GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY COLLEGE

(AUTONOMOUS)

AFFILIATED TO BENGALURU CITY UNIVERSITY, APPROVED BY AICTE, DELHI & RECOGNISED BY THE GOVT. OF KARNATAKA

RE-ACCREDITED BY NAAC WITH 'A+' GRADE

A Project Report on

"TWITTER(X) SENTIMENT ANALYSIS USING NEURAL NETWORK"

Submitted in Partial Fulfillment of the Requirements For the award of the degree

Master of Computer Applications

SUBMITTED BY

SHIVAPRAKASH D M **22P01190**

PRESIDENCY COLLEGE

Kempapura, Hebbal, Bengaluru – 24

Re-accredited by NAAC with 'A+' Grade

DEPARTMENT OF COMPUTER APPLICATIONS

PRESIDENCY COLLEGE

(AUTONOMOUS)

AFFILIATED TO BENGALURU CITY UNIVERSITY, APPROVED BY AICTE, DELHI & RECOGNISED BY THE GOVT. OF KARNATAKA

RE-ACCREDITED BY NAAC WITH 'A+' GRADE

CERTIFICATE:

This is to certify that **SHIVAPRAKAS.D.M** with Register No. **22P01190** has satisfactorily completed the fourth semester MCA Project titled **"TWITTER(X) SENTIMENT ANALYSIS USING NEURAL NETWORKS",** as a partial fulfillment of the requirements for the award of the Degree in Master of Computer Applications, awarded by Bengaluru City University, during the Academic Year **2023-2024.**

Project Guide :

Head of Department
(Department of computer Application)

Examiners

1. _____

2. _____

REG.NO:_____

Examination Center:_____

Date of the exam:_____

\

Declaration:

The project titled "**TWITTER(X) SENTIMENT ANALYSIS USING NEURAL NETWORK**" developed by me in the partial fulfillment for the award of Master of Computer Application. It is a systematic work carried by us under the guidance of **Ms. UMA MAGESWARI,** Assistant professor, Department of Computer Applications.

I, declare that this same project has not been submitted to any degree or diploma to the Bengaluru City University or any other Universities.

Name of the student: -

Date:-

Signature

Acknowledgement:

The development of software is generally bit complex and time consuming task. The goal of developing the project "**TWITTER(X) SENTIMENT ANALYSIS USING NEURAL NETWORK**" could not be archived without the encouragements of kindly helpful and supportive people. Here by we convey our sincere thanks for all of them.

I take this opportunity to express my gratitude to people who had been instrumental in the successful completion of this project.

I am thankful to our management trustee for providing us an opportunity to work and complete the project successfully.

I wish to express my thanks to our **Principal Dr. Suchithra R Nair** for her support to the project work. I would like to acknowledge my gratitude to our HOD of Master of Computer Applications. **Dr.Alli** for her encouragement and support. Without their encouragement and guidance this project would not have materialized.

The guidance and support received from our Internal Guide **Ms. Uma Mageswari,** who contributed to this project, was vital for the success of the project. We are grateful for his constant support and help.

INDEX:

Introduction

1. INTRODUCTION AND OBJECTIVE

1.1 INTRODUCTION:

In the current virtual landscape, social media systems like Twitter (now referred to as X) serve as vital channels for public expression and communication. With tens of millions of users posting critiques, feelings, and reactions every day, the evaluation of this sizable pool of unstructured textual records gives significant opportunities and demanding situations. Sentiment analysis, a subfield of herbal language processing (NLP), ambitions to pick out and categorize the emotional tone at the back of these words, providing precious insights into public opinion on diverse subjects, starting from politics to product opinions. Leveraging superior device studying strategies, especially neural networks, can decorate the accuracy and performance of sentiment category, making it a important place of research and application.

This challenge focuses on using neural networks, mainly Convolutional Neural Networks (CNNs), to perform sentiment analysis on tweets amassed from Twitter. By using powerful libraries consisting of TensorFlow and Keras, we goal to build a model capable of information the context and nuances of human language, that's often laden with sarcasm, idioms, and varying linguistic patterns. The integration of the Natural Language Toolkit (NLTK) facilitates pre-processing responsibilities including tokenization, stemming, and removing stop words, thereby refining the dataset for extra accurate predictions. Through this approach, we are hoping to liberate deeper insights into the feelings expressed in tweets and the way they relate to actual-world activities.

Overall, this task seeks to increase a strong framework for sentiment analysis using today's techniques in deep studying and NLP. By combining the strength of neural networks with effective statistics processing and visualization gear, we goal to create a comprehensive answer that not only predicts sentiment appropriately but additionally affords actionable insights for corporations, policymakers, and researchers interested in understanding the heartbeat of social media conversations. Through this undertaking, we are hoping to contribute to the continuing discourse on sentiment analysis and its programs in numerous domains, showcasing the transformative potential of device gaining knowledge of in navigating the complexities of human expression on line.

1.2 OBJECTIVES:

The primary goal of this task is to develop an powerful sentiment analysis model that leverages Convolutional Neural Networks (CNNs) to as it should be classify the feelings expressed in tweets. By using Tensor Flow and Keras, we intention to construct a model that not best distinguishes between fine, poor,

aneutral sentiments however additionally captures the nuanced emotions conveyed within the textual content. To achieve this, we can recognition on several key aspects, which include the collection of a various dataset, meticulous data preprocessing using NLTK, and the optimization of the CNN architecture to decorate classification accuracy.

Additionally, we aim to provide visualizations that help interpret the sentiment evaluation outcomes. By employing Python libraries including Matplotlib and Seaborn, we can present the distribution of sentiments, traits through the years, and correlations with external activities. This will facilitate a better understanding of public sentiment and allow stakeholders to make knowledgeable decisions based on the insights collected from Twitter dataset. Ultimately, the task seeks to illustrate the effectiveness of deep getting to know techniques in extracting meaningful data from social media content, contributing to the wider discipline of sentiment evaluation and its packages.

By implementing diverse fashions and evaluating their overall performance based totally on metrics like accuracy, precision, recollect, and F1-score, we purpose to determine the blessings and limitations of the usage of deep studying methods as opposed to classical strategies in sentiment evaluation. This comparative analysis will provide treasured insights into the efficiency and effectiveness of CNNs in coping with complicated language styles and could manual future research in selecting appropriate models for comparable obligations. Ultimately, this objective not only enhances our understanding of the methodologies used in sentiment analysis but also contributes to the broader field of natural language processing by showcasing the potential of neural networks in extracting sentiment from unstructured data.

SOFTWARE REQUIREMENT ANALYSIS

2. SOFTWARE REQUIREMENT ANALYSIS

Software Requirement Analysis (SRA) is a critical phase in any undertaking development lifecycle, wherein the unique wishes, functionalities, and constraints of the software are identified and documented. The primary intention of SRA is to make sure that the software program machine being developed meets the meant motive and aligns with the consumer's expectations. In the context of the Twitter (X) sentiment analysis challenge, SRA makes a speciality of expertise the technical and purposeful requirements vital to develop an effective sentiment class model the usage of neural networks and system learning strategies.

At this degree, the evaluation consists of defining the software surroundings, equipment, and libraries required for the mission. This challenge leverages Python because the programming language, at the side of crucial libraries like TensorFlow and Keras for constructing and education the Convolutional Neural Network (CNN) version. Additionally, equipment like Pandas and NumPy are wished for green data manipulation, whilst NLTK is vital for preprocessing the tweet information. Visualization gear inclusive of Matplotlib and Seaborn play a key position in offering insights from the facts.

2.1 LITERATURE SURVEY:

The gift gadget for sentiment analysis on platforms like Twitter (X) generally relies on traditional device gaining knowledge of algorithms consisting of Support Vector Machines (SVM), Naive Bayes, or Logistic Regression. These strategies regularly require massive characteristic engineering, where textual content facts is converted into established enter the use of strategies like Bag of Words (BoW) or Term Frequency-Inverse Document Frequency (TF-IDF). While powerful to a point, those models war to capture the deeper, contextual meaning of language, such as sarcasm, idiomatic expressions, and long-variety dependencies in text.

Moreover, these structures frequently lack the ability to process big volumes of unstructured statistics efficiently, main to limitations in scalability and accuracy. The absence of superior deep getting to know techniques like Convolutional Neural Networks (CNNs) means that the current systems won't absolutely take advantage of the capacity of neural networks to enhance sentiment classification. Additionally, visualization tools in these systems can be constrained, offering best simple insights with out deeper evaluation of trends or patterns in the data. As a result, there is a want for more sophisticated models and equipment to reap better accuracy and higher insights from sentiment evaluation.

2.2 PROPOSED SYSTEM:

The proposed gadget is designed to perform sentiment analysis on tweets the usage of a Convolutional Neural Network (CNN) version, more advantageous via effective system mastering and herbal language processing tools. The system first gathers tweet information, preprocesses it by means of cleansing and tokenizing the textual content the usage of NLTK, after which feeds it into the CNN model built with TensorFlow and Keras. The version is educated to categorise tweets into high-quality, terrible, or impartial sentiment categories. The CNN is chosen for its potential to seize spatial hierarchies within the facts, making it well-suitable for text information classification.

In addition to the middle sentiment category, the system will visualize the effects the use of Matplotlib and Seaborn to offer insights which include sentiment distribution, developments through the years, and correlations with occasions. Furthermore, traditional machine gaining knowledge of models, implemented thru Scikit-analyze, can be used for comparison to assess the effectiveness of the CNN. The whole machine will be consumer-friendly, allowing actual-time evaluation and presenting actionable insights based totally on Twitter information.

This enhanced approach will offer more precise and context-aware sentiment classification, enabling real-time, scalable analysis of public opinion with improved accuracy and insights. Through this state-of-the-art methodology, the proposed system seeks to address the limitations of current techniques and provide a more effective solution for understanding and responding to trends on Twitter.

SYSTEM SPECIFICATIONS

3. SYSTEM SPECIFICATIONS

The machine specification for the Twitter sentiment analysis task outlines the core functionality, additives, and operational drift required to satisfy the project's objectives. It defines how the device will method information, educate models, and deliver sentiment classification results, making sure accuracy and performance. The system specification outlines the hardware and software requirements, in addition to the key additives essential for the a hit implementation of the Twitter sentiment evaluation task.

3.1 HARDWARE REQUIREMENTS:

Given the computational intensity of deep gaining knowledge of models, the system calls for sturdy hardware to system massive datasets and teach Convolutional Neural Networks (CNNs). A device with at least 8GB of RAM, ideally 16GB or more, is usually recommended to deal with the tweet records effectively at some stage in schooling and testing. Additionally, a multi-center CPU or, ideally, a GPU is necessary for faster computation, mainly whilst working with TensorFlow and Keras, that could leverage GPU acceleration for good sized performance upgrades. Adequate disk space, around 100GB, is likewise required to save the dataset and version checkpoints.

In addition to RAM and processing energy, a quick Solid State Drive (SSD) is particularly endorsed for advanced study/write speeds for the duration of records processing, version saving, and loading. This will beautify the general system performance, specifically while dealing with massive datasets and training models with more than one epochs. A strong internet connection is likewise vital, particularly at some point of the statistics series segment, as the device will fetch tweets in real-time the use of APIs. Moreover, if going for walks experiments on larger datasets or acting more complicated responsibilities (e.G., hyperparameter tuning), cloud-primarily based offerings like Google Colab, AWS, or Azure may be applied for get admission to to excessive-performance GPUs and TPUs, bearing in mind quicker model education and scalability.

3.2 SOFTWARE REQUIREMENTS:

The machine is constructed the use of Python, that's supported by way of a huge variety of libraries crucial for this project. TensorFlow and Keras are used to increase and train the CNN model, at the same time as NLTK is crucial for preprocessing the textual information. For facts dealing with and manipulation, Pandas and NumPy are required to correctly control tweet datasets. Additionally, visualization libraries like Matplotlib and Seaborn are used for graphical illustration of the analysis, even as Scikit-research is hired for implementing conventional machine mastering algorithms and model assessment. The Flask framework will

also be needed for backend implementation if the system is to be deployed as an internet utility. Python version three.Eight or better is usually recommended for compatibility with the state-of-the-art libraries and applications.

The software stack also includes Jupyter Notebooks or other Integrated Development Environments (IDEs) consisting of PyCharm or VS Code for code development and testing. Version manage structures like Git are critical for tracking adjustments in code and ensuring collaborative improvement whilst running in groups.

3.3 REQUIREMENTS:

Functional Specifications: The proposed machine will perform statistics series, preprocessing, model education, evaluation, and visualization. First, it retrieves tweets thru APIs and preprocesses the statistics using techniques like tokenization, stopword removal, and lemmatization with NLTK. Next, the CNN version, constructed the use of TensorFlow and Keras, is educated to classify sentiments into advantageous, poor, or impartial classes. The machine will also assist a evaluation of CNN's performance with conventional machine gaining knowledge of fashions using Scikit-study. Once the version is educated, the outcomes are visualized with Matplotlib and Seaborn to offer insights into sentiment tendencies. The machine is modular and designed to permit flexibility in managing unique datasets and easily adjusting version parameters.

H/W and S/W CONFIGURATIONS

4. H/W and S/W CONFIGURATIONS

HARDWARE	
Processor	Intel(R) Core(TM) i5-8365U CPU @ 1.60GHz 1.90 GHz
RAM	8GB
HDD	512GB
SOFTWARE	
CLIENT SIDE TECHNOLOGIES	HTML, CSS, JAVASCRIPT
SERVER SIDE TECHNOLOGIES	PYTHON
DATABASE	-
WEB SERVER	RSET API
IDE	Visual Studio Code
TESTING & BUILD TOOLS	-

SOFTWARE PROFILE

5. SOFTWARE PROFILE

The software program profile for the Twitter sentiment evaluation undertaking outlines the key components, equipment, and libraries that will be applied throughout the development process. This profile serves as a complete manual to the software program architecture, offering an overview of the functionalities and integrations required for the assignment.

5.1 PYTHON:

The number one programming language for this mission is Python, acknowledged for its simplicity and sturdy libraries for records technology and machine getting to know. Python's versatility makes it a great desire for each data manipulation and version development.

Python is a excessive-level, interpreted programming language recognized for its simplicity and flexibility. It was created within the overdue Eighties via Guido van Rossum and has considering come to be one of the maximum popular programming languages global. Python's syntax is designed to be easy and readable, which makes it an exceptional desire for each beginners and experienced programmers. This accessibility encourages speedy utility improvement and permits developers to focus on solving problems instead of getting bogged down with the aid of complex syntax regulations. As a end result, Python has determined programs in diverse domains, inclusive of net improvement, information technology, system gaining knowledge of, automation, and medical computing.

One of Python's finest strengths lies in its extensive atmosphere of libraries and frameworks. Libraries including NumPy and Pandas offer effective equipment for data manipulation and analysis, at the same time as Matplotlib and Seaborn are widely used for data visualization. In the realm of gadget gaining knowledge of, frameworks like TensorFlow and Keras allow developers to build and install sophisticated fashions with relative ease. The Python Package Index (PyPI) hosts hundreds of 0.33-birthday celebration programs, making it easy to find gear which could assist make bigger Python's abilties for particular obligations. This wealthy surroundings contributes notably to Python's reputation in data-centric fields, wherein rapid prototyping and green managing of large datasets are critical.

Moreover, Python's network help is strong and active, with sever assets available for learning and troubleshooting. This network-pushed technique fosters collaboration and innovation, leading to the continuous improvement of the language and its libraries. Many instructional establishments and groups provide Python publications, workshops, and resources, making sure that new customers can quickly get up

to hurry. Additionally, Python's go-platform compatibility allows it to run on various running structures, which includes Windows, macOS, and Linux, making it a flexible desire for developers throughout distinctive environments. Overall, Python's ease of use, sizeable libraries, and supportive network make it a effective device for a huge range of programming wishes.

5.2 PANDAS:

Pandas is a effective open-supply records manipulation and evaluation library for Python, designed to provide smooth-to-use information systems and records evaluation tools. In the context of Twitter sentiment analysis, Pandas performs a important position in dealing with the vast quantities of tweet records that want to be processed and analyzed. The library introduces two number one records systems—Series and DataFrame—permitting users to effectively prepare and manage records in a tabular layout. This structure is specifically useful for working with datasets that incorporate numerous attributes, together with tweet textual content, person records, and sentiment labels.

Data Preprocessing:

Before acting sentiment analysis, it's crucial to clean and preprocess the tweet information, and Pandas simplifies this assignment substantially. The library offers capabilities for filtering, sorting, and aggregating data, which assist in disposing of irrelevant statistics, including retweets or non-English tweets. Pandas also enables the coping with of lacking or inconsistent statistics, enabling analysts to fill in gaps or drop incomplete entries as wished. Additionally, with the potential to apply custom capabilities across columns and rows, customers can without problems rework the tweet text, consisting of converting it to lowercase or disposing of unique characters, making sure that the records is prepared for similarly processing. Pandas can classifies the column and rows. Pandas offers data structure and operations for powerful, flexible, and easy-to-use data analysis and manipulation

Data Analysis and Visualization:

Once the information is preprocessed, Pandas permits for exploratory facts evaluation (EDA) to benefit insights into the sentiment distribution a few of the tweets. By leveraging its effective group-by means of functionality, analysts can calculate sentiment proportions, perceive trends over the years, and have a look at relationships among one of a kind variables, which includes user engagement and sentiment. Furthermore, Pandas integrates seamlessly with visualization libraries like Matplotlib and Seaborn, enabling users to create informative charts and graphs that depict sentiment trends, enabling stakeholders to visualize

the sentiment landscape effectively. Overall, Pandas serves as an essential tool in the sentiment analysis workflow, enhancing data manipulation and analysis capabilities, ultimately leading to more accurate and meaningful insights from Twitter data.

5.3 NUMPY:

NumPy, short for Numerical Python, is a essential library for numerical computing in Python. It provides assist for massive, multi-dimensional arrays and matrices, along with a group of mathematical features to carry out operations on those information systems effectively. NumPy's effective n-dimensional array item, known as ndarray, is valuable to its functionality, permitting users to address big datasets without difficulty. This capability makes NumPy an crucial device in facts technological know-how, device gaining knowledge of, and clinical computing, where performance and performance are important.

Array Manipulation and Mathematical Operations:

In the context of Twitter sentiment analysis, NumPy enables green manipulation of numerical records, that's essential for making ready and processing functions extracted from tweets. For example, when acting operations like calculating averages, variances, or other statistical measures, NumPy allows these calculations to be executed on complete arrays with out the want for express loops, ensuing in faster execution times. This performance is mainly useful while handling massive datasets, because it reduces computational overhead and complements overall performance. Moreover, NumPy supports broadcasting, permitting operations on arrays of different shapes, which simplifies many common information manipulation responsibilities.

Integration with Other Libraries:

NumPy serves as the muse for lots other libraries in the Python ecosystem, along with Pandas, TensorFlow, and Keras. In the sentiment analysis undertaking, NumPy can be used alongside Pandas for numerical operations on DataFrames, making it easier to handle functions extracted from tweet textual content, which includes term frequency or sentiment rankings. Additionally, whilst schooling machine getting to know fashions, NumPy arrays can be easily fed into these libraries, facilitating smooth data

transfer between different components of the analysis pipeline. Overall, NumPy's efficient handling of numerical data and its integration with other libraries make it an indispensable tool for enhancing the performance and capabilities of the Twitter sentiment analysis project.

5.4 Matplotlib:

Matplotlib is a extensively used plotting library for Python that offers a complete interface for developing static, animated, and interactive visualizations. Designed to be quite customizable, Matplotlib permits customers to generate plenty of plots, inclusive of line graphs, bar charts, histograms, scatter plots, and greater. Its bendy API makes it suitable for both easy and complex visualizations, catering to a vast range of information visualization needs in clinical computing, facts evaluation, and gadget getting to know. Given its ease of use and big features, Matplotlib has emerge as the de facto widespread for information visualization within the Python surroundings.

