?:\S|[^A-Za-z0-9]+"
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
def preprocess(text, stem=False):
    # Remove link,user and special characters
    text = re.sub(text_cleaning_re, ' ', str(text).lower()).strip()
    tokens = []
    for token in text.split():
        if token not in stop_words:
            if stem:
                tokens.append(stemmer.stem(token))
            else:
                tokens.append(token)
    return " ".join(tokens)
```

Fig 30

Displaying Word Cloud:

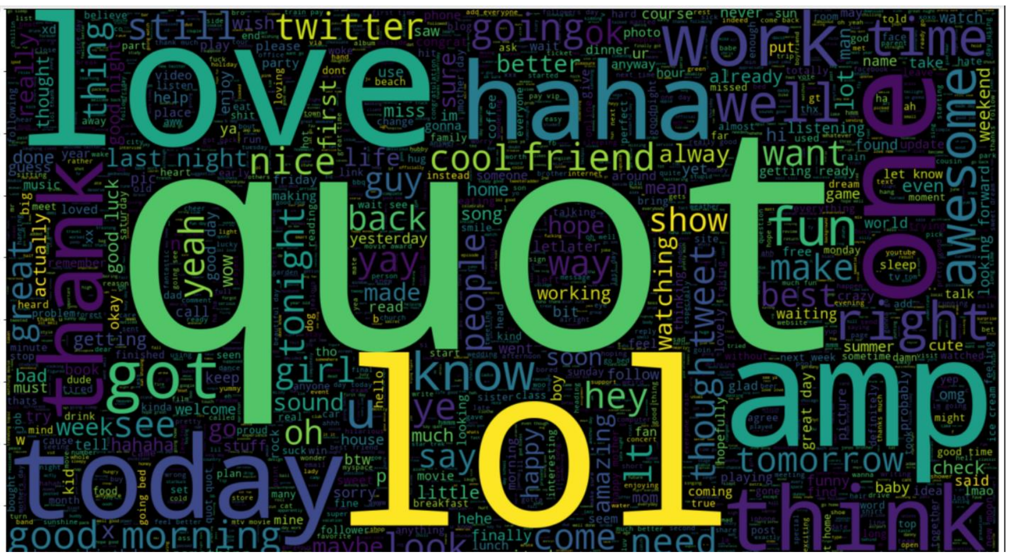
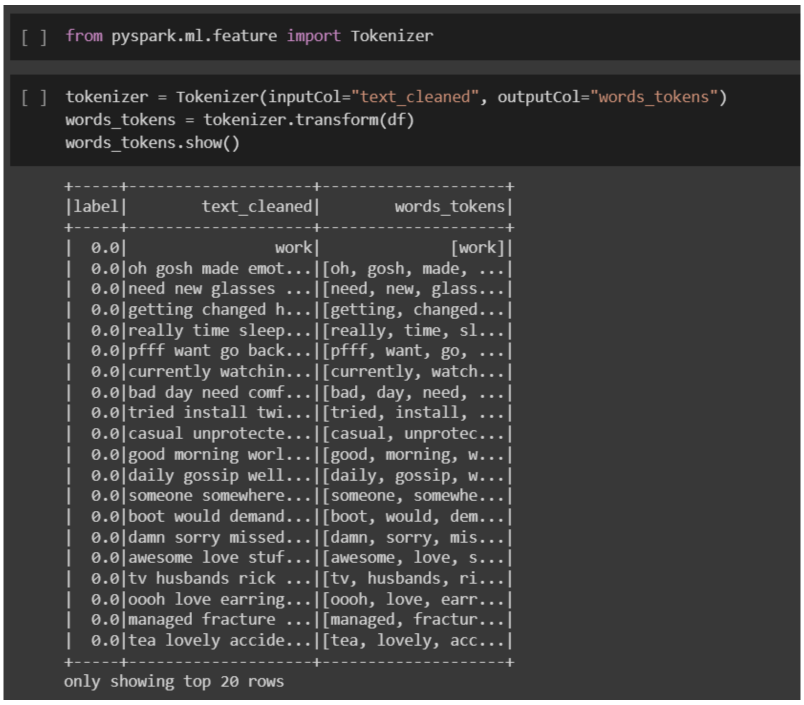


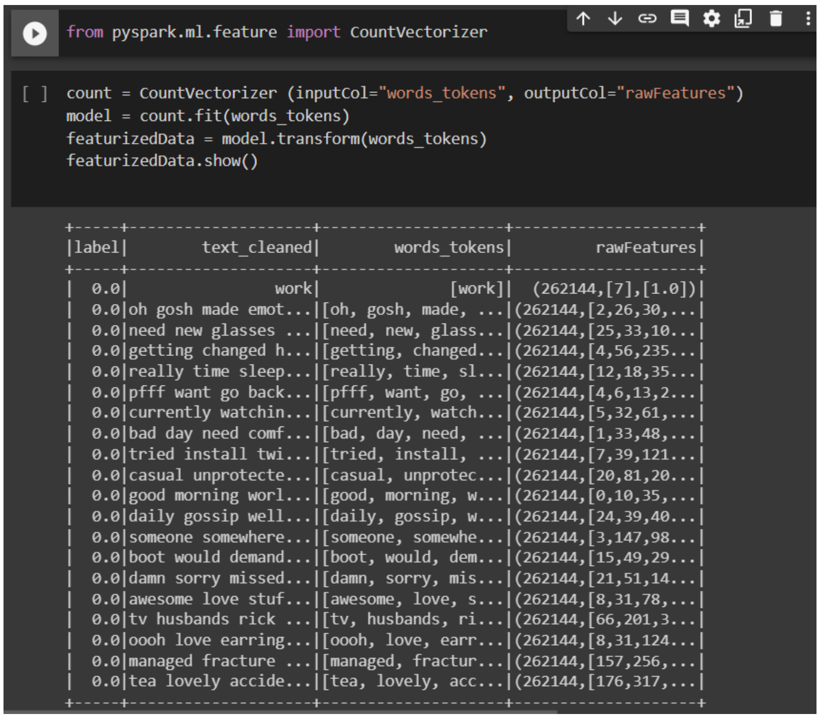
Fig 31

Tokenizing and Applying Vectorization the Text :



