International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Sentiment Analysis on E-Commerce Product Reviews Using Deep Learning

Ch.Mounika[1], K. Nageswara Rao[2]

ABSTRACT

With the rapid growth of e-commerce platforms, understanding customer feedback has become crucial for businesses to improve their products and services. Sentiment analysis, a subfield of Natural Language Processing (NLP), plays a vital role in extracting opinions from user-generated content such as product reviews. This project aims to perform sentiment analysis on e-commerce product reviews using a hybrid deep learning approach. A CNN-LSTM (Convolutional Neural Network - Long Short-Term Memory) model was implemented to classify customer reviews into positive and negative sentiments. The CNN layer effectively captures local features from the text, while the LSTM layer processes sequential dependencies for better contextual understanding. The dataset was preprocessed using techniques such as tokenization, stop word removal, and padding, followed by word embedding for semantic representation. Experimental results demonstrate that the CNN-LSTM model achieves an impressive accuracy of 95%, outperforming traditional machine learning models. The proposed model provides an efficient solution for automating sentiment classification in large-scale e-commerce platforms.

CHAPTER 1
INTRODUCTION
1.1 INTRODUCTION

These days, the surge in online shopping and digital platforms has resulted in an overwhelming amount of customer-generated content, particularly in the form of product reviews. These reviews hold significant power as they can influence purchasing decisions and shape public perception of a brand or product. However, manually analyzing thousands of reviews to extract meaningful insights is neither efficient nor scalable. This is where sentiment analysis plays a crucial role.

Sentiment analysis, also referred to as opinion mining, is the computational process of identifying and categorizing opinions expressed in a piece of text—especially to determine whether the writer's attitude toward a particular topic is positive, negative, or neutral. With the continuous expansion of e-commerce, businesses are increasingly relying on automated sentiment analysis tools to interpret customer feedback and improve user satisfaction.

Detecting sentiment accurately has become increasingly vital in today's e-commerce-driven landscape, where businesses compete on customer experience. Certainly, achieving high accuracy in sentiment classification is not an easy task. Human language is inherently complex and context-dependent, with elements like sarcasm, idioms, and varying expressions posing additional challenges. Furthermore, the sentiment conveyed in a review may not be explicit, requiring deep contextual understanding.

In this study, it is sought to develop a model that can accurately classify the sentiments expressed in e-commerce product reviews. Deep learning techniques have shown promising results in this domain due to their ability to learn hierarchical representations from raw text data. The proposed model utilizes a

hybrid architecture combining Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. CNN is effective in capturing local dependencies and identifying important n-gram features, while LSTM is capable of understanding the sequential nature and context of the reviews.

1.2 EVOLUTION OF SENTIMENT ANALYSIS

The ideology behind sentiment analysis has evolved alongside the growth of user-generated content in the digital ecosystem. Initially, sentiment analysis focused on identifying subjective expressions in text using rule-based and lexicon-based approaches. However, as online reviews, social media posts, and digital feedback became more complex and voluminous, the field of sentiment analysis adopted more advanced computational methods. Here's an overview of its evolution:

1.2.1 Lexicon-Based and Rule-Based Methods: In the early stages, sentiment analysis primarily relied on manually curated sentiment lexicons and rule-based systems. Words were labeled as positive, negative, or neutral, and the overall sentiment was computed by analyzing the frequency and intensity of these words. While straightforward, this approach lacked the ability to understand context, sarcasm, and complex sentence structures.

1.2.2 Traditional Machine Learning Techniques: The next phase saw the integration of supervised machine learning algorithms such as Naïve Bayes, Support Vector Machines (SVM), and Logistic Regression. These models were trained on labeled datasets and relied on hand-crafted features such as bag-of-words, TF-IDF scores, and n-grams. Although more adaptive than lexicon-based methods, these models still struggled with deeper semantic understanding.

1.2.3 Deep Learning and Word Embeddings: As the volume and complexity of textual data increased, deep learning techniques became prominent. Models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs) were introduced, which could automatically learn hierarchical and sequential features from raw text. Word embeddings like Word2Vec and GloVe significantly improved the semantic understanding of words, enabling more nuanced sentiment detection.

1.2.4 Contextualized Language Models: The development of Transformer-based architectures like BERT marked a significant milestone in sentiment analysis. These models provided contextualized word representations that allowed for deeper comprehension of syntax and semantics. Pre-trained models fine-tuned on sentiment-specific datasets drastically improved performance across various benchmarks.

1.2.5 Multilingual and Domain-Specific Sentiment Analysis: The latest advancements have focused on adapting sentiment analysis models for different languages and specific domains like e-commerce, healthcare, and finance. Domain adaptation techniques and transfer learning are being actively researched to enhance accuracy in context-specific scenarios, especially where labeled data is scarce.