Visualization in Twitter(X) Sentiment Analysis:

In the context of Twitter sentiment evaluation, Matplotlib performs a critical function in visualizing the insights derived from the sentiment category of tweets. After processing and reading the tweet facts, researchers can make use of Matplotlib to create clean and informative visual representations of sentiment distributions, tendencies over the years, and relationships between various factors, along with tweet extent and sentiment rankings. For example, line charts can illustrate how sentiment fluctuates for the duration of particular events or over positive intervals, at the same time as pie charts can depict the overall sentiment distribution (fantastic, negative, and neutral) inside a dataset. Such visualizations not most effective decorate the interpretability of the outcomes but also facilitate communication with stakeholders via presenting complicated information in an effortlessly digestible layout.

Customization and Integration:

One of Matplotlib's standout features is its sizeable customization competencies, permitting users to modify plot aesthetics, which includes colors, fonts, markers, and axes, to match their specific needs and preferences. This flexibility is specifically useful in professional settings wherein clean verbal exchange of information insights is important. Additionally, Matplotlib can be easily incorporated with different libraries, inclusive of Pandas and Seaborn, to beautify visualization abilities in addition. For instance, customers can create plots without delay from Pandas DataFrames or integrate Matplotlib's powerful plotting features with Seaborn’s advanced statistical visualizations. This integration allows analysts to produce excellent visualizations that convey meaningful information successfully, making Matplotlib an integral tool in the Twitter sentiment analysis task and different information-driven endeavors.

5.5 SEABORN:

Seaborn is a powerful Python information visualization library constructed on top of Matplotlib that gives a excessive-level interface for developing visually appealing and informative statistical portraits. Seaborn simplifies the process of generating complicated plots through supplying a wide variety of built-in subject matters and colour palettes, making it easy to create aesthetically eye-catching and expert-searching visualizations with minimal effort. While it inherits Matplotlib's flexibility, Seaborn is designed to deal with greater complicated datasets and robotically applies statistical ameliorations to supply insightful visualizations, making it best for exploratory facts analysis (EDA).

Usage in Twitter Sentiment Analysis :

In a Twitter sentiment evaluation task, Seaborn can be used to beautify the visualization of sentiment distributions, tendencies, and correlations. For instance, it could create certain visualizations of ways sentiment rankings are dispensed throughout one-of-a-kind time durations or consumer demographics, using plots including histograms, box plots, and violin plots. Seaborn also simplifies the introduction of pair plots or heatmaps to show relationships among more than one variables, including sentiment, retweets, and likes. Its statistical competencies allow for more nuanced evaluation, together with plotting regression traces or self belief periods, giving researchers deeper insights into the factors influencing sentiment on Twitter.

Seamless Integration and Customization :

Seaborn integrates seamlessly with Pandas, permitting customers to easily plot data at once from DataFrames with out extra preprocessing. It additionally works nicely with Matplotlib, permitting users to mix Seaborn's excessive-degree capabilities with Matplotlib's substantial customization alternatives. With Seaborn, customers can customize plots via adjusting plot aesthetics, color palettes, and gridlines, which facilitates make visualizations greater understandable and tailor-made to precise audiences. This aggregate of ease of use, advanced statistical plotting, and customization makes Seaborn a useful tool for producing fantastic visualizations in sentiment evaluation and different facts science tasks.

5.6 NLTK(NATURAL LANGUAGE TOOLKIT):

The Natural Language Toolkit (NLTK) is a complete library for natural language processing (NLP) in Python, broadly used for diverse textual content processing duties. Developed to guide linguistic studies and packages, NLTK presents clean-to-use interfaces to over 50 corpora and lexical sources, together with a

collection of text processing libraries for category, tokenization, stemming, tagging, parsing, and semantic reasoning. Its modular layout enables users to experiment with unique NLP techniques and methodologies, making it a famous choice amongst researchers and developers operating within the area of linguistics and artificial intelligence.

Applications in Twitter Sentiment Analysis:

In the context of Twitter sentiment evaluation, NLTK performs a critical function in preprocessing and analyzing the tweet information. Given the unstructured nature of tweets, NLTK provides numerous gear for cleaning and preparing the text for analysis. Key capabilities encompass tokenization, which splits text into person words or terms; stopword removal, which gets rid of commonplace phrases that add little which means (like "and," "the," and so on.); and stemming or lemmatization, which reduces phrases to their base forms (e.G., "walking" to "run"). These preprocessing steps are crucial for transforming raw tweet information right into a established layout that can be efficiently analyzed for sentiment.

Advanced Features and Customization:

Beyond basic preprocessing, NLTK gives advanced skills for sentiment evaluation and linguistic modeling. It includes various gadget learning algorithms that can be educated to classify sentiments primarily based on classified datasets, enabling customers to construct custom sentiment classifiers tailor-made to unique desires. NLTK additionally supports part-of-speech tagging and named entity popularity, which can provide additional context to the analysis by identifying the roles of words within the tweets. The library's flexibility and comprehensive documentation allow users to easily integrate it into their workflows, making it an essential tool for conducting thorough sentiment analysis and gaining deeper insights into textual data from platforms like Twitter.

5.7 TENSORFLOW:

TensorFlow is an open-supply machine studying framework developed by means of Google that offers a comprehensive ecosystem for building and deploying system learning and deep learning models. Launched in 2015, TensorFlow is designed to facilitate the development of complex computational graphs, allowing users to explicit mathematical operations as a chain of interconnected nodes and edges. This flexibility enables TensorFlow to deal with a whole lot of tasks, from simple linear regression to complex neural community architectures like convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

Key Features and Components:

One of the key capabilities of TensorFlow is its capacity to successfully execute computations across extraordinary platforms, along with CPUs, GPUs, and TPUs (Tensor Processing Units). This versatility makes it suitable for both studies and manufacturing environments. TensorFlow affords high-degree APIs, which include Keras, which streamline the process of building and schooling neural networks, making it available to customers with varying tiers of expertise. The framework also consists of TensorBoard, a effective visualization tool that permits developers to display model education, visualize performance metrics, and analyze model architecture in real-time. Tensor Flow Core API This low-diplo API provides precise flexibility over version settings and calculations, making it suitable for advanced designers or events that require flexibility

Applications in Machine Learning:

TensorFlow is extensively utilized in numerous packages, such as herbal language processing, laptop vision, and reinforcement getting to know. Its great library of pre-educated models and aid for transfer mastering allows practitioners to quickly adapt existing fashions to new obligations, considerably decreasing development time. Additionally, Tensor Flow's sturdy community support and complete documentation make it a go-to choice for researchers and developers working on modern-day device mastering tasks. Overall, Tensor Flow's flexibility, scalability, and wealthy surroundings make it an important device for everyone looking to implement gadget getting to know and deep studying solutions.

5.8 KERAS :

Keras is an open-supply excessive-stage neural networks API written in Python that runs on top of TensorFlow, Theano, and CNTK. Initially evolved by François Chollet in 2015, Keras become designed to simplify the procedure of building and training deep learning fashions, making it more on hand to both novices and experienced practitioners. Its person-friendly interface lets in developers to speedy prototype and iterate on deep learning architectures while not having to delve deeply into the underlying complexities of the decrease-degree libraries. This recognition on ease of use, modularity, and extensibility has contributed to Keras's rapid adoption inside the gadget gaining knowledge of network.

Core Features:

One of the center features of Keras is its intuitive and consistent API, which permits users to construct neural networks with the aid of stacking layers in a truthful manner. Users can easily define models the use of primary strategies: the Sequential API for linear stacks of layers and the Functional API for more complicated architectures regarding more than one inputs and outputs. Keras helps lots of layer sorts, together with

convolutional, recurrent, and fully connected layers, permitting users to construct diverse neural community architectures ideal to their specific duties. Additionally, Keras consists of built aid for common education utilities, which include loss functions, optimizers, and metrics, streamlining the version education procedure.

Integration with TensorFlow:

Keras is tightly included with TensorFlow, making it a great choice for constructing deep learning fashions in the TensorFlow ecosystem. This integration allows Keras users to leverage TensorFlow's superior capabilities, which include allotted education, version deployment, and TensorBoard for visualisation. Keras models can be effortlessly converted into TensorFlow's computational graphs, making an allowance for efficient execution throughout one-of-a-kind hardware configurations, together with GPUs and TPUs. This synergy enhances Keras's abilties, offering customers with get right of entry to to the overall strength of TensorFlow whilst maintaining the simplicity and ease of use that Keras is understood for.

Applications and Community Support:

Keras is extensively used in numerous domain names, including laptop vision, natural language processing, and time collection analysis, among others. Its versatility enables researchers and practitioners to use deep learning techniques to a large range of problems, from photograph classification and textual content technology to reinforcement gaining knowledge of. The Keras community is vibrant and lively, with enormous documentation, tutorials, and assets to be had to help customers in any respect stages. This network-pushed method fosters collaboration and understanding sharing, ensuring that Keras continues to conform and remain relevant in the rapidly changing landscape of deep learning. Overall, Keras serves as a powerful but person-friendly tool for anybody looking to enforce deep learning solutions efficiently.

5.9 SCI-KIT LEARN:

Scikit-study is an open-supply machine studying library for Python that provides a wide variety of equipment for information evaluation and modeling. Built on top of NumPy, SciPy, and Matplotlib, Scikit-

analyze is designed to be simple and efficient, allowing customers to enforce system gaining knowledge of algorithms and approaches with minimal code. Since its launch, it has end up one of the maximum famous libraries for machine gaining knowledge of, extensively utilized by information scientists and researchers for its complete suite of algorithms and utilities. Its person-friendly interface and steady API make it available for both novices and experienced practitioners, facilitating rapid experimentation and model development. Key

concepts and features include: Algorithmic decision-making methods, including: Classification: identifying and categorizing data based on patterns. It provides a comprehensive set of supervised and unsupervised learning algorithms, covering areas such as: Classification: Identifying which category an object belongs to. Regression: Predicting a continuous-valued attribute associated with an object.

Key Features:

One of the standout capabilities of Scikit-learn is its massive collection of supervised and unsupervised mastering algorithms, including regression, classification, clustering, and dimensionality discount strategies. The library includes famous algorithms such as linear regression, decision trees, help vector machines, and ok-method clustering, amongst others. Additionally, Scikit-learn gives utilities for model assessment and selection, allowing users to evaluate the overall performance in their models thru metrics like accuracy, precision, don't forget, and F1 score. This integrated functionality enables customers to streamline the model-constructing system, from training and validation to trying out and deployment.

Preprocessing and Pipelines:

Scikit-learn also offers robust data preprocessing talents which might be crucial for getting ready information for machine mastering. These capabilities encompass functions for dealing with missing values, scaling and normalizing information, encoding categorical variables, and characteristic extraction. The library's preprocessing module guarantees that users can efficiently transform their datasets into suitable formats for modeling. Furthermore, Scikit-analyze supports the advent of machine mastering pipelines, which allow customers to chain together preprocessing steps and version schooling right into a unmarried workflow. This promotes cleaner code and helps avoid facts leakage at some stage in version evaluation.

Community and Ecosystem:

The Scikit-study network is vibrant and supportive, with comprehensive documentation, tutorials, and assets available to help customers in mastering and imposing device studying strategies. The library is regularly up to date with new functions and improvements, reflecting the contemporary tendencies inside the subject. Its integration with different famous libraries in the Python atmosphere, such as Pandas for statistics manipulation and Matplotlib for visualization, makes it a versatile choice for cease-to-end information analysis and modeling duties. Overall, Scikit-analyze sticks out as an critical tool for all people interested by gadget getting to know, imparting effective capability even as keeping ease of use.

5.10 MACHINE LEARNING ALGORITHMS:

Machine Learning algorithms are the programs that can learn the hidden patterns from the data, predict the output, and improve the performance from experiences on their own. Different algorithms can be used in machine learning for different tasks, such as simple linear regression that can be used for prediction problems like stock market prediction, and the KNN algorithm can be used for classification problems.

Types of Machine Learning Algorithms:

Machine Learning Algorithm can be broadly classified into three types:

1. Supervised Learning Algorithms
2. Unsupervised Learning Algorithms
3. Reinforcement Learning algorithm

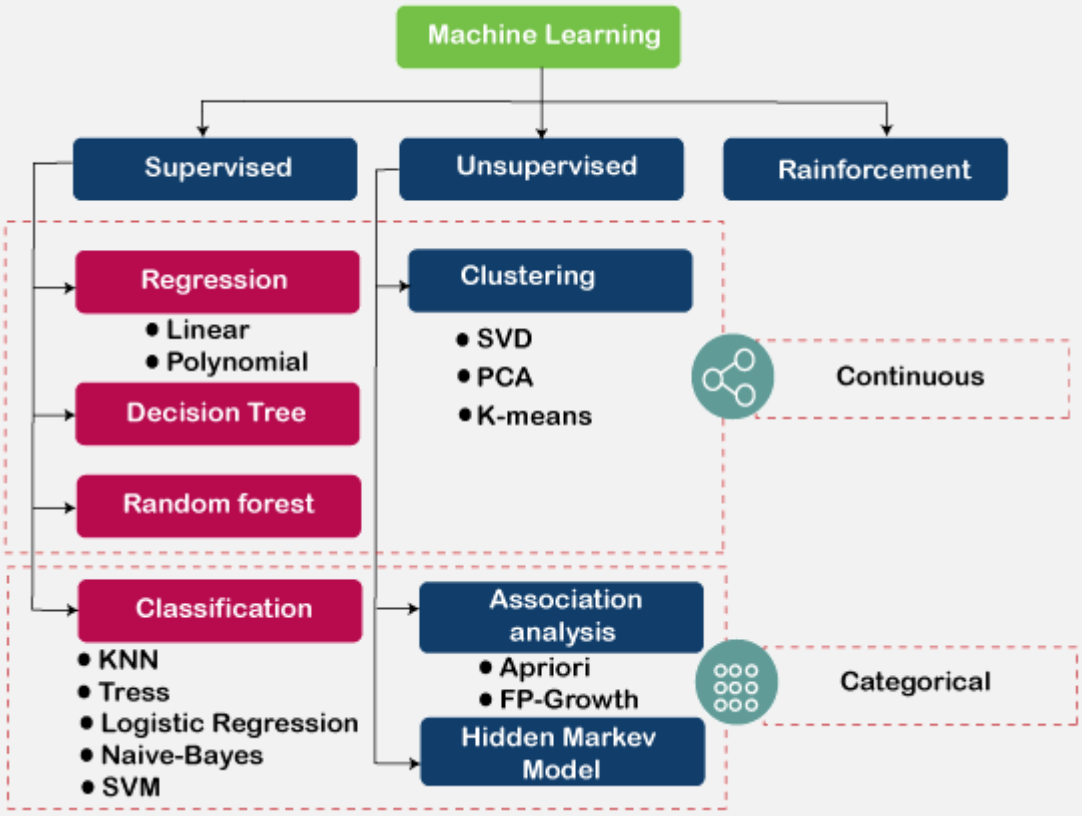


FIG 5.9: Different ML Algorithms

5.10.1 Supervised Learning Algorithms

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output. In supervised learning, the training data provided to the

machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.

Types of supervised Machine learning Algorithms:

Supervised learning can be further divided into two types of problems:

1**. Regression:**

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

2. **Classification:**

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Spam Filtering,
- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

5.10.2 Unsupervised Learning Algorithms

Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision. Unsupervised learning cannot be directly

applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.

Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:

1. **Clustering:**

   Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

2. **Association:**

   An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

5.10.3 Reinforcement Learning Algorithm

Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty. In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.nSince there is no labeled data, so the agent is bound to learn by its experience only. RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as game-playing, robotics, etc.

The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards. The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that "Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that." How a Robotic dog learns the movement of his arms is an example of Reinforcement learning. It is a core part of Artificial intelligence, and all AI agent works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention.

Machine Learning Application:

- **Image Recognition:** Machine learning is used in facial recognition, object detection, and medical image analysis, allowing systems to identify and classify images or features within them.

- **Natural Language Processing (NLP):** Applications like sentiment analysis, language translation, and chatbots rely on machine learning to understand and generate human language.
- **Fraud Detection:** Financial institutions use machine learning algorithms to detect fraudulent transactions by identifying unusual patterns and behaviors in financial data.
- **Recommendation Systems:** Platforms like Netflix, Amazon, and YouTube use machine learning to recommend content based on user behavior and preferences.
- **Autonomous Vehicles:** Machine learning enables self-driving cars to process sensor data, recognize objects, and make decisions in real time to navigate safely.
- **Healthcare Diagnostics:** Machine learning models assist in predicting diseases, analyzing medical images, and personalizing treatment plans, improving diagnosis and patient outcomes.
- **Predictive Maintenance:** In industries such as manufacturing and aviation, machine learning helps predict equipment failures before they occur, reducing downtime and maintenance costs.
- **Financial Market Prediction**: Machine learning models are used in stock market prediction and trading by analyzing historical data and market trends to make informed financial decisions.

5.10.4 List of Popular Machine Learning Algorithm

1. **Linear Regression Algorithm:**

Linear regression is one of the most popular and simple machine learning algorithms that is used for predictive analysis. Here, predictive analysis defines prediction of something, and linear regression makes predictions for continuous numbers such as salary, age, etc. It shows the linear relationship between the dependent and independent variables, and shows how the dependent variable(y) changes according to the independent variable (x). It tries to best fit a line between the dependent and independent variables, and this best fit line is knowns as the regression line.

The equation for the regression line is:

y= a0+ a*x+ b

Here,

y= dependent variable.

x= independent variable.

a0 = Intercept of line.

2. **Logistic Regression Algorithm:**

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or

False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc. Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

3. **Naïve Baye's**

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles. Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:

$$P(A|B)=(P(B|A).P(A))/ P(B)$$

Where,

=> P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

=> P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

=> P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

=> P(B) is Marginal Probability: Probability of Evidence.

4. **KNN(K-Nearest Neighbor):**

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready.

5.K-Means Clustering

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on. It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties. It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters. The

algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

5.11 NEURAL NETWORKS:

Neural networks are a class of machine getting to know fashions inspired via the structure and characteristic of the human brain. They are designed to apprehend styles, analyze from facts, and make decisions or predictions. A neural community includes interconnected layers of artificial neurons (or nodes) that process statistics and pass it from one layer to the next. These models excel at shooting complicated, non-linear relationships in facts, making them enormously effective for responsibilities inclusive of photo recognition, natural language processing, and time collection forecasting. Neural networks are the backbone of deep studying, which is a subset of gadget studying targeted on multi-layered (or deep) networks.

Architecture of Neural Networks:

The structure of a neural network is normally composed of three important types of layers: the enter layer, one or extra hidden layers, and the output layer.

Input Layer: This layer receives the raw records (e.G., pixel values for an image or phrases in a sentence). Each node in the enter layer corresponds to a characteristic within the dataset.

Hidden Layers: These are the layers among the input and output layers, where the network performs most of its studying. Each neuron within the hidden layers applies a weighted sum of inputs, observed by way of an activation function (inclusive of ReLU, sigmoid, or tanh) to introduce non-linearity. The more hidden layers a network has, the "deeper" it's far, allowing it to examine extra complicated styles in the information.

Output Layer: This layer produces the final predictions, consisting of a classification label or a non-stop price. For classification responsibilities, the output layer normally makes use of a softmax characteristic to supply chance ratings for extraordinary training.