```
[ ] from pyspark.ml.feature import Tokenizer

[ ] tokenizer = Tokenizer(inputCol="text_cleaned", outputCol="words_tokens")
    words_tokens = tokenizer.transform(df)
    words_tokens.show()

    +-----+--------------------+--------------------+
    |label|        text_cleaned|        words_tokens|
    +-----+--------------------+--------------------+
    |  0.0|                work|              [work]|
    |  0.0|oh gosh made emot...|[oh, gosh, made, ...|
    |  0.0|need new glasses ...|[need, new, glass...|
    |  0.0|getting changed h...|[getting, changed...|
    |  0.0|really time sleep...|[really, time, sl...|
    |  0.0|pfff want go back...|[pfff, want, go, ...|
    |  0.0|currently watchin...|[currently, watch...|
    |  0.0|bad day need comf...|[bad, day, need, ...|
    |  0.0|tried install twi...|[tried, install, ...|
    |  0.0|casual unprotecte...|[casual, unprotec...|
    |  0.0|good morning worl...|[good, morning, w...|
    |  0.0|daily gossip well...|[daily, gossip, w...|
    |  0.0|someone somewhere...|[someone, somewhe...|
    |  0.0|boot would demand...|[boot, would, dem...|
    |  0.0|damn sorry missed...|[damn, sorry, mis...|
    |  0.0|awesome love stuf...|[awesome, love, s...|
    |  0.0|tv husbands rick ...|[tv, husbands, ri...|
    |  0.0|oooh love earring...|[oooh, love, earr...|
    |  0.0|managed fracture ...|[managed, fractur...|
    |  0.0|tea lovely accide...|[tea, lovely, acc...|
    +-----+--------------------+--------------------+
    only showing top 20 rows
```

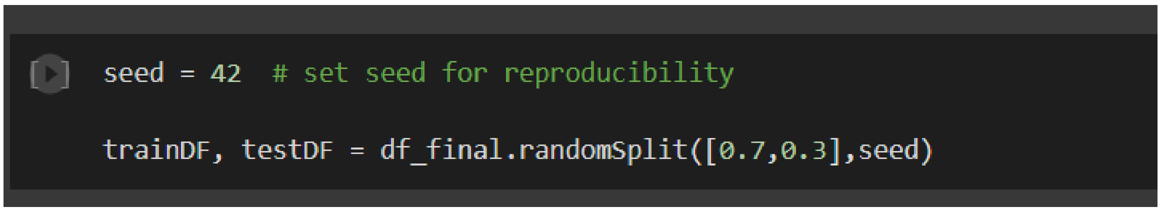
Fig 32



```
from pyspark.ml.feature import CountVectorizer

[ ] count = CountVectorizer (inputCol="words_tokens", outputCol="rawFeatures")
    model = count.fit(words_tokens)
    featurizedData = model.transform(words_tokens)
    featurizedData.show()