1.2.6 Real-Time and Multimodal Sentiment Analysis: With the rise of real-time communication and multimedia content, sentiment analysis has expanded beyond plain text. Modern systems now incorporate real-time streaming data, as well as audio and visual cues, to determine sentiment. This evolution reflects the increasing need for systems that can process and interpret human emotions as they unfold across various digital mediums.

1.3 PURPOSE

In this project, we aim to classify sentiments expressed in e-commerce product reviews using a supervised deep learning approach. Sentiment classification is a pivotal task in understanding customer

satisfaction and product quality. Using labeled datasets containing positive and negative reviews, the system is trained to distinguish between them by learning the underlying features and patterns.

Before building the model, the raw textual data is preprocessed to remove noise and inconsistencies. This involves steps such as removing null values, tokenization, stemming, and stop-word removal. The processed text is then converted into numeric format using word embeddings to make it understandable for the machine learning model. The hybrid CNN-LSTM model is used to extract both spatial and sequential patterns from the text.

We utilize Python libraries such as Pandas, NumPy, Seaborn, Matplotlib, and TensorFlow/Keras to build and evaluate our model. These tools also help us generate visual insights from the data in the form of graphs and plots, aiding in better interpretation and understanding of sentiment trends.

1.4 OBJECTIVE

The main objective of this project is to develop an intelligent sentiment analysis system capable of accurately classifying product reviews as positive or negative. E-commerce platforms generate vast amounts of review data daily, making manual analysis impractical. By using deep learning techniques, we aim to automate this process and provide a scalable solution. Th The main objective of this project is to develop an intelligent sentiment analysis system capable of accurately classifying product reviews as positive or negative. E-commerce platforms generate vast amounts of review data daily, making manual analysis impractical. By using deep learning techniques, we aim to automate this process and provide a scalable solution.

This system seeks to address the challenge of extracting emotional tone from unstructured text data by leveraging a CNN-LSTM deep learning model. The project also aims to compare performance across different preprocessing techniques and evaluate how data volume impacts model accuracy. Our goal is to identify the most effective configuration for achieving high sentiment classification accuracy and to assist businesses in understanding customer perceptions more effectively.

1.5 OVERVIEW OF PROJECT

Earlier approaches to sentiment analysis relied heavily on static word lists and simple classifiers, which were not sufficient to understand nuanced human language. These models often failed to capture sentiment accurately in cases involving sarcasm, negation, or contextual variations. Additionally, the lack of semantic understanding limited their applicability in real-world scenarios.

In this project, we propose a hybrid deep learning model that integrates the strengths of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. CNN is used to identify key phrases and patterns, while LSTM is employed to learn dependencies between words over time. The combination allows the system to understand both the spatial and temporal characteristics of the review text.

The model is trained and tested on a labeled dataset of e-commerce product reviews. Using engineered features derived from text embeddings, the model successfully classifies sentiment with high accuracy—reaching a performance score of 95%. This approach does not rely on domain-specific behavioral data, making it more flexible and easier to deploy across different e-commerce platforms.

CHAPTER 2
LITERATURE REVIEW

2.1 THEORETICAL BACKGROUND OF THE PROBLEM

Sentiment analysis, also known as opinion mining, is a subfield of Natural Language Processing (NLP) that deals with identifying and categorizing opinions expressed in text. The theoretical foundation of sentiment analysis lies in the idea that emotions and opinions can be computationally extracted from human language. The study of sentiment analysis involves linguistic feature extraction, text classification, and deep contextual understanding of written content.

Consumer reviews on e-commerce platforms reflect the thoughts and satisfaction levels of users. These reviews are unstructured, vary in length and language complexity, and often include implicit sentiments, sarcasm, and domain-specific jargon. The challenge lies in automatically recognizing the polarity of such reviews to determine whether a review is positive, negative, or neutral.

Techniques such as deep learning and word embeddings have revolutionized sentiment analysis by allowing models to understand text contextually. Models like Convolutional Neural Networks (CNNs) are used to capture local patterns in the text, while Long Short-Term Memory (LSTM) networks are employed to preserve word order and contextual dependencies. The combination of CNN and LSTM, as used in our project, enables robust sentiment classification by leveraging both feature extraction and sequence modeling.

Additionally, sentiment analysis benefits from techniques such as tokenization, stop-word removal, and embedding layers like Word2Vec, GloVe, or FastText. These methods help transform raw textual input into numerical form, making it understandable to neural networks. Deep learning-based sentiment analysis can further be enhanced using attention mechanisms and transformer-based architectures like BERT.