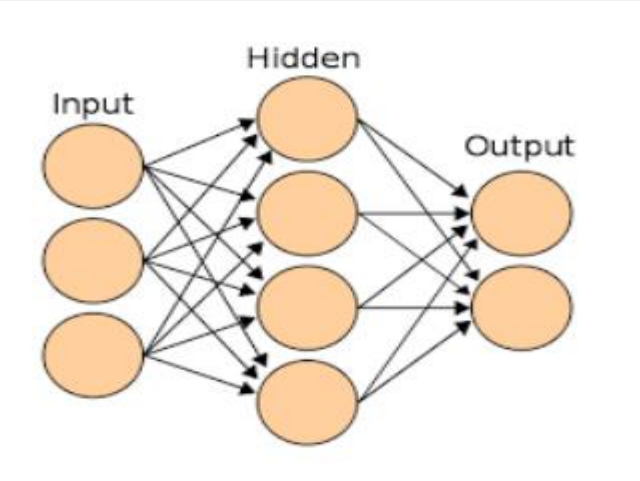


FIG 5.11: NEURAL NETWORK

Applications of Neural Networks:

Neural networks have revolutionized many fields due to their ability to model complex, high-dimensional data. Some key applications include:

- **Computer Vision:** Neural networks, particularly CNNs, are widely used in image and video analysis tasks, including facial recognition, object detection, and autonomous driving.
- **Natural Language Processing (NLP):** In NLP tasks such as sentiment analysis, translation, and chatbots, neural networks help machines understand and generate human language. RNNs and their variants (like LSTMs) are often used for these tasks.
- **Speech Recognition:** Neural networks enable machines to recognize and process spoken language, powering applications like virtual assistants (e.g., Siri, Alexa).
- **Healthcare:** Neural networks are used in diagnosing diseases from medical images, predicting patient outcomes, and analyzing genomic data for personalized medicine.

Types of Neural Networks:

There are various types of neural networks, each suited for different kinds of tasks:

1. Feed Forward Neural Networks (FNN)
2. Convolutional Neural Networks (CNNs)
3. Recurrent Neural Networks (RNNs)

5.11.1 Feed Forward Neural Networks (FNN):

The handiest kind, where statistics flows in a single route from the enter layer thru the hidden layers to the output layer. FNNs are commonly used for fundamental duties like type and regression. Feedforward Neural Networks (FNNs) are the maximum fundamental sort of synthetic neural networks, in which the

records flows in one path—from the enter layer, through one or greater hidden layers, and eventually to the output layer.

In an FNN, there are not any loops or cycles, which means every layer's output is used solely as input for the subsequent layer. The nodes in each layer perform mathematical operations (such as weighted sums and activations) on the incoming information to capture styles or relationships, permitting the community to study from the schooling records. This makes FNNs suitable for obligations which include class, regression, and simple sample popularity, where the statistics is static and does no longer require time-established processing.

Functioning and Applications:

During education, FNNs modify the weights of the connections between neurons the use of optimization algorithms like gradient descent and mistakes backpropagation. The community's intention is to minimize the error between the expected output and the actual target via iteratively adjusting weights primarily based at the schooling data. The activation functions applied in the hidden layers introduce non-linearity, permitting FNNs to learn complex patterns. Though relatively easy in comparison to different kinds of neural networks (together with recurrent or convolutional networks), FNNs are still extensively used for a number of programs, together with image classification, spam detection, and predictive analytics, in which relationships in records aren't dependent on sequences or spatial hierarchies.

5.11.2 Convolutional Neural Networks (CNNs):

Designed to manner sequential facts, RNNs have connections that shape loops, letting them hold records about preceding inputs. This makes them useful for obligations concerning time-series information, consisting of language modeling and speech reputation. Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) are superior forms of RNNs that assist cope with troubles like vanishing gradients during schooling.

Convolutional Neural Networks (CNNs) are a specialized kind of neural network designed for processing and analyzing grid-like data, together with pix or time-series records. The key function of CNNs

is using convolutional layers, which practice filters (or kernels) that slide throughout the enter information to seize local patterns, including edges, textures, or other visible functions. This process of convolution reduces the spatial dimensions of the information while maintaining vital information. CNNs normally have more than one convolutional layers stacked collectively, permitting the network to study increasingly more

abstract and complicated functions at each layer. Pooling layers, which includes max pooling, further reduce the size of the information whilst retaining key capabilities, enhancing computational efficiency.

Applications and Advantages:

CNNs are extensively used in laptop vision duties, including photograph classification, item detection, and facial popularity, because of their potential to mechanically examine and extract capabilities from pix. Unlike traditional machine gaining knowledge of models, which rely upon manual feature engineering, CNNs learn the functions at once from the information, making them fantastically effective for large-scale image processing. Beyond pc vision, CNNs are also carried out in fields which includes video evaluation, natural language processing (for duties like textual content type and sentence modeling), and healthcare (e.g., analyzing medical images). Their hierarchical feature extraction mechanism allows CNNs to capture both low-level details and high-level concepts, making them powerful tools for analyzing structured and unstructured visual data.

5.11.3 Recurrent Neural Networks (RNNs):

Designed to method sequential statistics, RNNs have connections that shape loops, allowing them to maintain statistics about preceding inputs. This makes them beneficial for tasks related to time-series facts, which includes language modeling and speech reputation. Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) are advanced kinds of RNNs that help address issues like vanishing gradients at some point of education.

Recurrent Neural Networks (RNNs) are a category of neural networks designed to deal with sequential information, such as time-collection, text, and speech. Unlike Feedforward Neural Networks (FNNs), RNNs have connections that allow data to be surpassed from one step within the collection to the subsequent, growing a "reminiscence" of previous inputs. This potential to preserve facts over time makes RNNs mainly ideal for responsibilities where the order of statistics topics, such as language modeling, device translation, and video processing. The key mechanism in RNNs is the looping shape of their structure, where the hidden nation is up to date at each time step and might have an effect on destiny states, allowing the community to research dependencies in sequences.

Applications and Challenges:

RNNs are extensively used in obligations like speech popularity, herbal language processing (NLP), and time-collection forecasting because of their functionality to version temporal or sequential relationships. However, wellknown RNNs face challenges just like the vanishing gradient hassle, which makes it hard to

examine long-time period dependencies in sequences. To triumph over this limitation, advanced RNN variants like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) have been developed. These models comprise specialised mechanisms (gates) to control the waft of information, letting them higher capture lengthy-term dependencies and perform well on tasks that require understanding context over extended sequences, such as text generation and sentiment analysis.

5.12 HTML:

HTML (HyperText Markup Language) is the standard language used to create and design webpages. It paperwork the structure of a web site by way of the usage of various factors or tags to arrange and display content including textual content, pix, hyperlinks, and multimedia. Developed within the early 1990s with the aid of Tim Berners-Lee, HTML has when you consider that evolved, with the contemporary version being HTML5, which introduced new functions and upgrades for modern-day net development. HTML provides the fundamental framework that builders use to create web sites, ensuring that browsers can interpret and render the content correctly.

Structure of HTML Documents:

HTML files are made up of elements enclosed in perspective brackets (< >) and include a gap tag, content material, and a final tag (e.G., Paragraph content). These elements define various components of the web site, which includes headings, paragraphs, links, and pictures. An HTML file starts offevolved with the <!DOCTYPE html> statement, accompanied via the <html> tag, which encapsulates the whole content material of the web site. Within the <html> tag, there are two predominant sections: the <head> and the <frame>. The <head> section incorporates metadata, consisting of the title of the page and hyperlinks to stylesheets, at the same time as the <frame> phase incorporates the seen content material that users have interaction with.

HTML Elements and Tags:

HTML factors are the building blocks of a web site. Some common tags encompass <h1> to <h6> for headings, for paragraphs, <a> for links, <img> for photographs, and for dividing sections of a page. Each tag serves a specific purpose, making it less difficult to format and structure content. For example, hyperlinks may be created the use of the <a> tag, with the href characteristic defining the URL to navigate HTML also lets in for embedding multimedia, together with videos (<video>), audio (<audio>), and graphics (<canvas>), giving developers flexibility in supplying content material.

HTML5 and New Features:

HTML5 is the state-of-the-art model of HTML, which added new factors and competencies, inclusive of semantic tags, progressed multimedia help, and APIs for improved internet functionality. Semantic tags like <header>, <footer>, <article>, and offer extra significant shape to the website, supporting both builders and search engines like google higher apprehend the content material. HTML5 additionally natively helps audio and video with out the want for 1/3-celebration plugins like Flash. The <canvas> detail enables developers to attract and animate photographs immediately on a website using JavaScript, expanding the possibilities for interactive net content.

Forms and User Input:

HTML includes a whole lot of enter factors that permit customers to engage with webpages via bureaucracy. Form tags together with <enter>, <textarea>, <choose>, and <button> allow users to post data to the server for processing. HTML5 introduced new enter sorts like date, email, range, and search, which enhance the person experience by means of imparting extra unique methods to capture facts. Form validation has also advanced in HTML5, allowing developers to put in force policies like required fields or enter layout (e.G., electronic mail format) immediately in HTML without relying totally on JavaScript.

Importance of HTML in Web Development

HTML is the inspiration of web development, because it offers the shape and layout for web sites. It works along with CSS (Cascading Style Sheets) for styling and JavaScript for interactivity, making it an critical issue of front-give up improvement. Without HTML, browsers might now not be able to display content material or provide navigation, making it important for constructing websites and net packages. Its simplicity, ease of use, and compatibility throughout all browsers make HTML a accepted language for web builders, from novices to professionals. As the web keeps to evolve, HTML stays a cornerstone of the net improvement atmosphere.

5.13 FLASK:

Flask is a lightweight and flexible internet framework written in Python. It is assessed as a "micro-framework" because it affords the essential equipment for building net programs however doesn’t include many integrated features like more complex frameworks (together with Django). Flask is designed to be easy, allowing developers to fast set up an internet server and create net programs. With its minimalistic approach, Flask gives builders manipulate over how they build their programs, providing flexibility for smaller projects, API development, or large applications when blended with extra extensions.

Simplicity and Flexibility:

One of the key reasons Flask is popular amongst developers is its simplicity. Flask follows a minimalistic philosophy by offering only the primary components had to build a web utility: routing (URL mapping), request managing, and templates for rendering HTML. Developers can add capability via importing extensions, together with Flask-SQLAlchemy for database integration or Flask-WTF for form dealing with. This "build-it-as-you-move" technique allows developers to scale the software by including only the important tools without being restrained by means of the framework's shape.

Routing and URL Mapping:

In Flask, routing is straightforward and defines how URLs map to features inside the code, which can be called "view capabilities." When a consumer visits a particular URL, Flask triggers the corresponding characteristic, and this feature approaches the request and returns a response (usually HTML, JSON, or different records). The @app.Course() decorator is used to define routes in Flask, making it clean to associate precise URLs with Python functions. This characteristic is beneficial for constructing RESTful APIs or serving unique pages of a internet site, together with a homepage, contact page, or a login page.

Templating and Jinja2:

Flask uses the Jinja2 templating engine to render dynamic HTML pages. With Jinja2, developers can embed Python logic (which includes loops and conditionals) directly in the HTML to create dynamic content material. For instance, if you want to display a listing of gadgets on a website, you may bypass that listing from the Python backend to the HTML template and render each object dynamically. This separation of good judgment and presentation makes net improvement extra prepared, as it lets in developers to keep their HTML templates clean even as writing enterprise good judgment in Python.

Extensibility and Modular Design:

Although Flask is minimum by means of design, it is rather extensible, meaning builders can combine third-birthday party libraries or use Flask’s very own extensions to feature capabilities along with database aid, consumer authentication, or file uploads. Flask doesn’t impose specific methods to prepare your code, making it suitable for both small applications and complex systems. For example, builders can shape their Flask initiatives into modular components using blueprints, which allows them to interrupt down their utility into more than one, plausible portions, making the improvement manner extra efficient for larger initiatives.

Flask in Modern Web Development:

Flask is broadly used in present day web development due to its flexibility and ease of use. It is normally used to build RESTful APIs, making it a remarkable desire for backend improvement in microservices architectures or for building API-pushed packages. Flask is also a famous choice for prototyping, because it lets in builders to fast spin up a server and take a look at new thoughts with minimal setup. Flask's adaptability and growing surroundings of extensions make it appropriate for tasks starting from simple web sites to complex agency-stage packages, solidifying its place in Python web development.

5.14 GOOGLE COLAB:

Google Colab (brief for Colaboratory) is a unfastened, cloud-based totally Jupyter notebook surroundings that lets in customers to put in writing and execute Python code immediately in their browser with none setup. It provides an handy platform for system mastering, information technological know-how, and preferred Python programming via offering built-in aid for libraries consisting of TensorFlow, PyTorch, NumPy, and Pandas. One of the key advantages of Google Colab is its ability to leverage effective hardware, which includes free access to GPUs and TPUs, making it best for schooling gadget getting to know models. Additionally, users can save their notebooks to Google Drive, collaborate with others in real-time, and percentage their work effortlessly. Google Colab's cloud-based totally infrastructure gets rid of the need for nearby installations, making it a popular tool for students, researchers, and builders.

In addition to its ease of use and powerful hardware get admission to, Google Colab offers several convenient functions for facts technology workflows. Users can import datasets directly from Google Drive, GitHub, or add documents from their nearby device, making it easy to work with numerous facts assets. Colab additionally supports rich textual content formatting, allowing for the inclusion of Markdown cells, which may be used to feature explanatory text, mathematical equations, or visualizations alongside code. This makes it a superb device for creating interactive reports or tutorials. Moreover, Colab supports

seamless integration with TensorBoard, facilitating actual-time tracking of gadget gaining knowledge of experiments. With these skills, Google Colab has come to be an vital tool for builders and researchers trying to test, prototype, and collaborate on records-pushed projects with out demanding approximately computational obstacles.

5.15 Twitter Dataset :

Introduction to Kaggle and Twitter Dataset:

Kaggle is a popular platform for data technology competitions, web hosting a huge kind of datasets that developers and researchers use for evaluation, device gaining knowledge of, and deep getting to know tasks. One of the commonly explored datasets on Kaggle is the Twitter Dataset, which typically carries textual content-based totally records from tweets that may be used for responsibilities consisting of sentiment analysis, topic modeling, or consumer conduct analysis. These datasets frequently consist of features like tweet content material, person statistics, timestamps, hashtags, and metadata consisting of likes, retweets, and replies. This makes Twitter information rich and flexible for exceptional kinds of natural language processing (NLP) obligations.

Structure of Twitter Dataset:

The structure of Twitter datasets can range relying at the reason of the information collection. Generally, those datasets consist of numerous columns. Key columns may encompass the textual content of the tweet, consumer ID, created_at (timestamp), hashtags, mentions, likes, and retweets. For sentiment evaluation, many Twitter datasets additionally include a pre-classified sentiment rating, indicating whether or not a tweet is high quality, terrible, or impartial. This structure allows for comprehensive textual content analysis in addition to exploring relationships between consumer conduct and engagement metrics like retweets and likes.

Common Use Cases for Twitter Data:

Twitter datasets are extensively used in sentiment analysis, which includes figuring out the emotional tone of a tweet. For example, groups use Twitter sentiment evaluation to gauge client reviews approximately their products or services. Similarly, inside the subject of social media analytics, researchers use Twitter facts to investigate traits, music the recognition of hashtags, or examine public reactions to political events. Moreover, Twitter datasets are regularly leveraged in gadget studying fashions to teach classifiers, discover junk mail money owed, or are expecting the virality of tweets primarily based on their capabilities and engagement metrics.

Preprocessing Twitter Data:

Before the use of Twitter statistics for analysis, preprocessing steps are commonly required. This often includes cleansing the text with the aid of removing URLs, mentions, hashtags, and special characters. Additionally, changing text to lowercase, tokenizing it (splitting it into individual words or phrases), and

removing prevent words (not unusual phrases like "the," "is," and many others.) are important to prepare the records for machine getting to know models. Libraries like NLTK (Natural Language Toolkit) and Pandas are generally utilized in Python to preprocess Twitter information successfully. These steps assist to normalize the records, making it greater appropriate for evaluation.

Machine Learning with Twitter Data:

Twitter datasets provide an tremendous base for schooling device getting to know fashions. For sentiment analysis, supervised studying algorithms which include Logistic Regression, Support Vector Machines (SVM), and neural networks like Convolutional Neural Networks (CNN) or Long Short-Term Memory networks (LSTMs) are generally applied. After preprocessing, features like phrase embeddings (e.G., Word2Vec or GloVe) can be used to convert the tweet textual content into numerical vectors, that can then be fed into the models. These models examine patterns in the textual content to expect sentiments or classify tweets based totally on predefined classes.

Challenges of Working with Twitter Data:

While Twitter datasets are a valuable aid, working with them provides sure challenges. Tweets regularly include slang, abbreviations, emojis, and casual language, which can complicate textual content analysis and model schooling. Additionally, the brevity of tweets (with a individual restriction) could make it hard to seize full context or sentiment. Another challenge is managing the full-size extent of statistics, mainly in real-time analysis. Depending at the use case, some projects require streaming records from Twitter APIs, which entails coping with fee limits and huge-scale statistics processing. Despite those challenges, Twitter datasets continue to be one of the most popular sources for exploring NLP and social media analytics.

SYSTEM DESIGN

6 SYSTEM DESIGN

System layout refers to the process of defining the architecture, additives, modules, interfaces, and data for a system to satisfy special requirements. It includes a excessive-degree plan that outlines how exceptional factors of the gadget will have interaction and feature collectively to acquire the desired targets. In the context of software program improvement, machine layout bridges the gap among conceptual models and concrete implementations. It makes a speciality of both the structural components of a device—which includes database schemas, APIs, and conversation protocols—and the useful components, making sure that the device meets consumer necessities and enterprise wishes.

Importance of System Design

Effective device layout guarantees scalability, maintainability, and overall performance of the gadget. It considers numerous elements consisting of the choice of architecture (monolithic or microservices), design patterns, protection, records waft, and consumer interactions. A properly-designed gadget can accommodate destiny adjustments, cope with increase, and carry out correctly below exceptional masses. Good machine layout also facilitates prevent common troubles which includes bottlenecks, redundant information processing, and protection vulnerabilities. By carefully planning the layout section, developers create a sturdy basis, making sure the software program gadget is reliable, flexible, and smooth to extend as necessities evolve.

System design typically involves several critical components, each playing a unique role in the overall architecture. These components include the data flow design, which outlines how data moves through the system and interacts with various components, and the database design, where the structure of data storage is defined (e.g., relational databases or NoSQL). User interface (UI) design ensures the system is user-friendly and accessible, while network design handles communication between different parts of the system, especially in distributed or cloud-based applications. Additionally, security design focuses on safeguarding data and ensuring system integrity through authentication, authorization, and encryption mechanisms. These components, when well-integrated, form a cohesive and reliable system capable of meeting both functional and non-functional requirements.

6.1 E-R DIAGRAM

An Entity-Relationship (ER) diagram for sentiment analysis represents the important thing entities and their relationships involved within the system. In a typical sentiment analysis venture, core entities include User, Tweet (or Text), Sentiment, and probably Hashtag or Keyword. The User entity might shop user statistics which includes person ID and username. The Tweet (or Text) entity could capture the tweet's content material, timestamp, and other metadata like tweet ID. The Sentiment entity might represent the result of the sentiment evaluation, which will be categorized as high-quality, poor, or neutral. Each tweet is associated with a user thru a "posted through" dating and is associated with a sentiment rating via a "has sentiment" relationship.

Relationships within the ER Diagram:

In the ER diagram for sentiment evaluation, relationships define how those entities interact. A User can publish many Tweets, establishing a one-to-many dating between the User and Tweet entities. Each Tweet could have one corresponding Sentiment (high quality, negative, or impartial), defining a one-to-one dating between Tweet and Sentiment. Additionally, if keywords or hashtags are tracked, they are able to have a many-to-many relationship with tweets, where a Tweet can incorporate a couple of Hashtags, and a Hashtag can appear in lots of special tweets. This ER diagram visually helps to map the glide and interaction of data in a sentiment evaluation system, ensuring clarity in the database shape.