    +-----+--------------------+--------------------+--------------------+
    |label|        text_cleaned|        words_tokens|         rawFeatures|
    +-----+--------------------+--------------------+--------------------+
    |  0.0|                work|              [work]|  (262144,[7],[1.0])|
    |  0.0|oh gosh made emot...|[oh, gosh, made, ...|(262144,[2,26,30,...|
    |  0.0|need new glasses ...|[need, new, glass...|(262144,[25,33,10...|
    |  0.0|getting changed h...|[getting, changed...|(262144,[4,56,235...|
    |  0.0|really time sleep...|[really, time, sl...|(262144,[12,18,35...|
    |  0.0|pfff want go back...|[pfff, want, go, ...|(262144,[4,6,13,2...|
    |  0.0|currently watchin...|[currently, watch...|(262144,[5,32,61,...|
    |  0.0|bad day need comf...|[bad, day, need, ...|(262144,[1,33,48,...|
    |  0.0|tried install twi...|[tried, install, ...|(262144,[7,39,121...|
    |  0.0|casual unprotecte...|[casual, unprotec...|(262144,[20,81,20...|
    |  0.0|good morning worl...|[good, morning, w...|(262144,[0,10,35,...|
    |  0.0|daily gossip well...|[daily, gossip, w...|(262144,[24,39,40...|
    |  0.0|someone somewhere...|[someone, somewhe...|(262144,[3,147,98...|
    |  0.0|boot would demand...|[boot, would, dem...|(262144,[15,49,29...|
    |  0.0|damn sorry missed...|[damn, sorry, mis...|(262144,[21,51,14...|
    |  0.0|awesome love stuf...|[awesome, love, s...|(262144,[8,31,78,...|
    |  0.0|tv husbands rick ...|[tv, husbands, ri...|(262144,[66,201,3...|
    |  0.0|oooh love earring...|[oooh, love, earr...|(262144,[8,31,124...|
    |  0.0|managed fracture ...|[managed, fractur...|(262144,[157,256,...|
    |  0.0|tea lovely accide...|[tea, lovely, acc...|(262144,[176,317,...|
    +-----+--------------------+--------------------+--------------------+
```

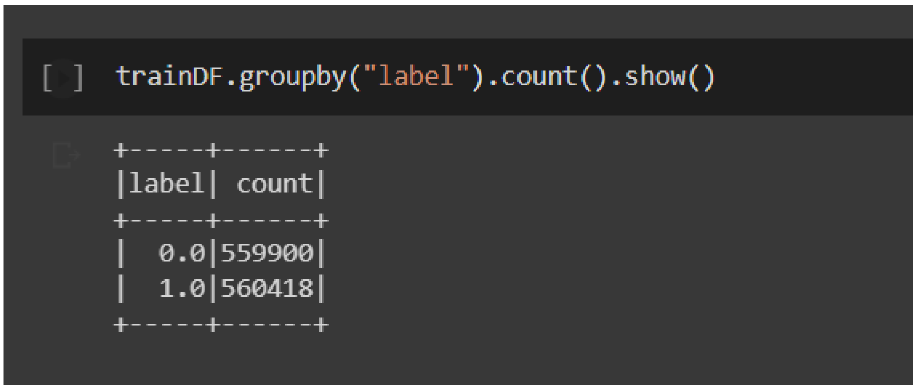
Fig 33

Splitting the Data:



```
seed = 42  # set seed for reproducibility

trainDF, testDF = df_final.randomSplit([0.7,0.3],seed)
```

Fig 34



```
trainDF.groupby("label").count().show()
```

```
+-----+------+
|label| count|
+-----+------+
|  0.0|559900|
|  1.0|560418|
+-----+------+
```

Fig 35

Deployment of the model:

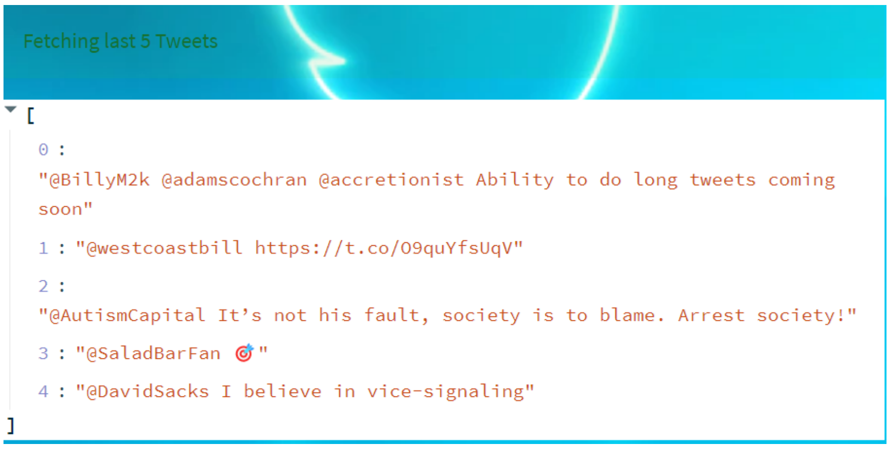
Streamlit

An open-source Python framework called Streamlit is used to create web applications for machine learning and data science. Using Streamlit, we can quickly design and launch web applications. You can use Streamlit to create apps in the same manner that you create Python code. Working on the interactive cycle of coding and watching outcomes on the web app is made simple by Streamlit.

Our WebApp is called TwFeels which is a platform where one can analyze a twitter handle and check the quality of tweets. The App has two functionalities:

1) Tweet Analyzer:

- **Show Recent Tweets**: Shows the latest five tweets tweeted out by the provided handle



Fetching last 5 Tweets