The goal of sentiment analysis on product reviews is to assist customers in making informed purchasing decisions and to help businesses understand customer satisfaction. As the amount of review data continues to grow exponentially, automated systems using deep learning have become essential for large-scale, accurate opinion mining.

2.2 RELATED RESEARCH TO SOLVE THE PROBLEM

1. Sentiment Analysis on Product Review Using Support Vector Machine (SVM)

In their study, Chunduru Anilkumar et al. (2023) explored sentiment analysis on product reviews utilizing Support Vector Machine (SVM) and Naïve Bayes classifiers. The authors collected a dataset of product reviews and performed extensive preprocessing, including tokenization, stop-word removal, and stemming, to prepare the data for analysis. They implemented SVM and Naïve Bayes algorithms to classify the sentiments of the reviews into positive and negative categories. The study found that SVM outperformed Naïve Bayes in terms of accuracy, highlighting SVM's effectiveness in handling high-dimensional feature spaces typical in text classification tasks. The authors concluded that SVM is a robust classifier for sentiment analysis of product reviews and recommended its application in similar contexts.

2. Sentiment Analysis on Product Reviews Using Machine Learning Techniques

In their research, Amandeep Kaur et al. (2020) explored various machine learning algorithms for sentiment analysis on product reviews collected from e-commerce platforms like Flipkart and Amazon. The paper discussed the importance of feature extraction techniques such as TF-IDF and Bag of Words (BoW) to convert unstructured text data into numerical format suitable for machine learning models. The

authors implemented classifiers such as Logistic Regression, Random Forest, Decision Tree, and Support Vector Machine (SVM) for polarity detection of reviews into positive, negative, or neutral classes. Their experimental analysis revealed that SVM and Random Forest provided better accuracy compared to Logistic Regression, especially when coupled with TF-IDF features. The study also recommended handling class imbalance using techniques like SMOTE (Synthetic Minority Over-sampling Technique) for better performance.

3. Sentiment Analysis of Online Product Reviews Using Hybrid Approach

This study by K. Rajalakshmi et al. (2021) introduced a hybrid approach for sentiment analysis on online product reviews by combining machine learning classifiers with lexicon-based methods. The authors proposed an ensemble model integrating SVM, Random Forest, and Naive Bayes along with a sentiment lexicon to improve classification accuracy. The paper emphasized that hybrid models could overcome the limitations of individual classifiers by capturing both syntactic and semantic aspects of the review text. The researchers applied their methodology to datasets collected from Flipkart and Amazon, achieving improved accuracy and F1-score over traditional standalone models. The paper concluded that hybrid approaches are highly effective in real-time sentiment analysis applications for e-commerce platforms.

4. Opinion Mining and Sentiment Analysis of E-Commerce Product Reviews Using Machine Learning

In this paper, Divya G et al. (2020) investigated opinion mining and sentiment analysis for e-commerce product reviews using machine learning algorithms. The authors collected datasets from e-commerce platforms like Amazon, focusing on customer feedback on various products. The study applied text preprocessing techniques including stemming, lemmatization, and removal of noise data before feeding the reviews into classification models. The research utilized SVM, Naive Bayes, and Decision Tree algorithms for sentiment classification. The experimental results indicated that SVM outperformed other models due to its robustness in handling high-dimensional feature spaces. The study also suggested the application of deep learning models like LSTM in future work for further enhancement in accuracy.

5. Sentiment Analysis on Flipkart Product Reviews Using Machine Learning

In their research work, G. Sowmiya et al. (2021) analyzed Flipkart product reviews using machine learning algorithms for sentiment classification. The authors explored various preprocessing methods, including stop word removal, punctuation elimination, and word tokenization to prepare the data for analysis. They implemented machine learning classifiers such as Logistic Regression, Random Forest, and K-Nearest Neighbour (KNN) to classify the reviews into positive, negative, and neutral sentiments. The study highlighted that Random Forest produced the highest accuracy compared to other models due to its ensemble learning capability. The authors concluded that machine learning algorithms could effectively classify customer sentiments, helping businesses make better decisions based on user feedback.

6. Real-Time Twitter Sentiment Analysis for Product Reviews Using Naïve Bayes Classifier

Khushboo Gajbhiye and Neetesh Gupta (2019) conducted a study focusing on real-time sentiment analysis of product reviews sourced from Twitter, utilizing the Naïve Bayes classifier. The authors collected real-time tweets related to various products and performed preprocessing steps including tokenization, normalization, and removal of noise to prepare the data for analysis. They employed the Naïve Bayes algorithm to classify the sentiments expressed in the tweets as positive, negative, or neutral. The study highlighted the effectiveness of Naïve Bayes in handling the nuances of short and informal

text typical of tweets. The authors concluded that Naïve Bayes is a viable option for real-time sentiment analysis of product reviews on social media platforms.