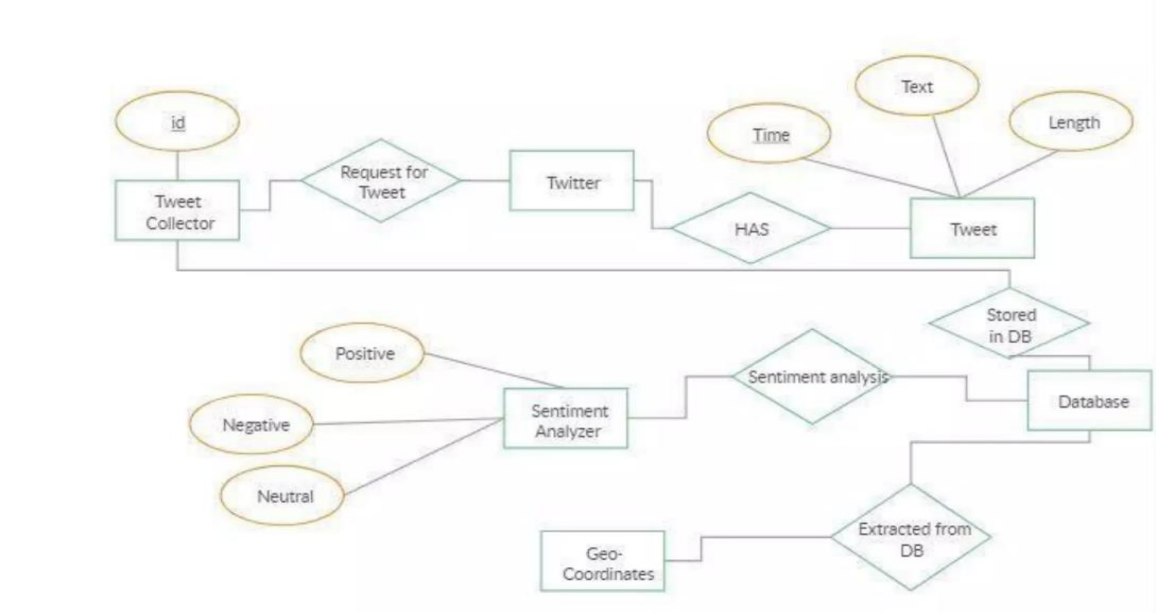


FIG 6.1: E-R Diagram For Sentiment Analysis.

6.2 DATA-FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical representation that depicts the go with the flow of information through a system, displaying how input information is processed into output consequences. It illustrates the machine's additives, which includes facts assets, facts processes, statistics shops, and information locations, and how they have interaction. In a sentiment evaluation assignment, as an example, the DFD may start with the input (along with uncooked tweets or textual content data from users or social media systems) flowing right into a records preprocessing step. In this step, responsibilities like text cleansing, tokenization, and filtering are finished to prepare the information for evaluation.

Key Elements of a DFD for Sentiment Analysis:

After preprocessing, the facts flows into the sentiment evaluation module, in which system studying or natural language processing algorithms classify the textual content as fine, poor, or neutral. The outcomes are then saved in a database (information shop), and the machine might also similarly procedure those outcomes for visualisation (e.G., displaying sentiment trends or charts on a dashboard). The DFD helps visualize the complete technique, making sure clear conversation approximately how records moves via the sentiment analysis system and wherein every operation is achieved, making it easier to become aware of inefficiencies or potential enhancements.

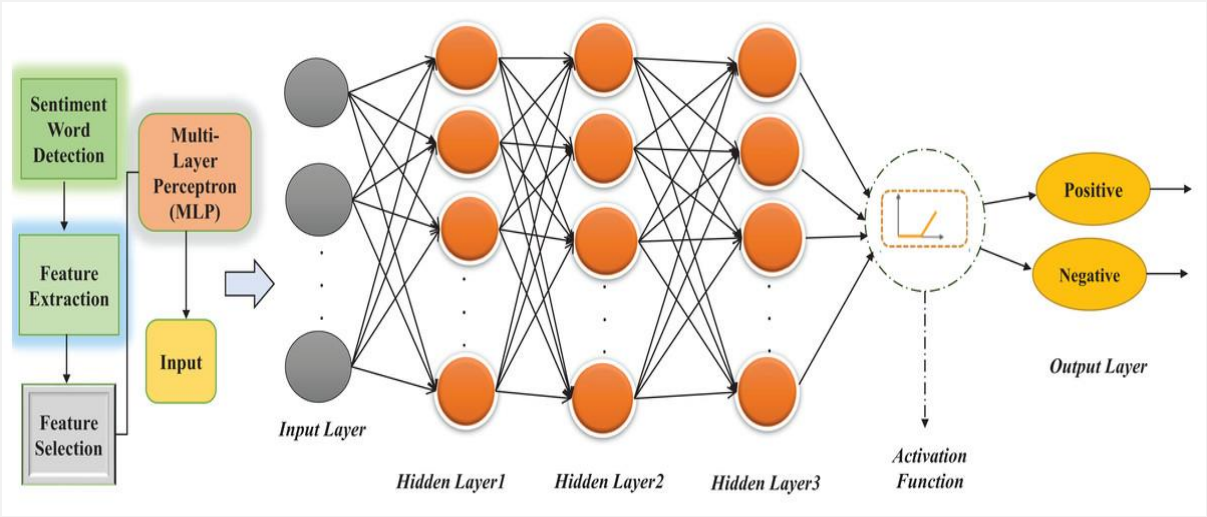


FIG 6.2: DATA FLOW DIAGRAM OF SENTIMENT ANALYSIS

6.3 FLOW- CHART

A flowchart is a visual illustration of a process or device, the usage of symbols to depict the flow of steps in sequential order. It serves as a roadmap to understand the logical collection and interactions within a device. Flowcharts commonly use distinct shapes, such as ovals for start and stop points, rectangles for processes or responsibilities, diamonds for choice points, and arrows to indicate the flow of manipulate from one step to some other.

In software program development, flowcharts are frequently used to map out the logic of algorithms, workflows, or machine techniques. For example, in a sentiment evaluation flowchart, the manner may start with data input (tweets or textual content), observed by using information cleansing and preprocessing, sentiment class (effective, terrible, or impartial), and output visualization. Decision factors can be used to symbolize branching paths, consisting of determining whether or not the statistics is legitimate or if further evaluation is required. By simplifying complex methods into easy-to-recognize visible diagrams, flowcharts help in planning, debugging, and speaking device good judgment effectively.

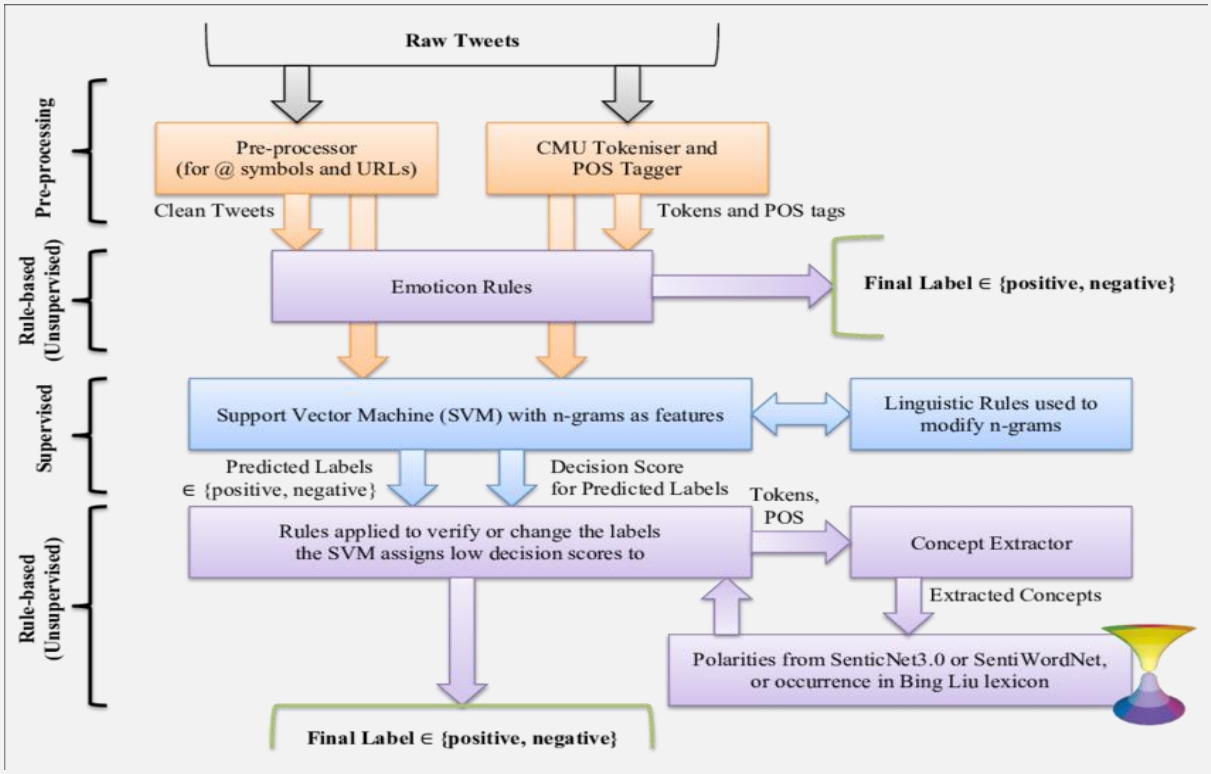


FIG 6.3: Flow Chart of Sentiment Analysis

6.4 USER – CASE DIAGRAM

A use case diagram is a visual illustration of the interactions among customers (actors) and a device, illustrating the various ways the device is utilized to attain specific dreams. It highlights the useful requirements of a system by using showing unique use instances (specific functionalities or tasks) that users can perform. In a use case diagram, the actors are typically outside entities, which includes customers or other structures, and are represented as stick figures. The use cases, represented as ovals, show the important thing moves or operations that the machine supports.

For instance, in a sentiment evaluation gadget, the use case diagram may include actors like Admin and User. The use instances can be "Submit Tweet for Analysis," "View Sentiment Results," and "Generate Reports." The diagram visually demonstrates the interaction between customers and the machine's functionalities, making it less difficult to understand how the device may be used and what functions need to be carried out.

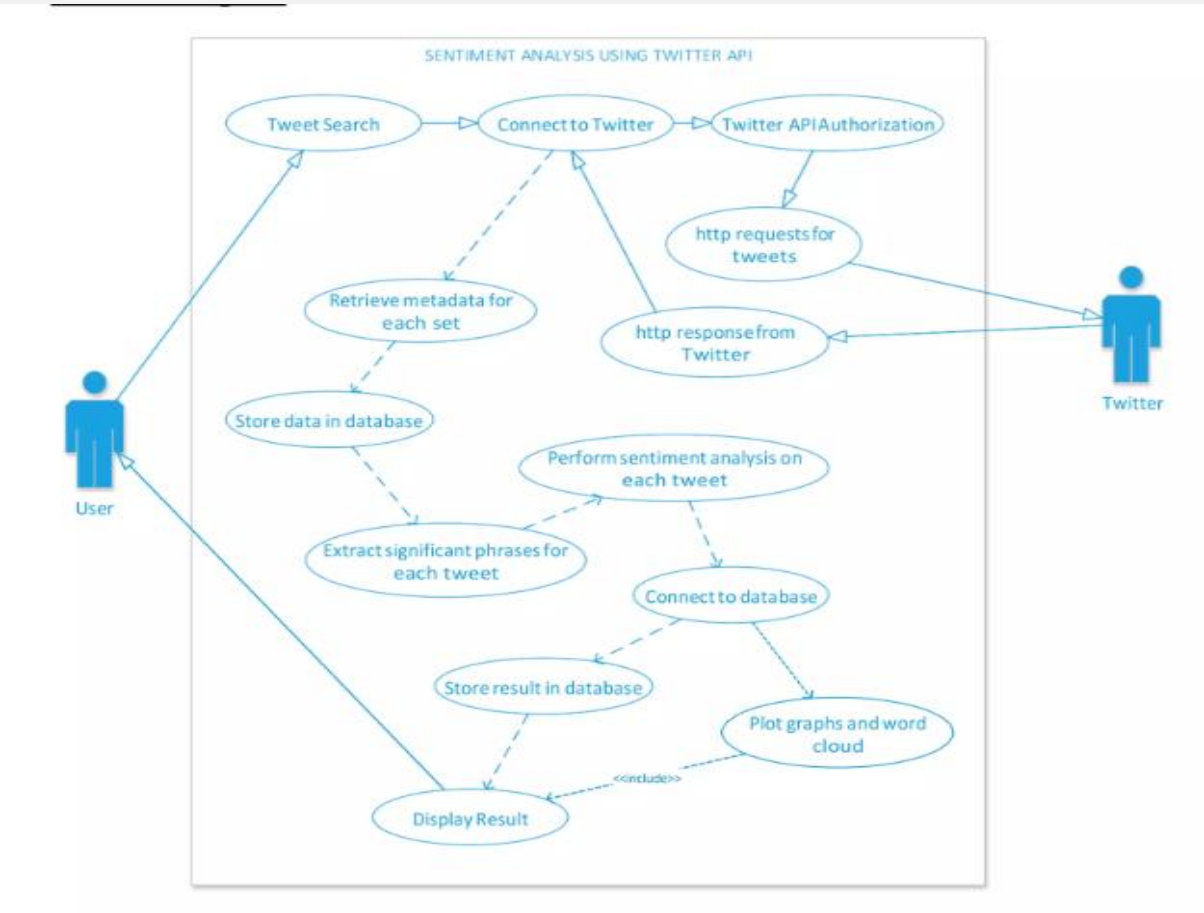


FIG 6.4 : USER CASE DIAGRAM

6.5 BLOCK DIAGRAM

A block diagram is a simplified graphical representation of a gadget, showing the key additives or purposeful blocks and their relationships. Each block generally represents a major part of the gadget, along with a procedure, module, or hardware element, and the lines or arrows among them depict records glide or communique pathways. Unlike more designated diagrams, block diagrams cognizance on the high-degree structure of the system in preference to unique implementation details.

For instance, in a sentiment analysis device, a block diagram might encompass blocks like "Data Collection," "Preprocessing," "Sentiment Classification," and "Output Visualization," with arrows indicating the glide of information between them. This type of diagram is beneficial for know-how the overall architecture and operation of a device at a look, making it helpful all through the layout and making plans levels of a mission.

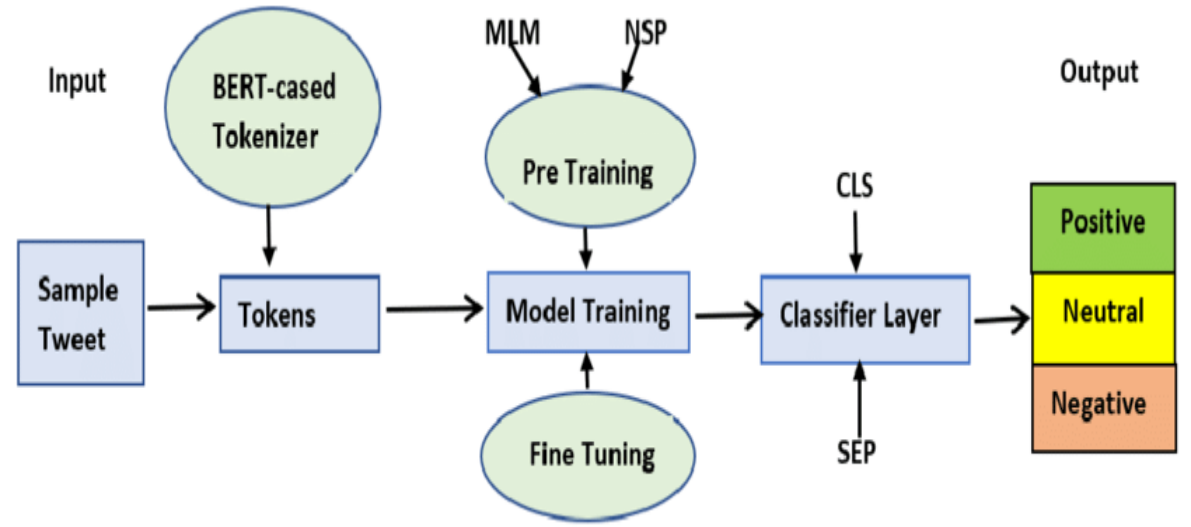


FIG 6.5 : BLOCK DIAGRAM OF SENTIMENT ANALYSIS

6.6 SYSTEM ARCHITECTURE

System structure refers to the high-degree shape of a device, defining how its additives are organized and engage with one another. It outlines the overall layout, which includes hardware, software, data go with the flow, and verbal exchange among unique modules. In a typical software system, the architecture frequently follows nicely-installed models which include purchaser-server, micro services, or layered architectures, depending at the gadget's necessities. The cause of system architecture is to make certain that the machine is scalable, maintainable, and able to meet purposeful and non-practical necessities, which include performance, safety, and reliability. It offers a blueprint that builders and engineers can comply with to enforce the system effectively.

Example in Sentiment Analysis System:

In a sentiment analysis system, for example, the structure might contain one of a kind layers. The records layer is chargeable for accumulating and storing information, together with tweets or other text enter. The processing layer handles responsibilities like statistics cleansing, preprocessing, and making use of device mastering algorithms (e.G., neural networks) to classify sentiment. The utility layer manages the person interface, wherein users can enter data and think about sentiment effects. Additionally, the structure may additionally consist of integration with external APIs, which include Twitter APIs for information collection or third-party visualization gear for supplying results. By organizing those components right into a coherent architecture, the device can function correctly and handle huge amounts of records with minimum performance bottlenecks.

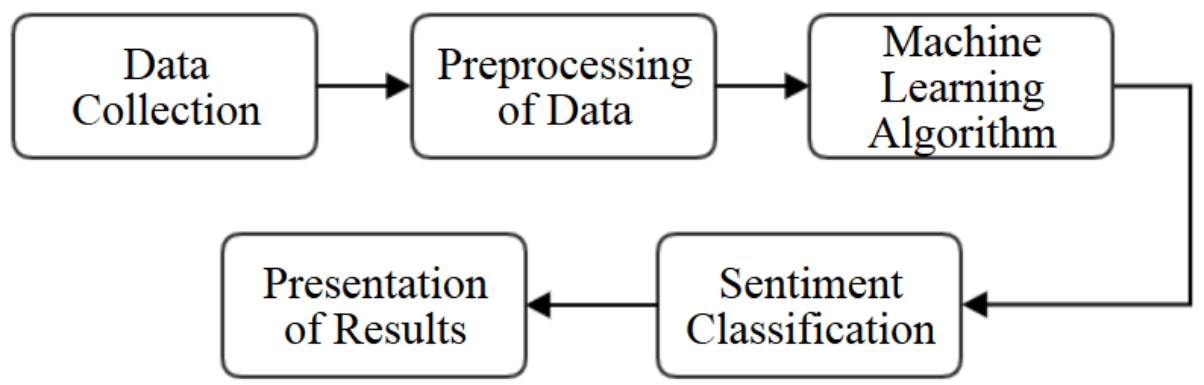


FIG 6.6 : SYSTEM ARCHITECTURE FOR SENTIMENT ANALYSIS

SYSTEM TESTING

7. SYSTEM TESTING

System testing is the process of evaluating a complete, integrated system to ensure it meets the specified requirements. It is conducted after the individual components or modules have been tested during unit and integration testing and involves testing the system as a whole. The primary goal of system testing is to verify that all components work together seamlessly and that the system functions correctly under various conditions, including both functional and non-functional aspects like performance, security, and usability.

In system testing, different types of tests are performed, such as functional testing, which checks if the system's features work as expected, and performance testing, which evaluates how the system behaves under stress or load. Additionally, security testing ensures the system protects data and handles security threats appropriately. By thoroughly testing the system, developers can identify and fix issues before deployment, ensuring the system is reliable and meets user needs.

Importance of System Testing:

System testing is crucial because it ensures that the entire system works as a cohesive unit, minimizing the risk of failures or malfunctions in production. It identifies defects that may not surface during earlier testing stages, such as integration issues or conflicts between modules. Thorough system testing improves system quality and user satisfaction by ensuring that the software meets both functional requirements and performance expectations. By catching and resolving issues early, system testing helps avoid costly fixes and potential downtimes after deployment, ensuring a smooth and stable release of the final product.