```
[
  0 :
  "@BillyM2k @adamscochran @accretionist Ability to do long tweets coming
  soon"
  1 : "@westcoastbill https://t.co/O9quYfsUqV"
  2 :
  "@AutismCapital It’s not his fault, society is to blame. Arrest society!"
  3 : "@SaladBarFan 🎯"
  4 : "@DavidSacks I believe in vice-signaling"
]
```

Fig. 36

- **Generate WordCloud:** Generates a wordcloud of the words that occur the most in the last 100 tweets by the provided handle

Generating Word Cloud

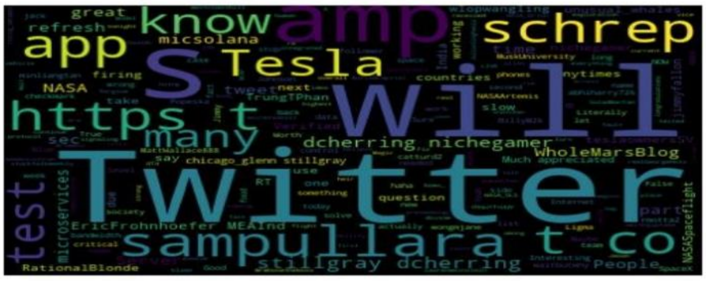
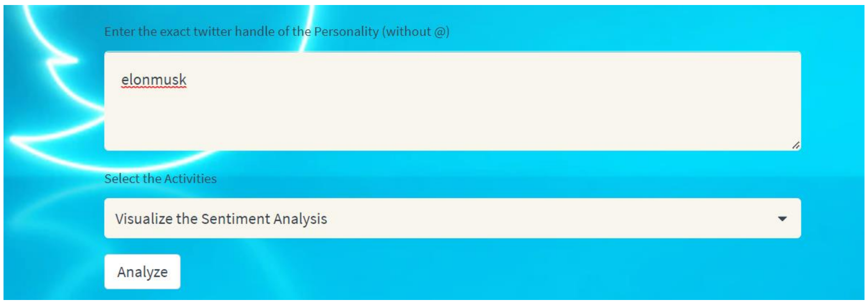


Fig. 37

- **Visualise the Sentiment Analysis:** Plots a barplot of the last 100 tweets by the provided handle classified into Positive, Negative and Neutral.



Enter the exact twitter handle of the Personality (without @)

elonmusk

Select the Activities

Visualize the Sentiment Analysis

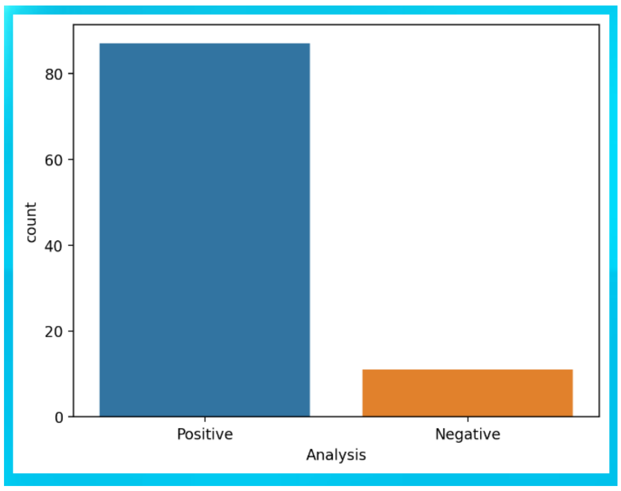
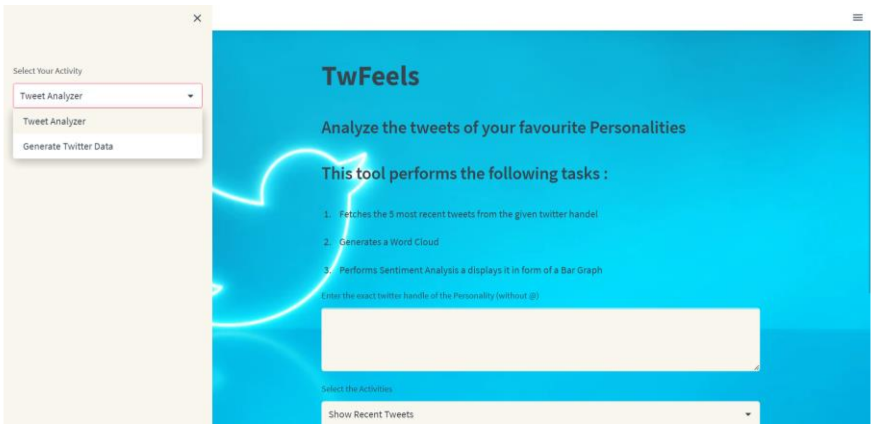


Fig. 38



Select Your Activity

Tweet Analyzer

TwFeels

Analyze the tweets of your favourite Personalities

This tool performs the following tasks :

1. Fetches the 5 most recent tweets from the given twitter handel
2. Generates a Word Cloud
3. Performs Sentiment Analysis a displays it in form of a Bar Graph

Enter the exact twitter handle of the Personality (without @)

Select the Activities

Show Recent Tweets

Fig. 39 Screenshot of the homepage

Chapter 4

Results and Performance

As we went on with applying feature extraction methods on all the models and checked the testing and cross validation performance. We went on with accuracy as our main metric to distinguish and select our best model of vectorisation metric.

Accuracy:

The percentage of accurate predictions to all predictions is known as classification accuracy, and it is calculated as follows.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

As we saw in our analysis our dataset had almost the same number of observations for Positive and negative tweets, therefore accuracy will be a good metric to check the performance of the model.

As we will observe in the tables following below the accuracies for all the models vary with different features extraction methods.

Feature extraction method	Testing performance (Accuracy)	Cross Validation performance (Accuracy)
Unigram	0.68	0.60
Bigram	0.70	0.60