7. Sentiment Analysis of E-commerce Product Reviews Using NLP Techniques

In their study, T. Manikandan et al. (2021) implemented Natural Language Processing (NLP) techniques for sentiment analysis on e-commerce product reviews. The authors used datasets collected from Amazon and Flipkart containing reviews on electronic gadgets and household items. They applied preprocessing steps like stemming, lemmatization, and part-of-speech tagging to clean the dataset. The study utilized machine learning classifiers like SVM, Naive Bayes, and Logistic Regression along with TF-IDF vectorization for feature extraction. The results indicated that SVM provided better classification accuracy, while the study also suggested that integrating deep learning techniques like LSTM could further improve the performance.

8. A Comparative Study of Machine Learning Techniques for Sentiment Analysis on Product Reviews

In this research, R. Priya et al. (2021) conducted a comparative study of various machine learning techniques for sentiment analysis on product reviews collected from Amazon. The study focused on evaluating the performance of classifiers like Logistic Regression, Random Forest, SVM, and Naive Bayes in terms of accuracy, precision, recall, and F1-score. The authors used Bag of Words and TF-IDF for feature extraction and applied preprocessing steps such as stopword removal, tokenization, and stemming. Their experimental analysis revealed that SVM achieved the highest accuracy, while Logistic Regression provided faster execution time. The paper concluded that the choice of classifier depends on the application-specific requirements such as accuracy or speed.

9. Sentiment Analysis on Customer Reviews of E-commerce Sites Using Deep Learning

In their research, R. Harika et al. (2023) presented a deep learning-based approach for sentiment analysis of customer reviews collected from e-commerce platforms. The study emphasized the use of LSTM and GRU (Gated Recurrent Unit) models for analyzing product reviews due to their capability of handling sequential data effectively. The authors used pre-trained word embeddings like Word2Vec to represent the reviews in vector format. The experimental results indicated that LSTM performed better than GRU and other traditional models in terms of accuracy and F1-score. The study concluded that deep learning techniques are highly suitable for real-time sentiment analysis on large-scale e-commerce review datasets.

10. A Machine Learning-Based Sentiment Analysis of Online Product Reviews with a Novel Term Weighting and Feature Selection Approach

Huiliang Zhao et al. (2021) proposed a machine learning-based approach for sentiment analysis of online product reviews, introducing a novel term weighting and feature selection method. The study involved collecting datasets from e-commerce platforms and applying preprocessing techniques such as tokenization, stop-word removal, and stemming. The authors introduced a unique term weighting scheme combined with feature selection to enhance the performance of traditional classifiers like SVM and Naïve Bayes. The experimental results demonstrated that the proposed approach improved classification accuracy and reduced computational complexity. The study concluded that integrating novel term weighting and feature selection methods can significantly enhance the effectiveness of machine learning algorithms in sentiment analysis tasks.

CHAPTER 3
SYSTEM ANALYSIS
3.1 EXISTING SYSTEM

The existing system used for sentiment analysis is based on an Improved Transformer Model combined with LSTM and Conditional Random Fields (CRF). First, input text is converted into word embeddings using Word2Vec with position encoding, then passed through an enhanced Transformer encoder that captures semantic relationships. The Transformer model utilizes a self-attention mechanism to capture contextual relationships and dependencies between words in a sentence. This helps in extracting meaningful features from the input text for accurate sentiment classification.

In the existing system, the Transformer model encodes the input sequence and captures global dependencies using multi-head attention. The output from the Transformer encoder is passed through a Conditional Random Field (CRF) layer, which captures label dependencies and improves the sequence labeling performance.

The system is mainly evaluated on movie reviews datasets like IMDB. It uses improved position encoding, self-attention, and CRF-based sequence modeling for sentiment classification tasks.

This model improves the accuracy of sentiment classification in datasets like IMDB by capturing long-term dependencies and contextual relationships between words effectively.

3.1.1 LIMITATIONS OF EXISTED SYSTEM

- The combination of Transformer, CRF and LSTM models increases the computational complexity of the system which may lead to overfitting, makes it unsuitable for real-time sentiment analysis applications due to high training and inference times.
- Variations in emotion classification among different annotators can lead to inconsistent labeling of same text, therefore impacting the accuracy of model training and evaluation results.
- The model lacks interpretability and explainability, as it does not provide clear reasoning for the predicted sentiment class, which is essential for real-world applications.
- The system requires large amounts of data and high computational resources (GPU/TPU) for training, which may not be feasible in all scenarios.