7.1 Sentiment Analysis Testing Strategies:

Testing strategies for sentiment analysis systems are essential to ensure the model's accuracy, performance, and reliability. Sentiment analysis involves analyzing text data to classify emotions, opinions, or attitudes (e.g., positive, negative, neutral). Given the complexity of language, testing must cover multiple aspects, from data preprocessing to model evaluation. Below are some key testing strategies used in sentiment analysis:

1. **Data Preprocessing Testing:**

The first step in sentiment analysis is preprocessing the text data. This involves tasks like cleaning the data (removing punctuation, special characters, and stop words), tokenization, and stemming/lemmatization. Testing strategies at this stage include:

- **Data Integrity Testing:** Ensures the preprocessing pipeline correctly handles raw data, and no critical information is lost.
- **Edge Case Testing:** Verifies how the system handles irregular text inputs, such as slang, emojis, hashtags, or mixed languages, which are common in social media data like tweets.
- **Performance Testing:** Assesses whether preprocessing scales well with large datasets, ensuring it doesn't become a bottleneck.

2. **Model Validation Testing:**

The core of sentiment analysis is the machine learning or deep learning model that classifies text sentiment. Testing strategies at this stage involve:

- **Cross-Validation:** Splitting the dataset into training and testing subsets multiple times to ensure the model is not overfitting. This helps in assessing the model's generalization capability.
- **Confusion Matrix Analysis:** Using metrics such as precision, recall, accuracy, and F1-score to evaluate model performance. It provides a detailed view of how well the model classifies each sentiment class (positive, negative, neutral).
- **A/B Testing:** Comparing different model architectures or algorithms (e.g., Logistic Regression vs. Neural Networks) to determine which performs better for sentiment classification.

3. Sentiment Polarity Testing:

Sentiment polarity testing checks how accurately the system detects and classifies sentiment tones (positive, negative, neutral). Strategies include:

- **Threshold Sensitivity Testing**: Adjusting classification thresholds to balance between precision and recall, especially when dealing with neutral sentiments or ambiguous language.
- **Sentiment Drift Testing:** Assessing how well the system adapts to changing sentiment over time, especially in domains like social media where trends or topics evolve rapidly.
- **Multilingual Testing:** If the system supports multiple languages, testing must cover how well it detects sentiment in non-English languages.

4. Real-Time and Batch Testing

Sentiment analysis systems may need to function in real-time (e.g., analyzing live tweets) or in batch mode (processing large datasets). Testing strategies include:

- Latency Testing: For real-time systems, checking how quickly the system can analyze and classify incoming data (e.g., live social media feeds).

- Scalability Testing: For batch processing systems, ensuring the system can handle large volumes of data efficiently without compromising accuracy.

5. Human-AI Feedback Testing

In sentiment analysis, human opinions may not always align with machine-generated sentiments. Testing strategies for handling this include:

- **Human Review Testing:** Comparing the system's predictions with manual sentiment analysis done by human reviewers, identifying discrepancies to improve model accuracy.
- **Sentiment Correction Mechanism Testing:** Implementing and testing feedback loops where human reviewers can correct sentiment predictions, helping the model learn from its mistakes.

7.2 Sentiment Analysis Testing Strategies:

1. **Unit Testing:**

Unit testing is a software program trying out method where person additives or functions of a software program utility are examined in isolation to verify their correctness. The primary goal is to validate that every unit of the code performs as predicted beneath numerous conditions. In the context of a sentiment analysis device, unit checks can also recognition on unique functions inclusive of textual content preprocessing (e.G., tokenization and prevent-phrase elimination), sentiment scoring algorithms, or database interplay techniques. Unit trying out helps become aware of bugs early within the improvement cycle, making it less complicated and much less luxurious to fix them. Developers often use trying out frameworks like JUnit (for Java) or pytest (for Python) to automate and run unit exams efficiently.

2. **Integration Testing:**

Integration checking out entails trying out the interactions among exclusive components or modules of a software program gadget to make certain they work together as meant. This form of checking out usually follows unit testing, wherein character devices are blended and tested as a set. In a sentiment analysis system, integration checks may observe how the facts flows from the statistics collection module through preprocessing to the sentiment classification module. It checks whether or not the additives trade data efficiently and carry out their features cohesively. Integration trying out can identify troubles that may not be obtrusive whilst trying out additives in isolation, together with statistics layout mismatches or unexpected conduct when modules engage. Testing frameworks regularly provide tools for integration testing, allowing builders to simulate aspect interactions successfully.

3. Acceptance Testing:

Acceptance testing is the final phase of testing, performed to determine whether the system meets the specified requirements and is ready for deployment. This testing is typically conducted by end-users or stakeholders and focuses on validating the system's functionality and usability in real-world scenarios. In the case of a sentiment analysis system, acceptance testing would involve verifying that the system correctly analyzes and classifies sentiment based on user inputs, meets performance benchmarks, and provides a user-friendly interface. Acceptance testing can take several forms, including alpha testing (conducted in-house) and beta testing (conducted by a select group of external users). The main objective is to ensure the system fulfills business needs and is acceptable for release, providing stakeholders with confidence in the system's reliability and functionality.

7.3 ERRORS:

Errors in sentiment evaluation testing can significantly effect the reliability and accuracy of the machine. Here are a few commonplace varieties of errors that may get up at some point of the checking out method:

1. Data Quality Issues:

One of the number one assets of mistakes in sentiment analysis is the fine of the input records. Inconsistent, noisy, or poorly classified facts can lead to inaccurate sentiment predictions. For instance, tweets with sarcasm, slang, or ambiguous language can confuse the sentiment class model, ensuing in misclassifications. Additionally, if the training information does now not properly constitute the variety of the goal population, the version may additionally carry out poorly on actual-international information.

2. **Model Overfitting or Underfitting:**

Errors can occur due to version overfitting or underfitting throughout training. Overfitting happens when a model learns the schooling facts too properly, capturing noise and outliers in place of popular patterns, leading to terrible performance on unseen facts. Conversely, underfitting occurs whilst a version is simply too simplistic to seize the underlying patterns in the data, ensuing in excessive mistakes quotes. Both eventualities can lead to unreliable sentiment evaluation effects.

3.**Inadequate Testing Coverage**

Inadequate checking out insurance can result in undiscovered bugs or mistakes inside the sentiment evaluation gadget. If trying out does not cover edge instances, inclusive of unusual textual content formats, multilingual inputs, or area-specific jargon, the gadget might also fail in real-international situations. It is important to consist of a comprehensive set of take a look at cases that account for various enter kinds and conditions to make certain robust performance. Finally, errors can occur during the deployment phase, such as mis configurations, version mismatches, or incompatibilities with external APIs or libraries. These issues can lead to failures in real-time sentiment analysis or discrepancies between testing and production environments, ultimately impacting user experience and trust in the system.

7.4 CODING , TESTING AND IMPLEMENTATION

1. Coding in Sentiment Analysis

In sentiment analysis, coding includes growing the important algorithms and models to procedure textual content facts and classify sentiment. The coding method commonly begins with data collection, in which tweets or textual content inputs are retrieved from platforms like Twitter the use of APIs. Next, the data undergoes preprocessing, which incorporates cleansing, tokenization, and normalization to prepare it for analysis. Developers implement diverse device mastering or deep learning models, including logistic regression, help vector machines, or neural networks (e.G., LSTM or CNN), the usage of frameworks like TensorFlow or Keras. The coding segment also includes writing capabilities for feature extraction, sentiment scoring, and facts visualization, ensuring that the device can efficiently interpret and gift effects.

2. Testing and Implementation

Once the coding is whole, rigorous trying out is carried out to validate the system's performance and accuracy. This consists of unit testing person additives, integration checking out to verify interactions

between modules, and recognition trying out with real person information to make certain the device meets user necessities. Testing goals to become aware of and clear up any problems, together with records first-class troubles or version misclassifications, before deployment. After a hit checking out, the sentiment evaluation system is carried out, generally via a user-friendly interface or dashboard, allowing quit-customers to enter text and acquire sentiment classifications in real time. The implementation section may additionally involve monitoring system performance submit-deployment to ensure continued accuracy and reliability as new statistics is processed.

SCREENSHOTS

8. SCREENSHOTS:

DATA CLEANING:

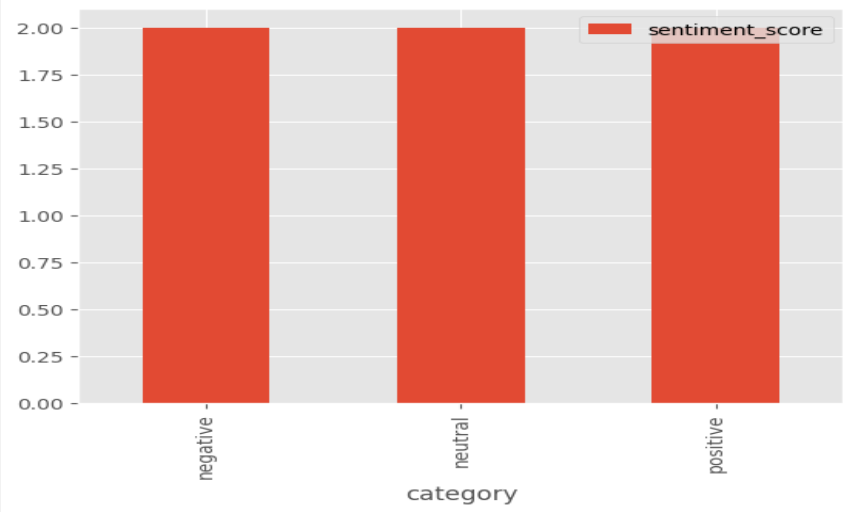
1) HEAD

	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
1048570	4	1960186342	Fri May 29 07:33:44 PDT 2009	NO_QUERY	Madelineducganx	My GrandMa is making Dinenr with my Mum
1048571	4	1960186409	Fri May 29 07:33:43 PDT 2009	NO_QUERY	OffRoad_Dude	Mid-morning snack time... A bowl of cheese noo...
1048572	4	1960186429	Fri May 29 07:33:44 PDT 2009	NO_QUERY	Falchion	@ShaDeLa same here say it like from the Termi...
1048573	4	1960186445	Fri May 29 07:33:44 PDT 2009	NO_QUERY	jonasobsessedx	@DestinyHope92 im great thaanks wbuu?
1048574	4	1960186607	Fri May 29 07:33:45 PDT 2009	NO_QUERY	sugababez	cant wait til her date this weekend

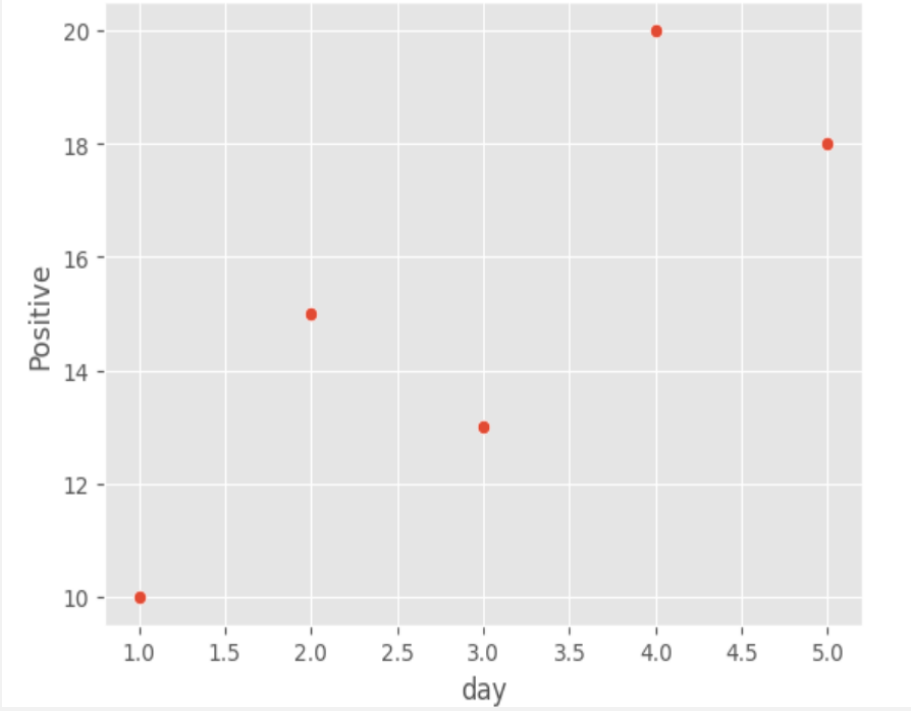
2) TAIL:

	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
1048570	4	1960186342	Fri May 29 07:33:44 PDT 2009	NO_QUERY	Madelineducganx	My GrandMa is making Dinenr with my Mum
1048571	4	1960186409	Fri May 29 07:33:43 PDT 2009	NO_QUERY	OffRoad_Dude	Mid-morning snack time... A bowl of cheese noo...
1048572	4	1960186429	Fri May 29 07:33:44 PDT 2009	NO_QUERY	Falchion	@ShaDeLa same here say it like from the Termi...
1048573	4	1960186445	Fri May 29 07:33:44 PDT 2009	NO_QUERY	jonasobsessedx	@DestinyHope92 im great thaanks wbuu?
1048574	4	1960186607	Fri May 29 07:33:45 PDT 2009	NO_QUERY	sugababez	cant wait til her date this weekend

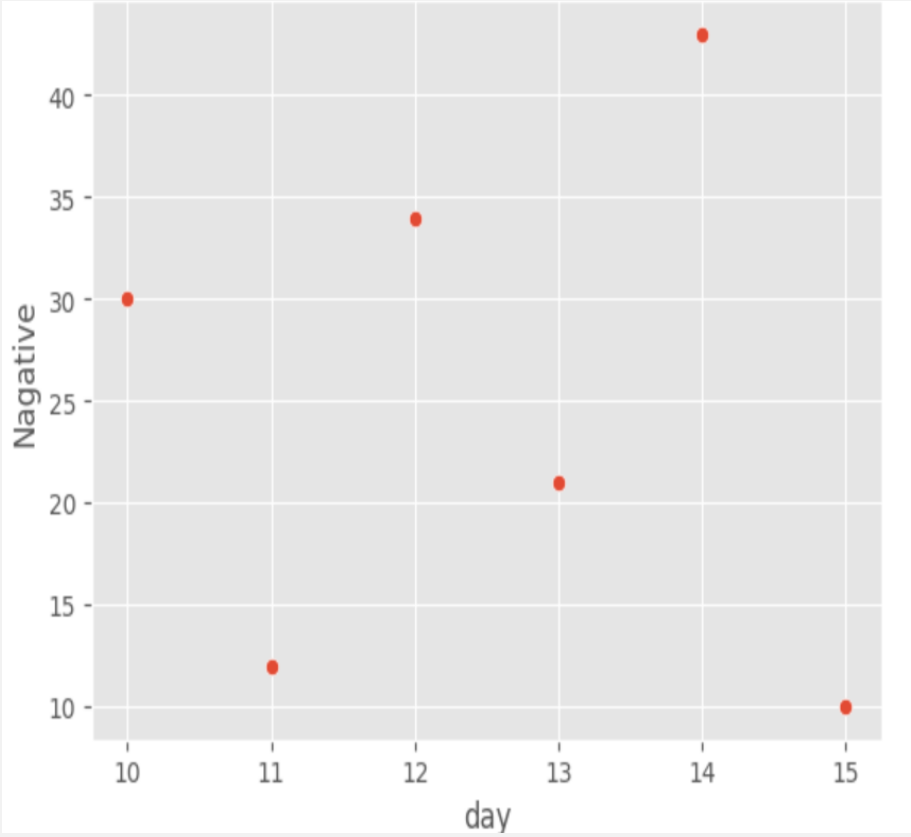
Distribution of Train Tweets:



Positive Tweets In Days:



Negative Tweets In Days:



Model Architecture:

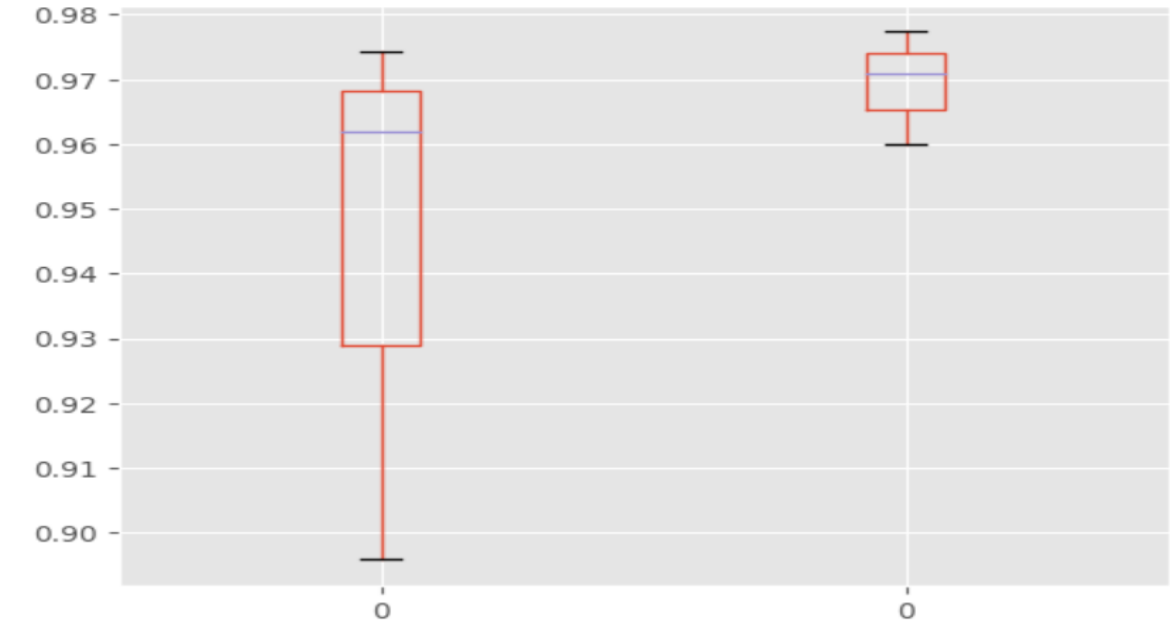
Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 16)	448
max_pooling2d (MaxPooling2D)	(None, 16, 16, 16)	0
conv2d_1 (Conv2D)	(None, 16, 16, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 32)	0
conv2d_2 (Conv2D)	(None, 8, 8, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 64)	0
dropout (Dropout)	(None, 4, 4, 64)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_2 (Dense)	(None, 500)	512,500
dropout_1 (Dropout)	(None, 500)	0
dense_3 (Dense)	(None, 10)	5,010