Trigram	0.70	0.61

Table 3. Gradient descent

Feature extraction method	Testing performance (Accuracy)	Cross Validation performance (Accuracy)
Unigram	0.70	0.64
Bigram	0.71	0.63
Trigram	0.72	0.635

Table 4. Logistic regression

Feature extraction method	Testing performance (Accuracy)	Cross Validation performance (Accuracy)
Unigram	0.68	0.62
Bigram	0.68	0.61
Trigram	0.67	0.59

Table 5. Naive bayes

Feature extraction method	Testing performance (Accuracy)	Cross Validation performance (Accuracy)

Unigram	0.71	0.64
Bigram	0.71	0.63
Trigram	0.71	0.61

Table 6. Random Forest

Feature extraction method	Testing performance (Accuracy)	Cross Validation performance (Accuracy)
Unigram	0.69	0.615
Bigram	0.70	0.61
Trigram	0.72	0.63

Table 7. SVC

Feature extraction method	Testing performance (Accuracy)	Cross Validation performance (Accuracy)
Unigram	0.66	0.59
Bigram	0.65	0.57
Trigram	0.64	0.60

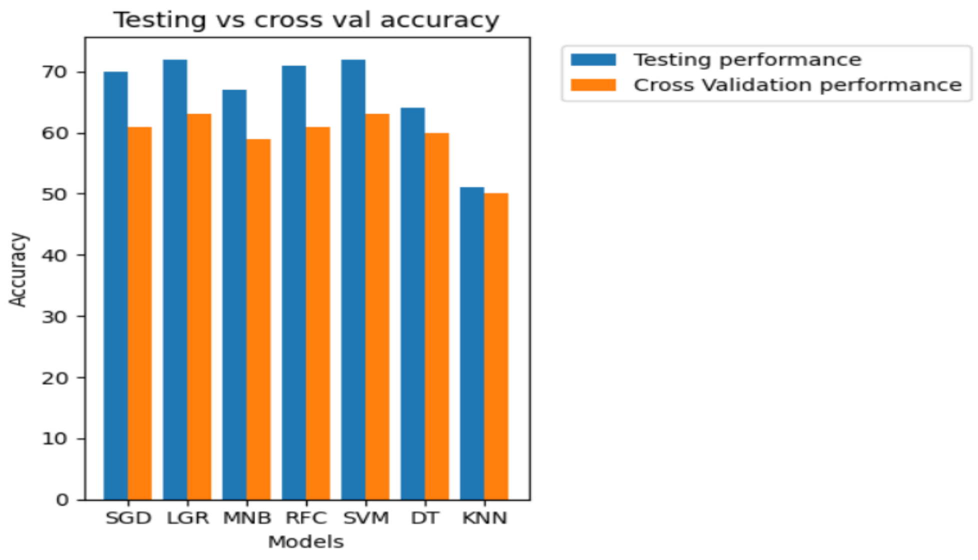
Table 8. Decision Tree

Feature extraction method	Testing performance (Accuracy)	Cross Validation performance (Accuracy)
Unigram	0.58	0.55

Bigram	0.52	0.51
Trigram	0.51	0.50

Table 9. KNN

So, as we can see the best performing models are Logistic regression (LR) with Trigram feature extraction technique and Support vector machine(SVM) with Trigram feature extraction technique, both of them tied at 72 % accuracy measure.



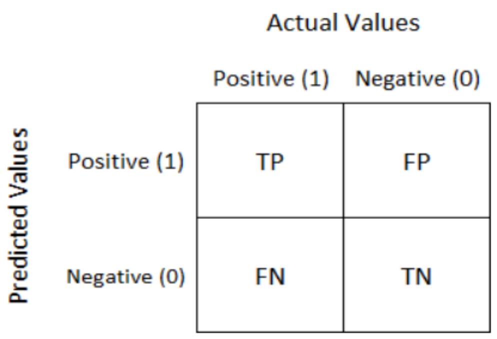
Graph 3. Testing performance vs Cross val performance

As we can see from fig, both SVM and LR have the same variance when comparing testing and cross validation accuracies. Therefore, any one of the two can be used as the final model. So we proceeded to use Logistic Regression with the Spark Stream and proceed to get these results after the creation of the model.

Confusion Matrix:

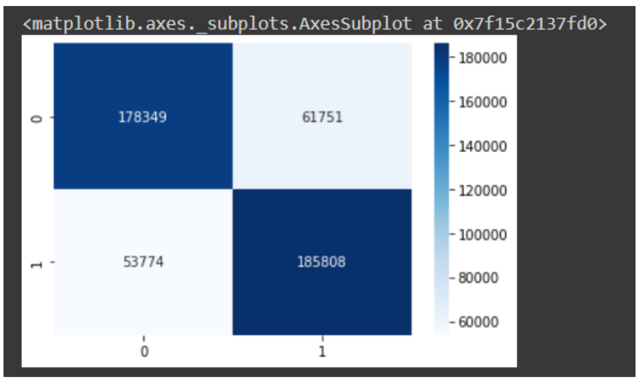
The confusion Matrix details the model's overall performance.

For instance, if there is a difficulty with binary categorization. We can categorise it as either YES or NO. Further, The project's usage of a Logistic Regression classifier predicts a class for a given input Tweet. We obtain the following confusion matrix after testing our model:



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Confusion matrix for our final model:

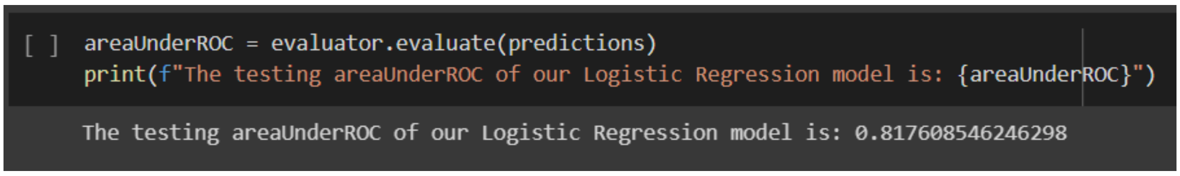