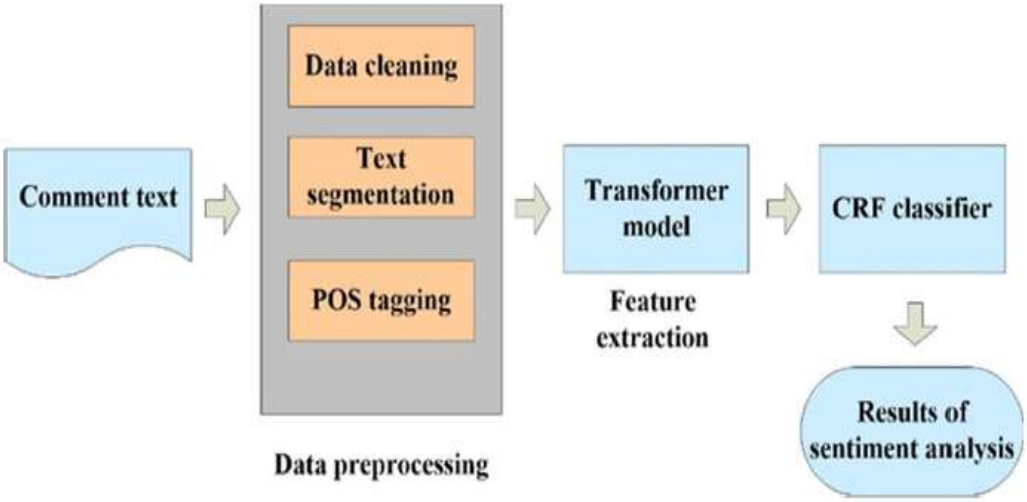


Fig 3.1.2: Architecture of existing system

3.2 PROPOSED SYSTEM

The proposed system focuses on performing sentiment analysis on e-commerce product reviews collected from platforms like Flipkart. The primary objective of the system is to classify customer reviews into two sentiment categories — Positive and Negative — based on the content of the review.

The proposed system utilizes a deep learning-based approach by implementing a hybrid CNN-LSTM (Convolutional Neural Network - Long Short-Term Memory) model. The CNN layer is used to extract local features and patterns from the input text data, while the LSTM layer captures the sequential and contextual dependencies present in the reviews.

To represent the textual data in numerical format, pre-trained GloVe (Global Vectors for Word Representation) embeddings are used. GloVe embeddings help in capturing semantic meaning and relationships between words, which enhances the performance of the model.

The system involves key steps such as data collection, data cleaning, data preprocessing (removing special characters, stop words, tokenization), text vectorization using GloVe, model building using CNN-LSTM, model training, evaluation using accuracy and other performance metrics, and finally, prediction of sentiment polarity**.**

3.2.1 ADVANTAGES OF PROPOSED SYSTEM

- The CNN-LSTM model automatically extracts relevant features from the raw review text, eliminating the need for manual feature engineering.
- The hybrid CNN-LSTM architecture effectively captures both local features and long-term dependencies in the text, making it suitable for analyzing long and complex reviews.
- The use of GloVe word embeddings enhances the semantic representation of words, improving the accuracy of sentiment classification.
- The system helps customers in quick decision-making by summarizing the overall sentiment of product reviews.
- The model is highly scalable and can be applied to different domains of text analysis beyond e-commerce, such as movie reviews, social media analysis, and feedback systems.
- The proposed system can handle large-scale review datasets and provide accurate sentiment classification results.

3.3 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Some of the key considerations involved in the feasibility analysis are:

- Technical Feasibility
- Social Feasibility
- Operational Feasibility
- Economical Feasibility

3.3.1 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being

placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.3.2 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.3.3 OPERATIONAL FEASIBILITY

The project is operationally feasible as the user having basic knowledge about computer and Internet. It is based on client-server architecture where client is users and server are the machine where datasets are stored.

3.3.4 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

CHAPTER 4
SYSTEM REQUIREMENT SPECIFICATION

4.1 MINIMUM HARDWARE REQUIREMENTS

- Processor : Intel Core i3/i5 or equivalent
- Ram : 8 GB.
- Storage : Minimum 256 GB SSD/HDD
- Internet Connection : Required for Colab, GloVe download, pip installs
- Monitor : 15-inch LED/LCD display

4.2 MINIMUM SOFTWARE REQUIREMENTS

- Operating System : Windows 10/Linux/macOS
- Platform : Google Colab
- Tool : Python 3.8 or higher
- Libraries : Tensorflow, Keras, NumPy, Pandas, Matplotlib, Seaborn
- Browser : Google Chrome/Firefox (For Colab access)

4.3 FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Generally, functional requirements are expressed in the form “system shall do”. The plan for implementing functional requirements is detailed in the system design. In requirements engineering, functional requirements specify particular results of a system. Functional requirements drive the

application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements is:

user/stakeholder request -> feature -> use case -> business rule

- Functional requirements drive the application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements.
- Functional requirements may be technical details, data manipulation and other specific functionality of the project to provide the information to the user.