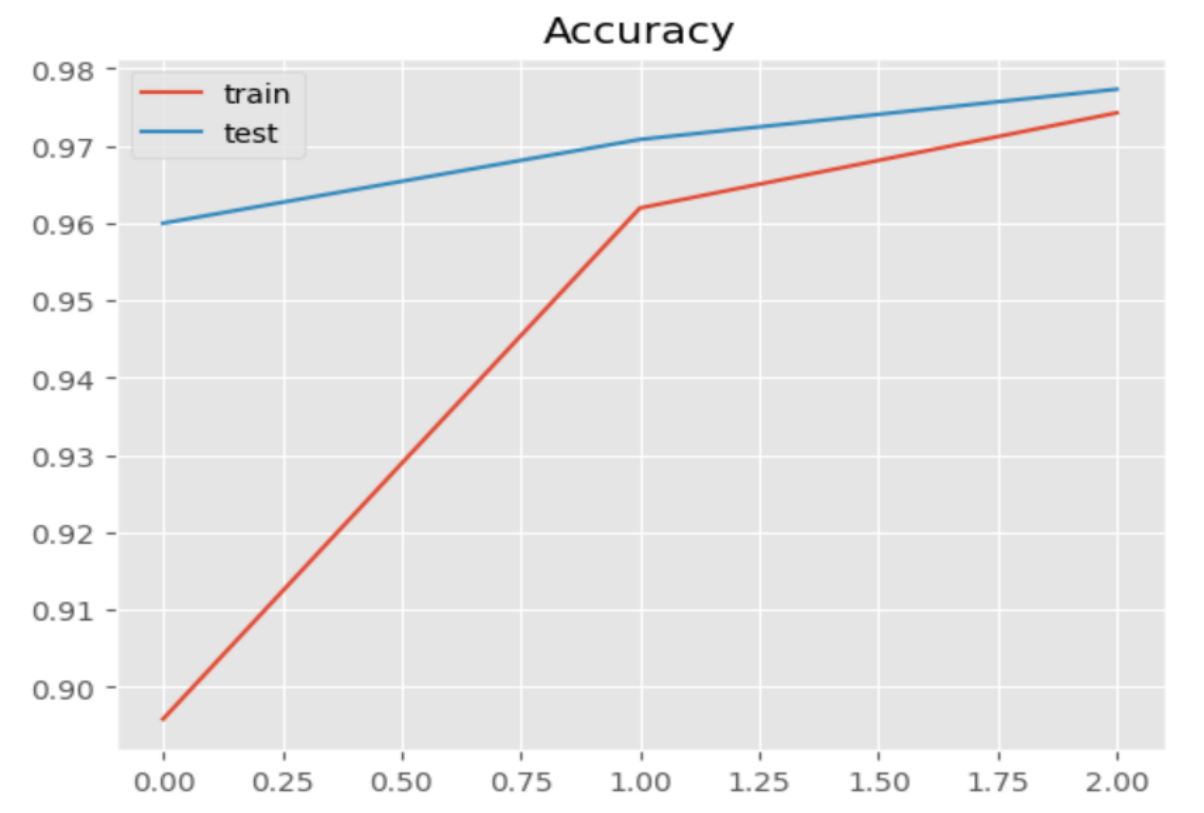
Total params: 541,094 (2.06 MB)
Trainable params: 541,094 (2.06 MB)
Non-trainable params: 0 (0.00 B)

MNIST Dataset Accuracy:

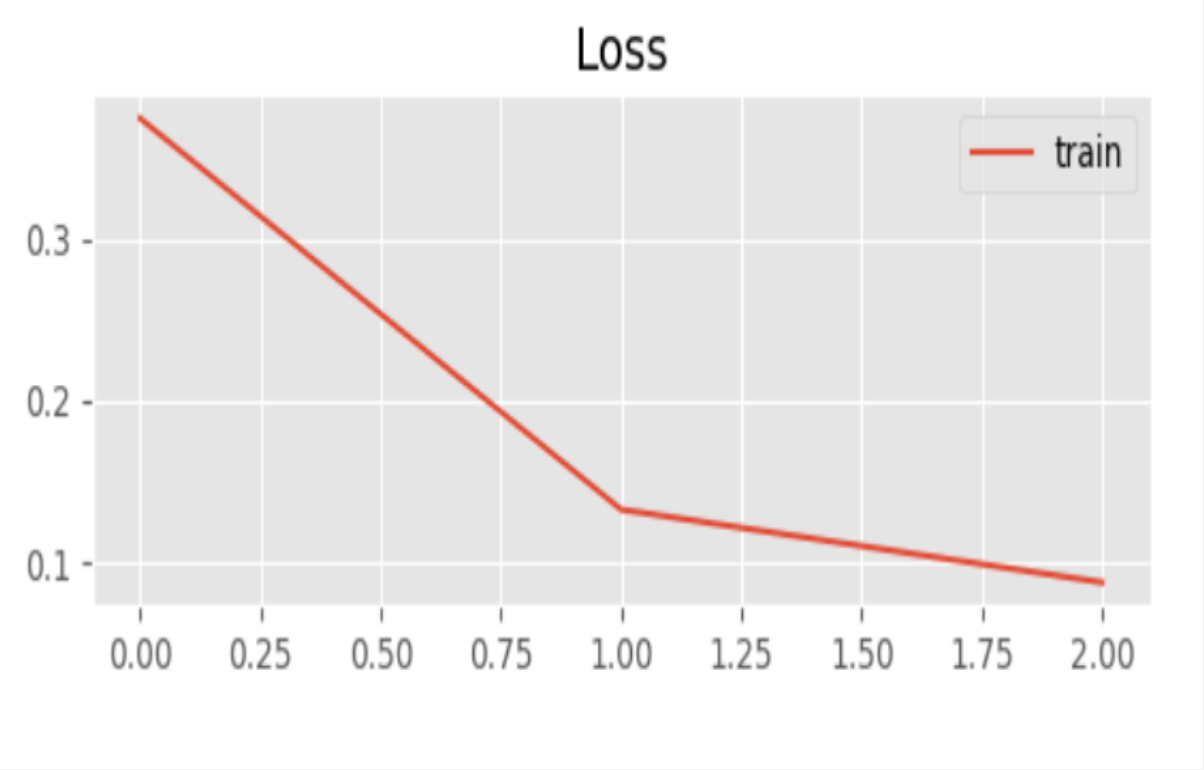
```
[0.9599999785423279, 0.9708333611488342, 0.9773333072662354]
[0.8958518505096436, 0.9619629383087158, 0.9742777943611145]
<Axes: >
```



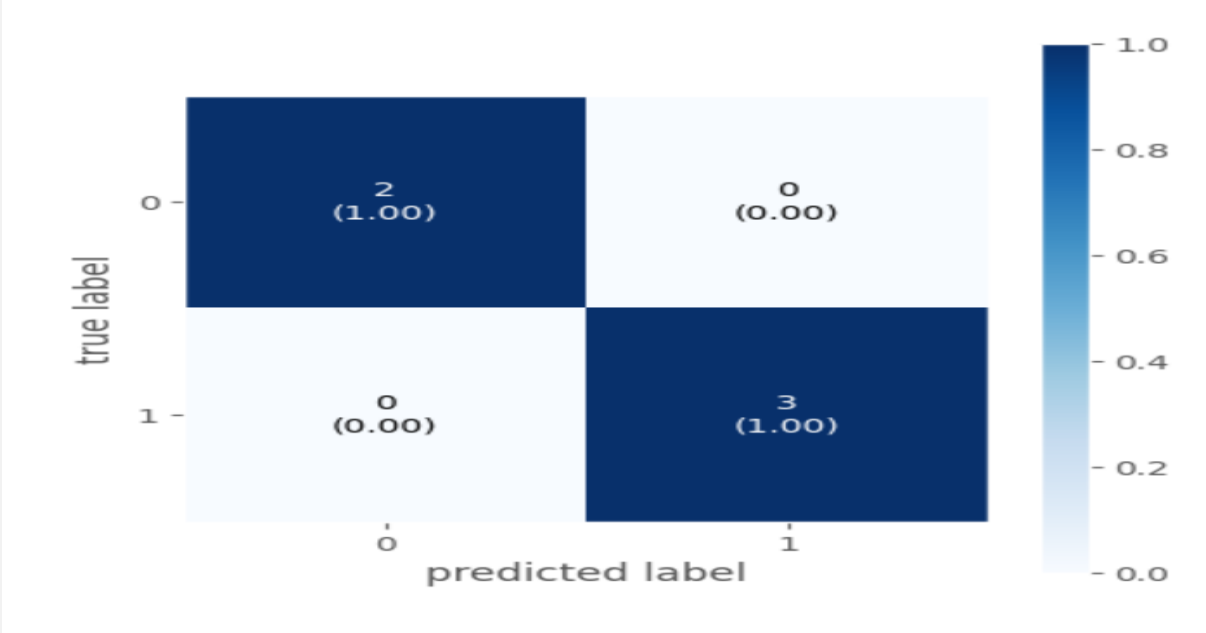
ACCURACY:



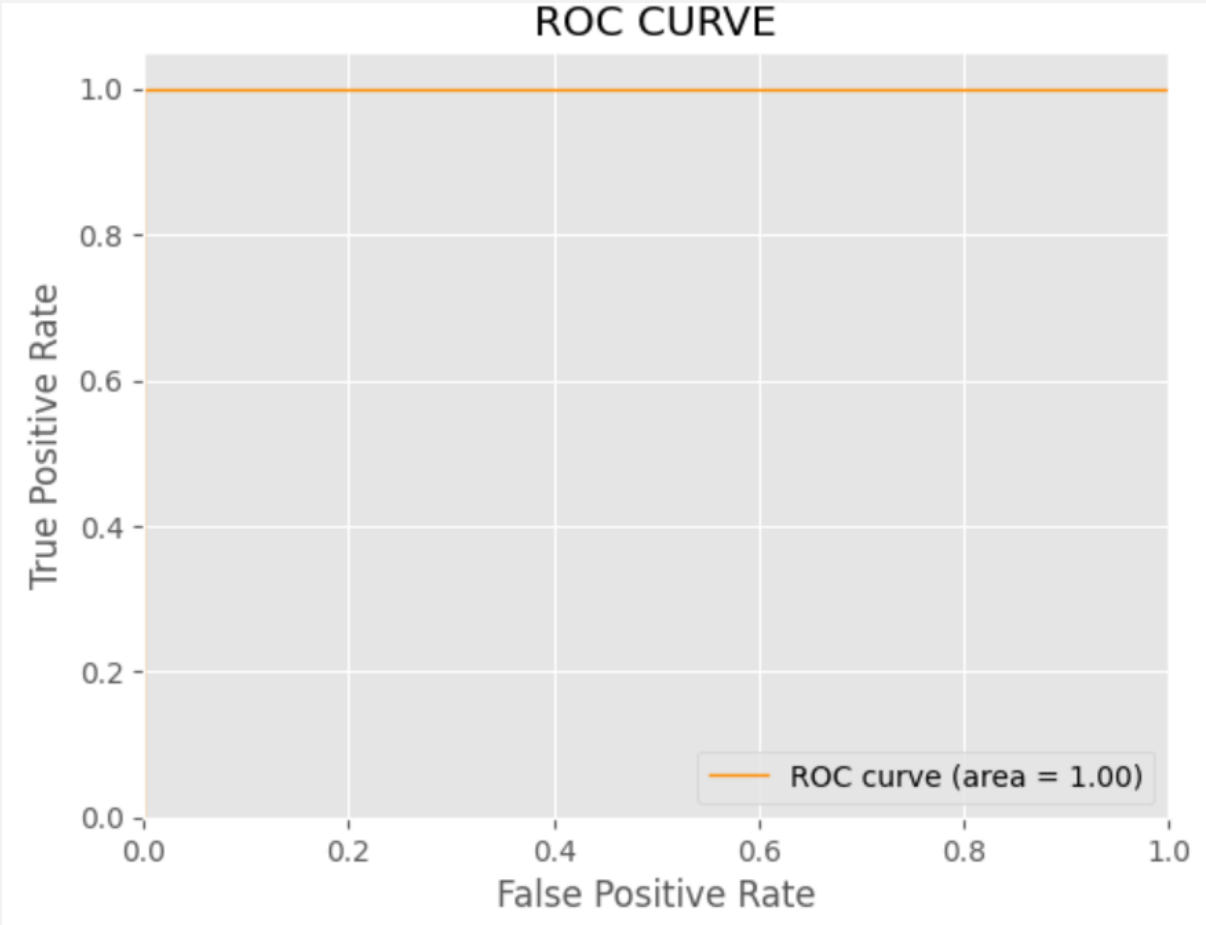
LOSS:



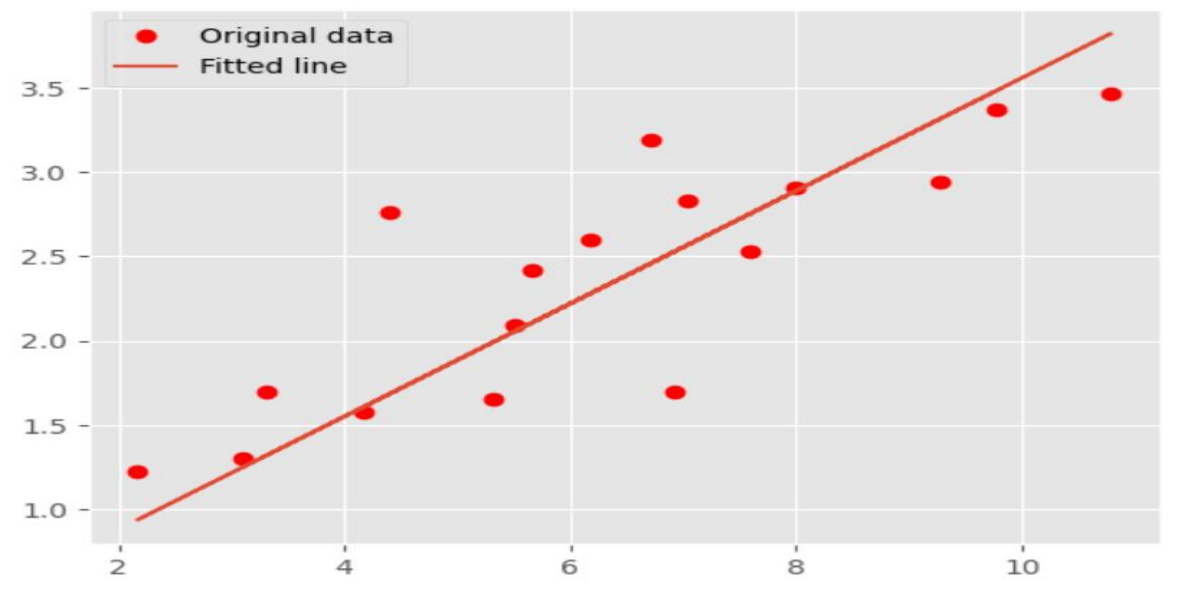
Confusion Matrix:



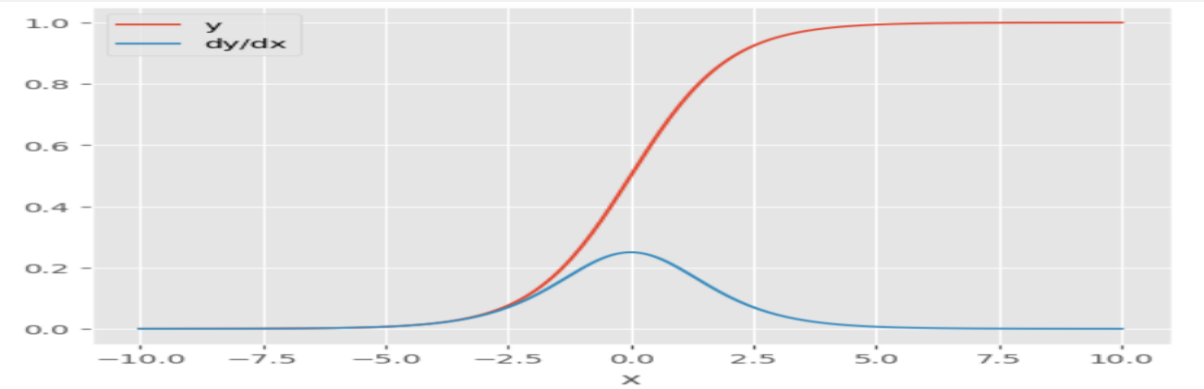
ROC(Receiver Operating Characteristic) Curve:



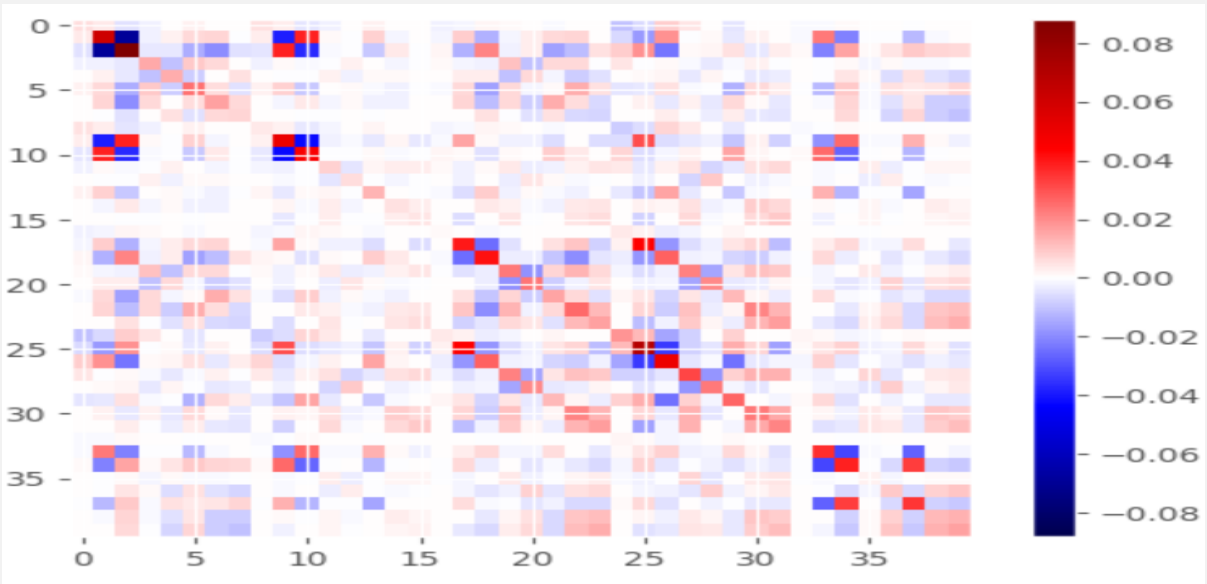
Linear Model:



Auto Differentiation:



Tensor Source:



Built-In-RNN:

Model: "functional_14"

Layer (type)	Output Shape	Param #	Connected to
input_layer_3 (InputLayer)	(None, None)	0	-
input_layer_4 (InputLayer)	(None, None)	0	-
embedding_3 (Embedding)	(None, None, 64)	64,000	input_layer_3[0][0]
embedding_4 (Embedding)	(None, None, 64)	128,000	input_layer_4[0][0]
encoder (LSTM)	[(None, 64), (None, 64), (None, 64)]	33,024	embedding_3[0][0]
decoder (LSTM)	(None, 64)	33,024	embedding_4[0][0], encoder[0][1], encoder[0][2]
dense_10 (Dense)	(None, 10)	650	decoder[0][0]

Total params: 258,698 (1010.54 KB)
Trainable params: 258,698 (1010.54 KB)
Non-trainable params: 0 (0.00 B)

Bidirectional RNN:

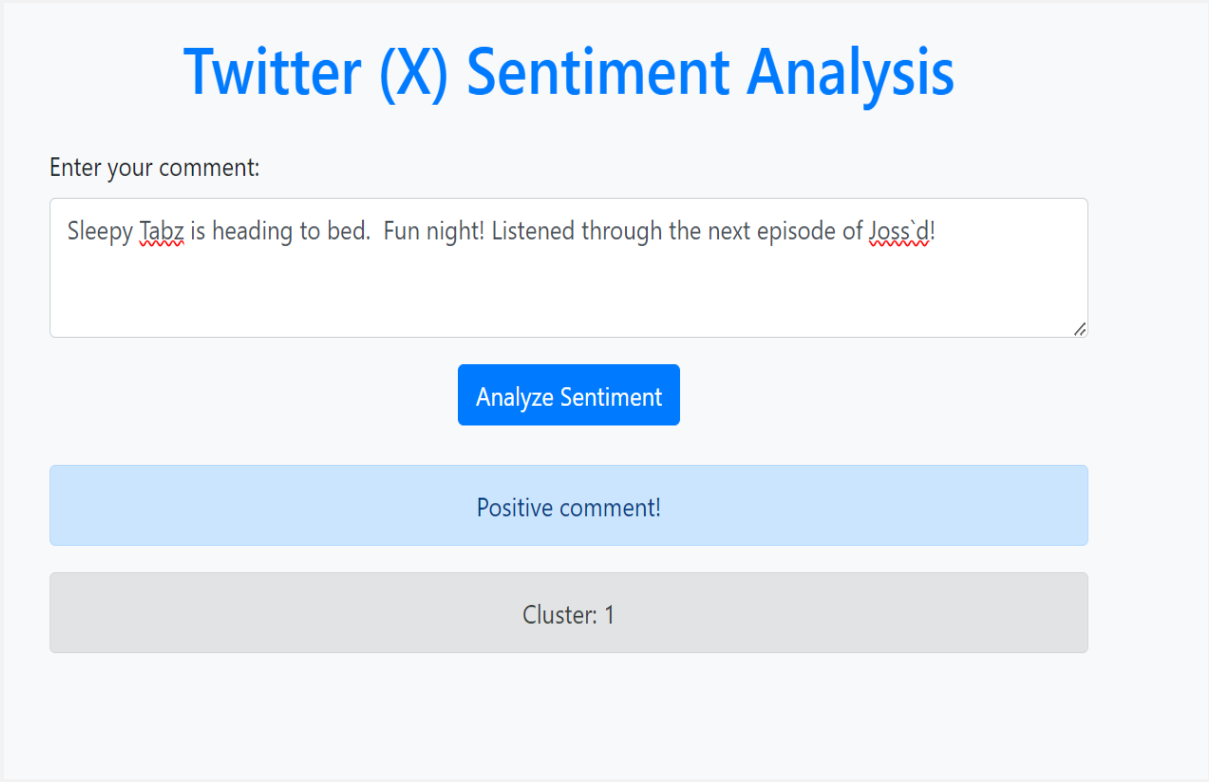
Model: "sequential_4"