```
<matplotlib.axes._subplots.AxesSubplot at 0x7f15c2137fd0>
```

Fig. 40 Confusion matrix

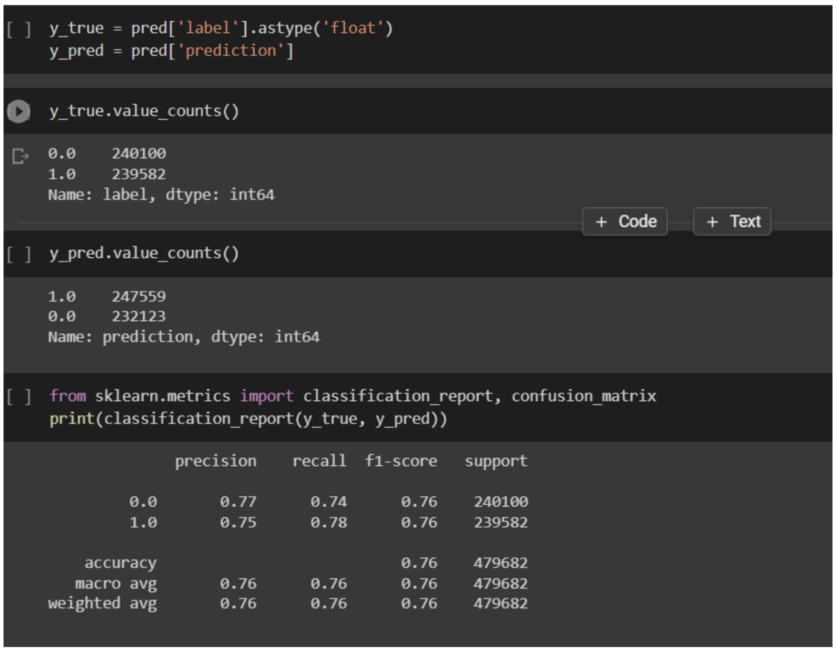
As we can see it is a fine model as we have very large True Positive and True Negative values as compared to False Positives and False Negatives.

Some other results:



```
[ ]  areaUnderROC = evaluator.evaluate(predictions)
     print(f"The testing areaUnderROC of our Logistic Regression model is: {areaUnderROC}")

     The testing areaUnderROC of our Logistic Regression model is: 0.817608546246298
```

Fig. 41



```
[ ] y_true = pred['label'].astype('float')
    y_pred = pred['prediction']

    y_true.value_counts()

    0.0    240100
    1.0    239582
    Name: label, dtype: int64

[ ] y_pred.value_counts()

    1.0    247559
    0.0    232123
    Name: prediction, dtype: int64

[ ] from sklearn.metrics import classification_report, confusion_matrix
    print(classification_report(y_true, y_pred))

                  precision    recall  f1-score   support

             0.0       0.77      0.74      0.76    240100
             1.0       0.75      0.78      0.76    239582

        accuracy                           0.76    479682
       macro avg       0.76      0.76      0.76    479682
    weighted avg       0.76      0.76      0.76    479682
```

Fig. 42

Result:

Support Vector Classifier with trigram is the best performing algorithm for the used dataset and produce a optimal model for our webapp.

Chapter 5

Conclusion

Natural Language Processing, or NLP, is a subfield of artificial intelligence that organises communication between speakers of a common language and personal computers.
Perusing, interpreting, comprehending, and understanding human dialects in a significant way is a clear goal of NLP. Most NLP techniques rely on AI to extract value from human dialects.

This report has presented a system for real-time sentiment prediction on Twitter streaming data for user tweets. The proposed system has been developed using Twitter Streaming API, Apache Spark, and regular machine learning models. It consists of two components, namely, developing an offline sentiment analysis model and an online prediction pipeline. The offline model component is used to obtain the best machine learning model, which will be used on the online sentiment prediction using the best procured model using Stream lit

We have evaluated seven machine learning models, which are, Gradient descent, Logistic regression, Decision Trees, KNN, Random Forest, Naïve Bayes and SVM, using the Twitter based tweet dataset. The empirical results have proved that the LR model and SVM model using the trigram feature extraction method has achieved the best performance compared with the other models with 72 % prediction accuracy. However, with the hardware configuration the project performed, results for n > 4 were not able to be computed due to ram power restrictions. The online prediction pipeline component is used to predict the user tweets' sentiment polarity in real-time. It has used the Twitter Streaming API to collect streaming tweets of a particular user in real-time then evaluated based on our offline model.

Future Work -

As you may have noticed, there are still many tweets misclassified as the accuracy is 70 per-cent. This is because tweets are based on a language common to our languages. The language does not follow traditional grammatical rules, nor does it belong to a single writing style. It might mean something else, but it's full of slang that expresses completely different emotions, so you either trick the classifier into assigning it to a polarity that doesn't belong, or you often end up expressing conflicting polarities, and then Classify as Neutral. This is a small part of a bigger problem. This issue affects web app accuracy.

Therefore, to solve this problem, we need to support more and more machine learning frameworks that support web-based scripting languages such as jQuery, Javascript, and PHP. However, developing more efficient unsupervised machine learning models based on neural networks and supporting web-based apps to build more such apps and make them available to the public for my own use. In order to do so, we need to do more research and development.
We plan to use bigger datasets and look into the effects of adding more distinct features to the input vector in the near future.

Applications -

Fundamentally, the sentiment analysis app proves to be an asset in predicting the image of the brand and its products. It permits the brand to:

- Analyse the image of their brand in the opinion of the public;
- Predicting and informing the brand beforehand about its future downfall;

- Discover examples and patterns;
- watch out for the introduction by the influencers.

This permits us to conform to the situation in a like manner and give the item a legitimate introduction.

In general, sentiment analysis can be utilised to:

- Mechanize media observing procedure and the going with ready framework
- Screen notices or audits of the brand on various stages (web journals, online life, survey destinations, discussions, and so forth.)
- Order direness of notices as per the significance scoring (i.e., which stage, sort of client is crucial to the brand)

The entire activity comprises of two phases:

- At the underlying stage, the organisation responds to the approaching outcomes and adjusts.
- After some time, supposition investigation can change the strategy from responding to dealing with the observation.

REFERENCES

[1] H. Elzayady, K. M. Badran, and G. I. Salama, “Sentiment Analysis on Twitter Data Using Apache Spark Framework,” in Proceedings - 2018 13th International Conference on Computer Engineering and Systems, ICCES 2018, Institute of Electrical and Electronics Engineers Inc., Feb. 2019, pp. 171–176. doi: 10.1109/ICCES.2018.8639195.

[2] B. Pang and L. Lee, “Opinion mining and sentiment analysis,” 2008.

[3] T. Carpenter, and T. Way, “Tracking Sentiment Analysis through Twitter,”. ACM computer survey. Villanova:VillanovaUniversity, 2010.

[4] H. Saif, Y.He, and H. Alani, “Semantic Sentiment Analysis of Twitter,” Proceedings of the Workshop on Information Extraction and Entity Analytics on Social Media Data. United Kingdom: Knowledge Media Institute, 2011.

[5] K. Ravi, V. Ravi and B. Shivakrishna, "Sentiment Classification Using Paragraph Vector and Cognitive Big Data Semantics on Apache Spark," 2018 IEEE 17th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC), Berkeley, CA, USA, 2018, pp. 187-194, doi: 10.1109/ICCI-CC.2018.8482085.

[6] B. Yan, Z. Yang, Y. Ren, X. Tan and E. Liu, "Microblog Sentiment Classification Using Parallel SVM in Apache Spark," 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 2017, pp. 282-288, doi: 10.1109/BigDataCongress.2017.43.

[7] A.Sharma, and S. Dey, “Performance Investigation of Feature SelectionMethods and Sentiment Lexicons for Sentiment Analysis,” Association for the Advancement of Artificial Intelligence, 2012.

[8] P. Goncalves, F. Benevenuto, M. Araujo and M. Cha, “Comparing and Combining Sentiment Analysis Methods”, 2013.

[9] A. P. Jain and P. Dandannavar, "Application of machine learning techniques to sentiment analysis," 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), Bangalore, India, 2016, pp. 628-632, doi: 10.1109/ICATCCT.2016.7912076.

[10] J. de Godoi Brandão and W. P. Calixto, "N-Gram and TF-IDF for Feature Extraction on Opinion Mining of Tweets with SVM Classifier," 2019

International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, 2019, pp. 1-5, doi: 10.1109/IDAP.2019.8875900.

[11] A. Go, R. Bhayani, and L. Huang, “Twitter Sentiment Classification using Distant Supervision.” [Online]. Available: http://tinyurl.com/cvvg9a

Aryan_Proj_May2023-2

ORIGINALITY REPORT

15%	11%	10%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.infoworld.com Internet Source	4%
2	www.hindawi.com Internet Source	2%
3	www.mdpi.com Internet Source	1%
4	www.reddit.com Internet Source	1%
5	Bo Yan, Zijiang Yang, Yitian Ren, Xing Tan, Eric Liu. "Microblog Sentiment Classification Using Parallel SVM in Apache Spark", 2017 IEEE International Congress on Big Data (BigData Congress), 2017 Publication	1%
6	Hossam Elzayady, Khaled M. Badran, Gouda I. Salama. "Sentiment Analysis on Twitter Data using Apache Spark Framework", 2018 13th International Conference on Computer Engineering and Systems (ICCES), 2018 Publication	1%