The following are Functional requirements of our system:

1. Ability to collect a diverse and large dataset of product reviews from the Flipkart e-commerce platform.
2. Pre-processing steps such as text cleaning, tokenization, stop word removal and vectorization to convert raw text data into a suitable format for deep learning models.
3. Conversion of processed textual data into word embeddings using GloVe to capture the semantic meaning of words.
4. Implementation of a CNN-LSTM model to analyze the sequential nature of the text data and classify the sentiments of product reviews as Positive or Negative.
5. Evaluation of the proposed model using performance metrics such as Accuracy, Precision, Recall, and F1-score to assess the effectiveness of sentiment classification.
6. Visualization of results through graphs and charts to help users better understand the sentiment distribution and model performance.
7. Provision of user-friendly interface for entering product reviews and displaying the sentiment classification results effectively.

4.4 NON-FUNCTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. The plan for implementing non-functional requirements is detailed in the system architecture.

- The non-functional requirements are "system shall be <requirement>".
- Non-functional requirements are often called qualities of a system.
- Non-functional requirements describe how the system performs, not what it does.

Accuracy: The system should achieve high accuracy in distinguishing between positive and negative reviews to minimize false positives and false negatives.

Reliability: The system should be reliable and consistent in its performance, producing accurate results even under varying conditions and with different datasets.

Scalability: The system should be scalable to handle increasing volumes of data and user requests without significant degradation in performance.

Performance: The system should exhibit low latency and high throughput, providing real-time or near-real-time prediction capabilities to analyze sentiments quickly.

Security: The system should adhere to security best practices to protect against unauthorized access, data breaches, and malicious attacks that could compromise its integrity or confidentiality.

Interpretability: The system should provide explanations or insights into its decision-making process, enabling users to understand why certain reviews are classified as positive or negative.

Adaptability: The system should be adaptable to evolving trends and tactics used by customers one-commerce platforms, incorporating new features or updating models as needed.
Maintainability: The system should be easy to maintain and support, with clear documentation, modular design, and well-defined interfaces to facilitate updates, bug fixes, and enhancements.
Usability: The system should be user-friendly and intuitive, with a clear interface and helpful feedback mechanisms to guide users through the sentiment analysis process.

CHAPTER 5
METHODOLOGY
5.1 DATA COLLECTION
The dataset for sentiment analysis on e-commerce product reviews was sourced from Kaggle and consists of a diverse collection of product reviews from the Flipkart platform. The dataset contains reviews provided by customers about various products along with corresponding ratings and review sentiments. The dataset includes attributes such as the product name, product price, rating, customer review text, summary of the review and the sentiment label categorized as "Positive," "Negative," or "Neutral." This dataset serves as a valuable resource for developing and training machine learning and deep learning models for sentiment analysis tasks. However, the dataset also has certain limitations like the presence of irrelevant reviews, duplicate reviews, and class imbalance issues. Overall, the dataset comprises 2,05,052 customer product reviews, enabling comprehensive training and evaluation of sentiment analysis models using deep learning techniques. The Flipkart product reviews dataset from Kaggle supports research and development efforts aimed at understanding customer feedback, improving product quality, and enhancing customer satisfaction in the e-commerce industry.
5.2 PROPOSED SYSTEM
The proposed system uses a hybrid deep learning model that combines Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) networks for sentiment analysis of e-commerce product reviews. The CNN is used to extract local features and important patterns from the text data, while the LSTM is used to capture the sequential dependencies and contextual information present in the review text. This hybrid CNN-LSTM model is highly efficient for handling large-scale unstructured text data and provides superior performance in terms of accuracy and reliability. The model is trained to classify product reviews into two categories: Positive and Negative sentiments.
The CNN-LSTM model works based on the following principles:

- CNN Layer: Extracts key features from text data like n-grams, phrases, and patterns.
- LSTM Layer: Captures the sequence of words and contextual dependencies across the review text.
- Dense Layer: Performs the final classification of the review into the respective sentiment category. The model uses pre-trained GloVe embeddings to convert the review text into word vectors that capture semantic meaning. Finally, performance evaluation is carried out using standard metrics.