Layer (type)	Output Shape	Param #
bidirectional (Bidirectional)	(None, 5, 128)	38,400
bidirectional_1 (Bidirectional)	(None, 64)	41,216
dense_11 (Dense)	(None, 10)	650

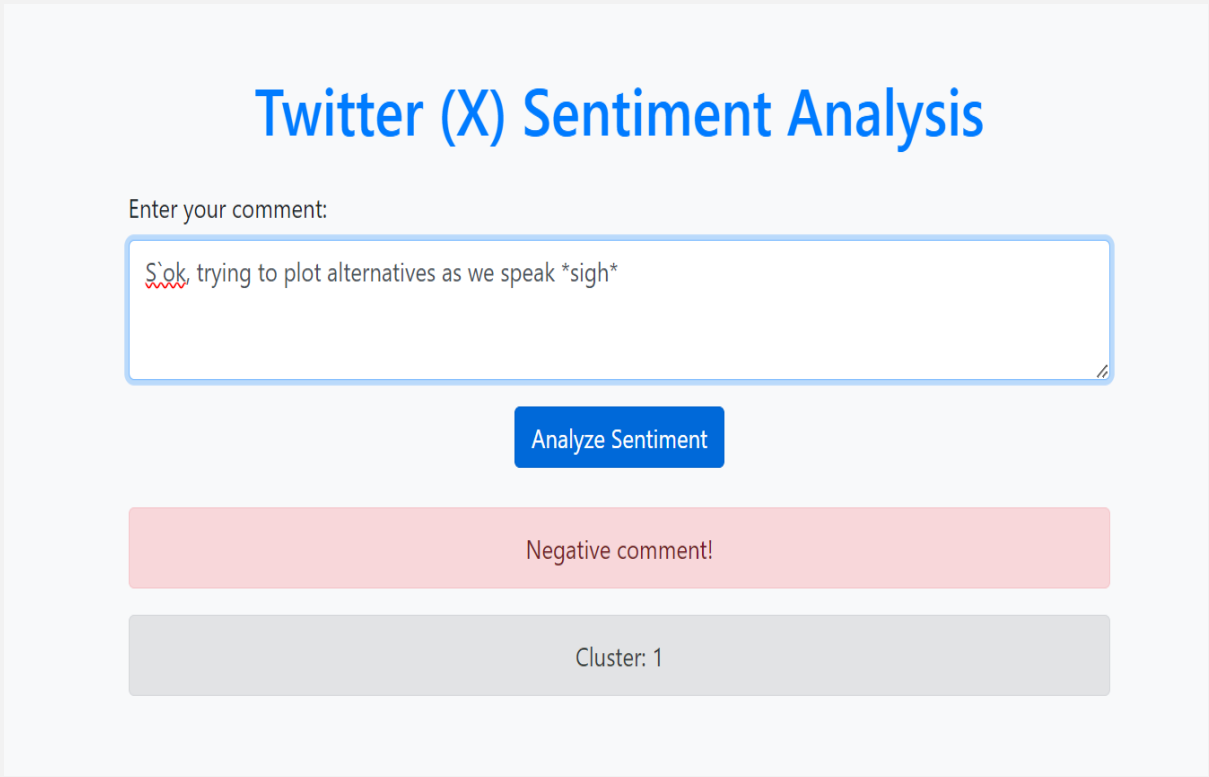
Total params: 80,266 (313.54 KB)
Trainable params: 80,266 (313.54 KB)
Non-trainable params: 0 (0.00 B)

8.1 UI DESIGN:

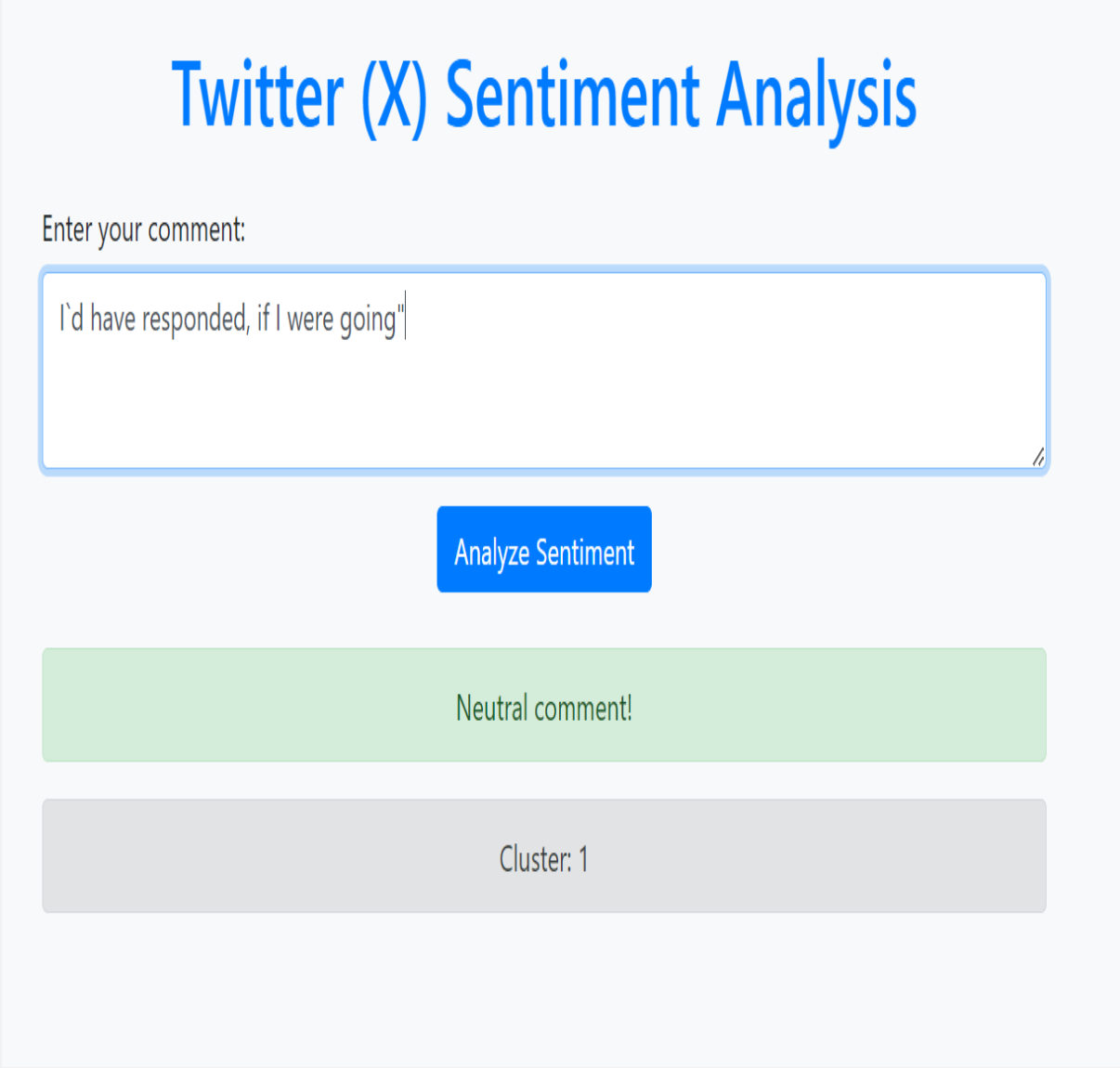
POSITIVE :



NEGATIVE:



NEUTRAL:


Twitter (X) Sentiment Analysis
Enter your comment:
I`d have responded, if I were going"
Analyze Sentiment
Neutral comment!
Cluster: 1

CODING

9. CODING

9.1 Twitter(X) Sentiment Analysis . ipynb

IMPORTING PYTHON LIBRARIES:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from tensorflow import keras
plt.style.use('ggplot')
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import classification_report, confusion_matrix
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.model_selection import train_test_split
from mlxtend.plotting import plot_confusion_matrix
import matplotlib.cm as cm
from matplotlib import rcParams
from collections import Counter
from nltk.tokenize import RegexpTokenizer
import re
import string
from tensorflow.keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing import sequence
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

Data Cleaning:

```
# Load Tweet dataset
df1=pd.read_csv('/content/drive/MyDrive/Project21/Twitter_Dataset.csv',encoding='ISO-8859-1')
df1.head()
df1.tail()
print(df.iloc[0])
print(df.iloc[0:3])



print(train.shape)
print(test.shape)

train.dropna(inplace=True)
train.info()

data = {
   'category': ['positive', 'neutral', 'negative', 'positive', 'negative', 'neutral'],
   'sentiment_score': [0.9, 0.1, -0.5, 0.8, -0.7, 0.0]
}
df = pd.DataFrame(data)
df.groupby('category').count().plot(kind='bar')
plt.show()



import pandas as pd
import numpy as np

data = {
   'column1': [1, 2, None, 4],
   'column2': ['a', 'b', 'c', None]
}
```

```
df = pd.DataFrame(data)
missing_values_count = np.sum(df.isnull().any(axis=1))
print(missing_values_count)

print('Count of columns in the data is:', len(data.columns))
print('Count of rows in the data is:  ', len(data))

mport pandas as pd
data_train = {
   'text': ['This is a sample text.', 'Another example.', 'And one more.']
}

data_test = {
   'text': ['Testing data here.', 'Yet another test.', 'Final example text.']
}

df_train["length"] = df_train["text"].apply(lambda x: len(x))
df_test["length"] = df_test["text"].apply(lambda x: len(x))

print("Train Length Stat")
print(df_train["length"].describe())

print("\nTest Length Stat")
print(df_test["length"].describe())
```

Data Pre-Processing:

```
import pandas as pd
data_list = [{'text': 'sample text 1', 'label': 0},
            {'text': 'sample text 2', 'label': 1},
            {'text': 'sample text 3', 'label': 0}]

# Convert list to DataFrame
data = pd.DataFrame(data_list)
```

```
# Select specific columns
data = data[['text', 'label']]
print(data)

data['label'][data['label']==4]=1
data_pos = data[data['label'] == 1]
data_neg = data[data['label'] == 0]

data_pos = data_pos.iloc[:int(20000)]
data_neg = data_neg.iloc[:int(20000)]

data = pd.concat([data_pos, data_neg])
data['text']=data['text'].str.lower()
data['text'].tail()

import nltk
nltk.download('stopwords')
stopwords_list = stopwords.words('english')

import nltk
nltk.download('stopwords')

stopwords_list = stopwords.words('english')

STOPWORDS = set(stopwords.words('english'))
def cleaning_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])
data['text'] = data['text'].apply(lambda text: cleaning_stopwords(text))
data['text'].head()

english_punctuations = string.punctuation
punctuations_list = english_punctuations
def cleaning_punctuations(text):
    translator = str.maketrans('', '', punctuations_list)
```

```
return text.translate(translator)
data['text']= data['text'].apply(lambda x: cleaning_punctuations(x))
data['text'].tail()

def cleaning_email(data):
    return re.sub('@[^\s]+', ' ', data)
data['text']= data['text'].apply(lambda x: cleaning_email(x))
data['text'].tail()

def cleaning_URLs(data):
    return re.sub('((www\.[^\s]+)|(https?://[^\s]+))',' ',data)
data['text'] = data['text'].apply(lambda x: cleaning_URLs(x))
data['text'].tail()

def cleaning_numbers(data):
    return re.sub('[0-9]+', '', data)
data['text'] = data['text'].apply(lambda x: cleaning_numbers(x))
data['text'].tail()



st = nltk.PorterStemmer()
def stemming_on_text(data):
    text = [st.stem(word) for word in data]
    return data

data['text']= data['text'].apply(lambda x: stemming_on_text(x))

import nltk
nltk.download('wordnet')

lm = nltk.WordNetLemmatizer()
def lemmatizer_on_text(data):
    text = [lm.lemmatize(word) for word in data]
    return data
```

```
data['text'] = data['text'].apply(lambda x: lemmatizer_on_text(x))

import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('ggplot')
#import datasets
tips = sns.load_dataset('tips')
iris = sns.load_dataset('iris')

positive = pd.DataFrame({
    'day': [1, 2, 3, 4, 5],
    'Positive': [10, 15, 13, 20, 18]
})
sns.scatterplot(data=positive, x='day', y='Positive')
plt.show()

Negative = pd.DataFrame({
    'day': [10,11,12,13,14,15],
    'Nagative': [30,12,34,21,43,10]
})
sns.scatterplot(data=Negative, x='day', y='Nagative')
plt.show()

import seaborn as sns
import matplotlib.pyplot as plt
iris = sns.load_dataset('iris')
g = sns.PairGrid(data=iris, hue='species')
g.map_diag(sns.boxplot)
g.map_offdiag(sns.kdeplot)

g.add_legend()
plt.show()
```

Building LSTM Model:

```
# Read file from gdrive
import pandas as pd

import numpy as np
data=pd.read_csv('/content/drive/MyDrive/Project21/Twitter_Dataset.csv', encoding='ISO-8859-1')

MAX_NB_WORDS = 50000
MAX_SEQUENCE_LENGTH = 250
EMBEDDING_DIM = 100

column_names = ['ID', 'Timestamp', 'Date', 'Query', 'Username', 'Tweet']
data = pd.read_csv('/content/drive/MyDrive/Project21/Twitter_Dataset.csv', encoding='ISO-8859-1', names=column_names)

from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=MAX_NB_WORDS, filters='!"#$%&()*+,-./:;<=>?@[\]^_`{|}~', lower=True)
tokenizer.fit_on_texts(data['Tweet'].values)
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))

from tensorflow.keras.preprocessing.sequence import pad_sequences
X = tokenizer.texts_to_sequences(data['Tweet'].values)
X = pad_sequences(X, maxlen=MAX_SEQUENCE_LENGTH)
print('Shape of data tensor:', X.shape)

from sklearn.preprocessing import LabelEncoder
mlb = LabelEncoder()
sentiment = data['Query'].to_numpy()
mlb.fit(sentiment)
Y = mlb.transform(sentiment)
print('Shape of label tensor:', Y.shape)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.10, stratify=Y, random_state = 42)
print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
```

Building CNN Model:

```
# Train-Test Split
from sklearn.model_selection import train_test_split
training_bs, test = train_test_split(data, test_size=0.10 random_state=42)

training_bs.loc[training_bs['Tweet']=='positive', 'sentiment_score'] = int(1)
training_bs.loc[training_bs['Tweet']=='negative', 'sentiment_score'] = int(2)
training_bs.loc[training_bs['Tweet']=='no_sentiment', 'sentiment_score'] = int(3)
training_bs.loc[training_bs['Tweet']=='sadness', 'sentiment_score'] = int(4)
training_bs.loc[training_bs['Tweet']=='fear', 'sentiment_score'] = int(5)

training_bs.loc[training_bs['Tweet']=='trust', 'sentiment_score'] = int(6)
training_bs.loc[training_bs['Tweet']=='anger', 'sentiment_score'] = int(7)
training_bs.loc[training_bs['Tweet']=='surprise', 'sentiment_score'] = int(8)
training_bs.loc[training_bs['Tweet']=='joy', 'sentiment_score'] = int(9)
training_bs.loc[training_bs['Tweet']=='disgust', 'sentiment_score'] = int(10)

all_training_words = ''.join([ word for tokens in training_bs["Tweet"] for word in tokens])
training_sentence_lengths = [len(tokens) for tokens in training_bs["Tweet"]]
TRAINING_VOCAB = sorted(list(set(all_training_words)))
print("%s words total, with a vocabulary size of %s" % (len(all_training_words), len(TRAINING_VOCAB)))

all_test_words = ''.join([word for tokens in test["Tweet"] for word in tokens])
test_sentence_lengths = [len(tokens) for tokens in test["Tweet"]]
TEST_VOCAB = sorted(list(set(all_test_words)))
print("%s words total, with a vocabulary size of %s" % (len(all_test_words), len(TEST_VOCAB)))
print("Max sentence length is %s" % max(test_sentence_lengths))
```

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences

max_vocab_size = 10000  # Number of unique words in the dataset
embedding_dim = 100     # Dimensionality of the embedding vectors
max_sequence_length = 200  # Maximum length of input sequences

X_train = np.random.randint(1, max_vocab_size, size=(1000, max_sequence_length))
y_train = np.random.randint(0, 2, size=(1000,))

model = Sequential()
model.add(Embedding(input_dim=max_vocab_size,
output_dim=embedding_dim, input_length=max_sequence_length))
model.add(Conv1D(filters=128, kernel_size=5, activation='relu'))
model.add(MaxPooling1D(pool_size=4))
model.add(Flatten())
model.add(Dense(10, activation='relu'))

model.add(Dense(1, activation='sigmoid'))  # Binary classification
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=3, batch_size=32, validation_split=0.2)

from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
model = Sequential()
model.add(Conv2D(filters=16, kernel_size=3, padding='same', activation='relu',
 input_shape=(32, 32, 3)))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
```

```
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(10, activation='softmax'))
model.summary()
```

MNIST DATASET:

```
from tensorflow.keras.datasets import mnist
(xtrain,ytrain),(xtest,ytest)=mnist.load_data()

from tensorflow.keras.datasets import mnist
# use Keras to import pre-shuffled MNIST database
(X_train, y_train), (X_test, y_test) = mnist.load_data()
print("The MNIST database has a training set of %d examples." % len(X_train))
print("The MNIST database has a test set of %d examples." % len(X_test))
xtrain.shape
plt.imshow(xtrain[1,:,:],cmap='gray')

def visualize_input(img, ax):
    ax.imshow(img, cmap='gray')
    width, height = img.shape
    thresh = img.max()/2.5
    for x in range(width):
        for y in range(height): #Fixed indentation for nested for loop
            ax.annotate(str(round(img[x][y],2)), xy=(y,x),

horizontalalignment='center',
verticalalignment='center',
color='white' if img[x][y]<thresh else 'black')
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(111)
visualize_input(X_train[0], ax)
```

```
ytrain[:50]
L=pd.DataFrame(ytrain)
L[0].value_counts()

#Represent Training & Testing samples suitable for #tensorflow backend
x_train=xtrain.reshape(xtrain.shape[0],784).astype('float32')
x_test=xtest.reshape(xtest.shape[0],784).astype('float32')

x_test.shape
x_train/=255
x_test/=255

from tensorflow import keras
y_train = keras.utils.to_categorical(ytrain, 10)
y_test = keras.utils.to_categorical(ytest, 10)



from tensorflow.keras.models import Sequential
model = Sequential()
model.add(Dense(784,activation='relu'))
model.add(Dense(100, activation ='relu'))
model.add(Dense(10,activation='softmax'))

model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
epochs = 3

batch_size = 512
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=.1, verbose=True)
loss,accuracy  = model.evaluate(x_test, y_test, verbose=False)

print(history.history['val_accuracy'])
print(history.history['accuracy'])
ta = pd.DataFrame(history.history['accuracy'])
```