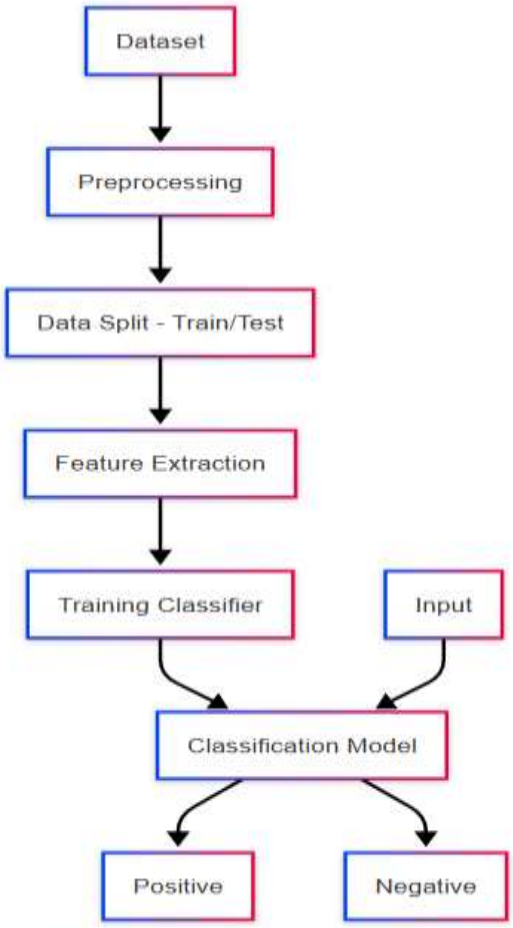


Fig 5.2.1: Architecture of Proposed System

DATA CLEANING

Data collected from the Flipkart platform is often unstructured and contains noise in the form of spelling mistakes, special characters, stop words, emojis, and other irrelevant data. To improve the accuracy and reliability of the sentiment analysis model, the data must be cleaned before being fed to the deep learning model. Data Cleaning process includes dropping missing values and duplicates, removing invalid values from the dataset. This cleaning process ensures that only relevant information is retained in the review text for further processing.

In the proposed system, neutral sentiments are removed from the dataset because that data is very less than positive and negative sentiments. It ensures to focus on binary sentiment classification. Since the dataset taken is imbalanced, it is balanced by using the process of oversampling in which the number of samples in minority class are increased up to the number of majority class samples. This helps the model learn fairly from both positive and negative samples.

DATA PREPROCESSING

The data may be in the format of either structured or unstructured. A structured format has patterns that are well defined and the unstructured data do not have a proper structure. Among structured and semi-structured formats, comparison of the better structured and then unstructured way. Text and the data have to be cleaned to highlight the attributes which we want our deep learning system to work on accuracy.

The data preprocessing includes- converting text to lowercase, removal of special characters, punctuation, stop words removal and tokenization.
Once cleaned and tokenized, the data is split into two parts- one for training the model and one for testing how well it performs.

FEATURE GENERATION

In the proposed system, feature generation is done using text vectorization techniques to convert the textual data into numerical form that can be processed by the deep learning model. The following methods are used:

- Word Embedding: GloVe pre-trained embeddings are used to convert words into dense vectors that capture semantic relationships between words.
- Tokenization: Splitting the review text into individual words or tokens.
- Padding: Ensuring all input sequences have the same length by padding shorter sequences with zeros.

This representation enables the CNN-LSTM model to understand the contextual and semantic meaning of the product reviews.

TRAINING THE MODEL

The CNN-LSTM model is trained on the preprocessed dataset after converting the text data into numerical vectors using GloVe embeddings. The model learns the complex patterns and relationships present in the customer reviews to classify them into appropriate sentiment categories. The dataset is divided into training and testing sets for effective learning and evaluation. The final trained model is stored and can be used for predicting the sentiment of new customer product reviews provided by the user.

MODEL EVALUATION

After training the model, its performance is evaluated using various performance metrics. The confusion matrix is used to visualize the results of the classification and evaluate the effectiveness of the model. Most of the approaches will consider this as a problem of classification that estimates if the review is positive or negative:

Confusion Matrix Terms:

- True Positive (TP): Correctly predicted positive reviews.
- True Negative (TN): Correctly predicted negative reviews.
- False Positive (FP): Incorrectly predicted positive reviews.
- False Negative (FN): Incorrectly predicted negative reviews.

Confusion matrix:

A table that outlines the performance of a classifier on a bunch of test information for which the genuine values are known. It is used to visualize algorithm performance.