```
va = pd.DataFrame(history.history['val_accuracy'])
tva = pd.concat([ta,va] , axis=1)
tva.boxplot()

loss, acc = model.evaluate(x_test, y_test, verbose=0)
print('Accuracy: %.3f'  % acc)
print('Loss: %.3f' % loss)

import numpy as np
x_test_flattened = x_test.reshape(-1, 28 * 28) / 255.0  # Flatten to (num_samples, 784) and normalize
y_predict = model.predict(x_test_flattened)
print(y_predict[0])

from sklearn import metrics
y_pred = []
for val in y_predict:
    y_pred.append(np.argmax(val))
cm = metrics.confusion_matrix(ytest,y_pred)
print(cm)
cr=metrics.classification_report(ytest,y_pred)
print(cr)
```

Implementing Tensor Flow and Keras:

```
def tensorflow_based_model():
    inputs = Input(name='inputs',shape=[max_len])
    layer = Embedding(2000,50,input_length=max_len)(inputs)
    layer = LSTM(64)(layer)
    layer = Dense(256,name='FC1')(layer)
    layer = Activation('relu')(layer)
    layer = Dropout(0.5)(layer)
    layer = Dense(1,name='out_layer')(layer)
    layer = Activation('sigmoid')(layer)
    model = Model(inputs=inputs,outputs=layer)
    return model
```

```
model = tensorflow_based_model()
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])

plt.title('Accuracy')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='test')
plt.legend()
plt.show();

%matplotlib inline
from matplotlib import pyplot
from numpy import where
pyplot.subplot(211)
pyplot.title('Loss')
pyplot.plot(history.history['loss'], label='train')
pyplot.legend()

import numpy as np
from sklearn.metrics import confusion_matrix
Y_test = np.array([0, 1, 1, 0, 1])
y_pred = [0, 1, 1, 0, 1]
y_pred = np.array(y_pred)
print(f"Shape of Y_test: {Y_test.shape}")
print(f"Length of Y_test: {len(Y_test)}")
print(f"Shape of y_pred: {y_pred.shape}")
print(f"Length of y_pred: {len(y_pred)}")
if len(Y_test) != len(y_pred):
    raise ValueError("Y_test and y_pred must have the same number of samples.")
CR = confusion_matrix(Y_test, y_pred)
print("Confusion Matrix:")
print(CR)
print('\n')
print("confusion matrix")
print('\n')
```

```
CR=confusion_matrix(Y_test, y_pred)
print(CR)
print('\n')
fig, ax = plot_confusion_matrix(conf_mat=CR,figsize=(5, 5),
                                show_absolute=True,
                                show_normed=True,
                                colorbar=True)
plt.show()

fpr, tpr, thresholds = roc_curve(Y_test, y_pred)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')

plt.legend(loc="lower right")
plt.show()



import tensorflow as tf
train_X = [3.3, 4.4, 5.5, 6.71, 6.93, 4.168, 9.779, 6.182, 7.59, 2.167,
        7.042, 10.791, 5.313, 7.997, 5.654, 9.27, 3.1]
train_Y = [1.7, 2.76, 2.09, 3.19, 1.694, 1.573, 3.366, 2.596, 2.53, 1.221,
        2.827, 3.465, 1.65, 2.904, 2.42, 2.94, 1.3]
NUM_EXAMPLES = len(train_X)



#create model paramters with initial values
W = tf.Variable(0.)
b = tf.Variable(0.)
```

```
train_steps = 100

learning_rate = 0.01
for i in range(train_steps):
  with tf.GradientTape() as tape:
    yhat = train_X * W + b
    error = yhat - train_Y
    loss = tf.reduce_mean(tf.square(error))
  dW, db = tape.gradient(loss, [W, b])
  W.assign_sub(dW * learning_rate)
  b.assign_sub(db* learning_rate)
  if i % 20 == 0:

print("Loss at step {:03d}: {:.3f}".format(i, loss))
print(f'W : {W.numpy()} , b  = {b.numpy()} ')

%matplotlib inline
import matplotlib.pyplot as plt
# Graphic display
plt.plot(train_X, train_Y, 'ro', label='Original data')
plt.plot(train_X, np.array(W * train_X + b), label='Fitted line')
plt.legend()
plt.show()

def linear_model(X_val):
  print((W*X_val + b).numpy())
  linear_model(8)
```

Tensors Slicing:

```
import tensorflow as tf
import numpy as np
t1 = tf.constant([0, 1, 2, 3, 4, 5, 6, 7])

print(tf.slice(t1,
```

```
            begin=[1],
            size=[3]))

print(t1[1:4])
print(t1[-3:])

print(tf.gather(t1,
             indices=[0, 3, 6]))
t1[::3]



t2 = tf.constant([[0, 1, 2, 3, 4],
              [5, 6, 7, 8, 9],
              [10, 11, 12, 13, 14],
              [15, 16, 17, 18, 19]])
print(t2[:-1, 1:3])



alphabet = tf.constant(list('abcdefghijklmnopqrstuvwxyz'))
print(tf.gather(alphabet,
            indices=[2, 0, 19, 18]))

x = tf.Variable(2.0)
y = tf.Variable(3.0)
with tf.GradientTape() as t:
  x_sq = x * x
  with t.stop_recording():
    y_sq = y * y
  z = x_sq + y_sq
grad = t.gradient(z, {'x': x, 'y': y})
print('dz/dx:', grad['x'])
print('dz/dy:', grad['y'])

x = tf.linspace(-10.0, 10.0, 200+1)
```

```
delta = tf.Variable(0.0)
with tf.GradientTape() as tape:
  y = tf.nn.sigmoid(x+delta)



dy_dx = tape.jacobian(y, delta)
print(y.shape)
print(dy_dx.shape)

plt.plot(x.numpy(), y, label='y')
plt.plot(x.numpy(), dy_dx, label='dy/dx')
plt.legend()
_ = plt.xlabel('x')

x = tf.random.normal([7, 5])
layer1 = tf.keras.layers.Dense(8, activation=tf.nn.relu)
layer2 = tf.keras.layers.Dense(6, activation=tf.nn.relu)
with tf.GradientTape() as t2:
  with tf.GradientTape() as t1:
    x = layer1(x)
    x = layer2(x)
    loss = tf.reduce_mean(x**2)
g = t1.gradient(loss, layer1.kernel)
h = t2.jacobian(g, layer1.kernel)

print(f'layer.kernel.shape: {layer1.kernel.shape}')
print(f'h.shape: {h.shape}')



n_params = tf.reduce_prod(layer1.kernel.shape)
g_vec = tf.reshape(g, [n_params, 1])
h_mat = tf.reshape(h, [n_params, n_params])

def imshow_zero_center(image, **kwargs):
```

```
  lim = tf.reduce_max(abs(image))
  plt.imshow(image, vmin=-lim, vmax=lim, cmap='seismic', **kwargs)
  plt.colorbar()
imshow_zero_center(h_mat)
```

Working with RNN:

```
model = keras.Sequential()
model.add(layers.Embedding(input_dim=1000, output_dim=64))
model.add(layers.LSTM(128))
model.add(layers.Dense(10))
model.summary()

encoder_vocab = 1000
decoder_vocab = 2000
encoder_input = layers.Input(shape=(None,))
encoder_embedded = layers.Embedding(input_dim=encoder_vocab, output_dim=64)(
    encoder_input
)
output, state_h, state_c = layers.LSTM(64, return_state=True, name="encoder")(
    encoder_embedded
)
encoder_state = [state_h, state_c]
decoder_input = layers.Input(shape=(None,))
decoder_embedded = layers.Embedding(input_dim=decoder_vocab, output_dim=64)(
    decoder_input
)
decoder_output = layers.LSTM(64, name="decoder")(
    decoder_embedded, initial_state=encoder_state
)
output = layers.Dense(10)(decoder_output)
model = keras.Model([encoder_input, decoder_input], output)
model.summary()
```

```
model = keras.Sequential()
model.add(
    layers.Bidirectional(layers.LSTM(64, return_sequences=True), input_shape=(5, 10))
)

model.add(layers.Bidirectional(layers.LSTM(32)))
model.add(layers.Dense(10))
model.summary()
```

Working With Pre-Processing Layers:

```
data = np.array(
    [
        [0.1, 0.2, 0.3],
        [0.8, 0.9, 1.0],
        [1.5, 1.6, 1.7],
    ]
)
layer = layers.Normalization()
layer.adapt(data)
normalized_data = layer(data)
print("Features mean: %.2f" % (normalized_data.numpy().mean()))
print("Features std: %.2f" % (normalized_data.numpy().std()))

vocab = ["a", "b", "c", "d"]
data = tf.constant([["a", "c", "d"], ["d", "z", "b"]])
layer = layers.StringLookup(vocabulary=vocab)
vectorized_data = layer(data)
print(vectorized_data)

data = tf.constant([["a"], ["b"], ["c"], ["b"], ["c"], ["a"]])
lookup = layers.StringLookup(output_mode="one_hot")
lookup.adapt(data)
test_data = tf.constant([["a"], ["b"], ["c"], ["d"], ["e"], [""]])
encoded_data = lookup(test_data)
```

```
print(encoded_data)


adapt_data = tf.constant(
    [
        "The Brain is wider than the Sky",


 "For put them side by side",
"The one the other will contain",
 "With ease and You beside",
    ]
)
text_vectorizer = layers.TextVectorization(output_mode="tf-idf", ngrams=2)
text_vectorizer.adapt(adapt_data)
print(
    "Encoded text:\n",
    text_vectorizer(["The Brain is deeper than the sea"]).numpy(),
)
inputs = keras.Input(shape=(text_vectorizer.vocabulary_size(),))
outputs = layers.Dense(1)(inputs)
model = keras.Model(inputs, outputs)
train_dataset = tf.data.Dataset.from_tensor_slices(
    (["The Brain is deeper than the sea", "for if they are held Blue to Blue"], [1, 0])
)
train_dataset = train_dataset.batch(2).map(lambda x, y: (text_vectorizer(x), y))
print("\nTraining model...")
model.compile(optimizer="rmsprop", loss="mse")
model.fit(train_dataset)
inputs = keras.Input(shape=(1,), dtype="string")
x = text_vectorizer(inputs)
outputs = model(x)
end_to_end_model = keras.Model(inputs, outputs)
print("\nCalling end-to-end model on test string...")
test_data = tf.constant(["The one the other will absorb"])
```

```
test_output = end_to_end_model(test_data)
print("Model output:", test_output)
```

9.2 UI –DESIGN:

Index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Twitter (X) Sentiment Analysis</title>
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
    />
    <style>
      body {
        background-color: #f8f9fa;
      }
      .container {
        margin-top: 50px;
      }
      h1 {
        color: #007bff;
      }
      .form-control {
        background-color: #ffffff;
      }
      .btn-primary {
        background-color: #007bff;
        border-color: #007bff;
      }
      .btn-primary:hover {
```

```
      background-color: #0069d9;
      border-color: #0062cc;

    }
    .alert-success {
      background-color: #d4edda;
      color: #155724;
      border-color: #c3e6cb;
    }

    .alert-danger {
      background-color: #f8d7da;
      color: #721c24;
      border-color: #f5c6cb;
    }
    .alert-info {
      background-color: #cce5ff;
      color: #004085;
      border-color: #b8daff;
    }
  </style>
</head>

<body>
  <div class="container mt-5">
    <div class="row justify-content-center">
      <div class="col-md-8">

        <h1 class="mb-4 text-center">Twitter (X) Sentiment Analysis</h1>
        <form method="POST" action="{{ url_for('analyze_sentiment') }}">
          <div class="form-group">
            <label for="comment">Enter your comment:</label>
            <textarea
              class="form-control"
```

```
      id="comment"
      name="comment"
      rows="3"

      placeholder="Type your comment here..."
    ></textarea>
  </div>
  <div class="text-center">
    <button type="submit" class="btn btn-primary">
      Analyze Sentiment
    </button>
  </div>
</form>

{% if sentiment is defined %}
<div class="mt-4">
  {% if sentiment == 0 %}
  <div class="alert alert-danger text-center" role="alert">
    Negative comment!
  </div>
  {% elif sentiment == 1 %}
  <div class="alert alert-info text-center" role="alert">
    Positive comment!

  </div>
  {% else %}
  <div class="alert alert-success text-center" role="alert">
    Neutral comment!
  </div>
  {% endif %}

  <div class="mt-2 alert alert-secondary text-center" role="alert">
    Cluster: {{ cluster }}
  </div>
```

```
        </div>
        {% endif %}
       </div>
     </div>

   </div>

   <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
  </body>
</html>
```

app.py:

```
from flask import Flask, render_template, request
import pickle
import re
import nltk
import pandas as pd
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
import warnings
from sklearn.exceptions import InconsistentVersionWarning

nltk.download('stopwords')
warnings.filterwarnings("ignore", category=InconsistentVersionWarning)
warnings.filterwarnings("ignore", category=DeprecationWarning)
emoticon_pattern = re.compile(r'(?::|;|=)(?:-)?(?:\)|\(|D|P)')
emojis = [':)', ':D']
stopwords_set = set(stopwords.words('english'))
app = Flask(__name__)

with open('clf.pkl', 'rb') as f:
    clf = pickle.load(f)
```

```
with open('tfidf.pkl', 'rb') as f:
    tfidf = pickle.load(f)

def preprocessing(text):
    text = re.sub('<[^>]*>', '', text)

    emojis_found = emoticon_pattern.findall(text)
    text = re.sub(r'[\W+]', ' ', text.lower()) + ' '.join(emojis_found).replace('-', '')
    porter = PorterStemmer()
    text = [porter.stem(word) for word in text.split() if word not in stopwords_set]
    return " ".join(text)

data_path = r"C:\Users\DELL\Desktop\Sentiment-Analysis\Tweets.csv"

data = pd.read_csv(data_path, encoding='ISO-8859-1', header=None)

data.columns = ['textID','text','selected_text','sentiment']
print(data.columns)

data['text'] = data['text'].fillna('')
data['cleaned_text'] = data['text'].apply(preprocessing)

tfidf_vectorizer = TfidfVectorizer()
X = tfidf.transform(data['cleaned_text'])

num_clusters = 4
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
data['cluster'] = kmeans.fit_predict(X)

@app.route('/', methods=['GET', 'POST'])
def analyze_sentiment():
    if request.method == 'POST':
        comment = request.form.get('comment')
```

```
preprocessed_comment = preprocessing(comment)

        comment_vector = tfidf.transform([preprocessed_comment])

        prediction_prob = clf.predict_proba(comment_vector)[0]

        if prediction_prob[1] >= 0.7:
            sentiment = 1  # Positive sentiment
        elif prediction_prob[0] >= 0.7:
            sentiment = 0  # Negative sentiment
        else:

            sentiment = 2  # Neutral sentiment

        cluster = kmeans.predict(comment_vector)[0]

        return render_template('index.html', sentiment=sentiment, cluster=cluster)

    return render_template('index.html')


@app.route('/cluster', methods=['POST'])
def cluster_comments():
    comment = request.form.get('comment')

    preprocessed_comment = preprocessing(comment)

    comment_vector = tfidf_vectorizer.transform([preprocessed_comment])
    cluster = kmeans.predict(comment_vector)

    return render_template('index.html', cluster=cluster[0])

if __name__ == '__main__':
    app.run(debug=True)
```

CONCLUSION AND FUTURE ENHANCEMENT

10. CONCLUSION AND FUTURE ENHANCEMENTS

10.1 CONCLUSION:

Sentiment evaluation plays a important position in expertise public opinion, patron comments, and tendencies by using extracting valuable insights from massive volumes of textual statistics. Through the software of machine learning and natural language processing strategies, sentiment evaluation systems can mechanically classify critiques as wonderful, terrible, or impartial, providing groups and researchers with actionable intelligence. This undertaking demonstrates the effectiveness of using advanced algorithms, which includes neural networks and deep getting to know fashions, in enhancing the accuracy of sentiment class, specifically in complex and large-scale datasets.

The development method worried careful coding, rigorous testing, and systematic implementation, ensuring that the sentiment evaluation machine may want to take care of numerous input information, which include tweets from various domain names. By using libraries like TensorFlow, Keras, NLTK, and scikit-study, along with statistics manipulation equipment like Pandas and NumPy, the machine is able to preprocessing facts, extracting functions, and turning in accurate sentiment predictions. The integration of those libraries and frameworks significantly streamlines the procedure, from statistics cleansing to model assessment and output visualization.

Moreover, thorough trying out strategies had been carried out, such as unit testing, integration checking out, and reputation trying out, to ensure the reliability and accuracy of the system. These tests identified capacity errors, which includes facts fine problems and version performance, which were addressed to optimize the gadget for real-international use. This method guarantees that the machine can adapt to extraordinary user requirements, datasets, and contexts, providing a flexible and scalable solution for sentiment analysis tasks.

In conclusion, this project illustrates the potential of combining machine learning algorithms with sentiment analysis to derive meaningful insights from textual data. As technology and models continue to evolve, sentiment analysis systems can become more accurate and sophisticated, offering deeper understanding of human language and sentiment. The outcomes of this project highlight the value of sentiment analysis across various industries, from social media monitoring to customer service optimization, enabling data-driven decision-making.

10.2 FUTURE ENHANCEMENT:

- **Support for Multilingual Sentiment Analysis**: Expanding the machine to address a couple of languages beyond English, permitting it to analyze sentiment in international datasets, in particular in regions where social media posts and person feedback are in distinctive languages.
- **Incorporating Advanced Deep Learning Models:** Implementing cutting-edge deep studying fashions inclusive of BERT, GPT, or different transformer-based totally architectures to enhance the accuracy of sentiment category, especially in dealing with complex linguistic nuances like sarcasm or irony.
- **Contextual Sentiment Understanding:** Enhancing the machine to higher capture the context of conversations, allowing it to distinguish among sarcasm, irony, and impartial statements more accurately via leveraging context-aware fashions like XLNet.
- **Real-Time Sentiment Analysis**: Optimizing the machine to carry out sentiment evaluation in real-time for stay streams of facts (e.G., live tweets or customer service chats), providing immediately insights to organizations or customers.
- **Sentiment Trends and Topic Modeling**: Adding the ability to pick out and tune sentiment developments through the years, coupled with topic modeling features, permitting customers to hit upon emerging subjects and associated sentiments from massive datasets mechanically.
- **User Feedback Loop for Continuous Learning:** Implementing a remarks mechanism in which users can accurate sentiment predictions, allowing the model to examine and enhance constantly primarily based on real-world interactions and feedback.
- **Sentiment Analysis for Multi-modal Data:** Extending the gadget to address multi-modal facts, which includes combining textual content with photos, motion pictures, or audio, to analyze sentiment across unique content types, that is particularly useful in social media tracking and advertising analytics.

BIBLIOGRAPHY

11.BIBLIOGRAPHY

1. Books Referred:

- Cambria, Erik, et al. SenticNet 4: A Semantic Resource for Sentiment Analysis Based on Conceptual Primitives. Proceedings of COLING, 2016.
- Zhang, Xiang, et al. Character-level Convolutional Networks for Text Classification. Advances in Neural Information Processing Systems (NeurIPS), 2015.
- LeCun, Yann, et al. Deep Learning. Nature, 2015.
- Vader, C. J. Hutto, and Eric Gilbert. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Proceedings of the International AAAI Conference on Web and Social Media (ICWSM), 2014.
- Kumar, Arun, and Sebastian Reddy. Twitter Sentiment Analysis Using Machine Learning Techniques. International Journal of Engineering and Technology, 2019.
- Goldberg, Yoav. Neural Network Methods for Natural Language Processing. Morgan & Claypool Publishers, 2017.

2. Websites Referred:

- https://colab.research.google.com/
- https://en.wikipedia.org/wiki/Neural_network_(machine_learning)
- https://www.youtube.com/results?search_query=sentiment+analysis_using_neural_network
- https://www.tensorflow.org/guide/keras
- https://www.python.org/
- https://www.ibm.com/topics/machine-learning
- https://www.kaggle.com/datasets/kazanova/sentiment140
- https://github.com/ShivaprakashDM/500-AI-Machine-learning-Deep-learning-Computer-vision-NLP-Projects-with-code.git