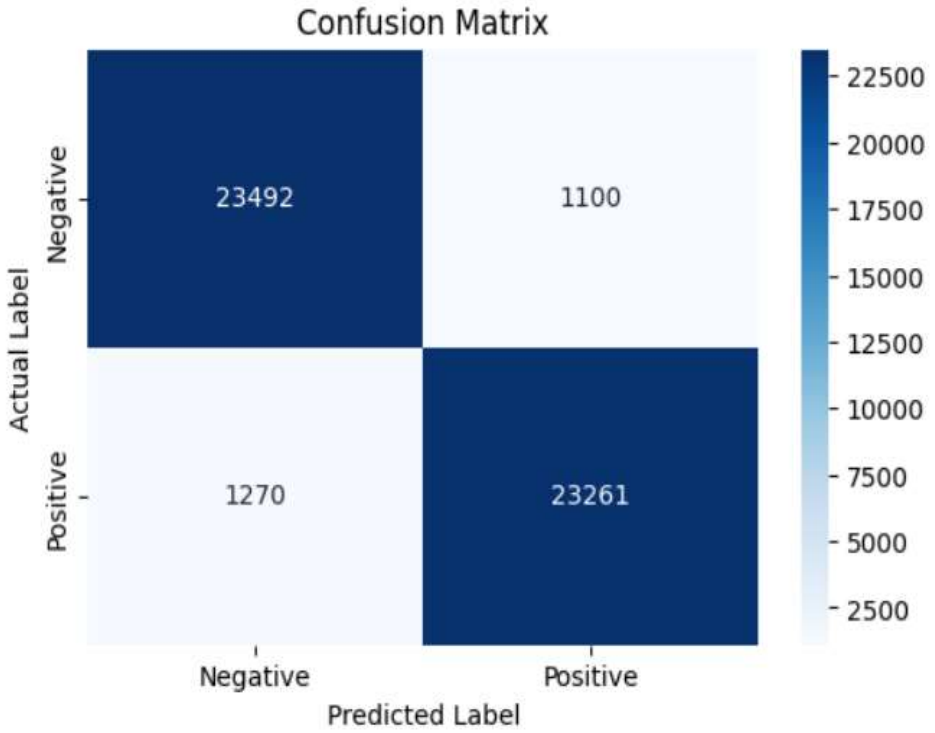


Fig 5.2.2: Sample of confusion matrix for CNN-LSTM Model

CHAPTER 6
DESIGN

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirements have been specified and analyzed, system design is the first of the three technical activities - design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

6.1 INPUT AND OUTPUT DESIGN

6.1.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in

to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system.

The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

6.1.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the systems relationship to help user decision-making.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

6.2 UML DIAGRAMS

A diagram is a graphical presentation of a set of elements, most often is rendered as a graph of vertices and arcs. A UML diagram can contain nine types in it where vertices are denoted as things and arcs are denoted as relationships.

The Unified Modeling Language diagram is that is, the unified modeling language is probably the most widely known and used notation for object-oriented analysis and design. The Unified Modeling Language is used to visualize, specify, construct and document the artifacts. A Modeling language is a language whose vocabulary and rules focus on the conceptual and physical representation of a system. Modelling is the designing of software applications before coding. According to UML no one diagram can capture the different elements of a system in its entirety. Hence UML is made up of nine diagrams that can be used to model a system at different points of time in the software life cycle of a system in its entirety. The eight diagrams include:

Use Case Diagram: This diagram shows how users interact with the system. The main actor is the user who performs tasks like loading data, preprocessing reviews, training the model, evaluating performance, predicting sentiments, and visualizing results.

Activity Diagram: An activity diagram demonstrates the sequential workflow of the sentiment analysis system. It shows how the system progresses from the loading and cleaning of the dataset, through text preprocessing and model training, to model evaluation and real-time sentiment prediction.

Class Diagram: This diagram illustrates the major classes and modules used in the project, along with their attributes and operations. The main classes include Dataset Handler, Text Preprocessor, Tokenizer Manager, Model Builder, Inference Engine and Visualizer.

Sequence Diagram: Sequence diagrams show the chronological flow of control between objects during specific interactions. It shows how the user input flows through data preprocessing, tokenization, model loading or training, and finally results in a sentiment prediction.

State Diagram: The state machine diagram models the lifecycle states of a user input (review) in the system. A review transitions from a raw input state to a preprocessed state, then to tokenized and padded, followed by prediction, and finally classified as either Positive or Negative. This helps track how input data evolves through the different stages of sentiment analysis.

Component Diagram: Component diagrams depict the logical components of the sentiment analysis system and their dependencies. This outlines the system's components like the dataset module, preprocessing engine, model trainer, evaluation module, and prediction engine. It highlights their interconnections and dependencies.

Deployment Diagram: Deployment diagrams illustrate the physical deployment of software components on hardware nodes. It shows how the system is deployed in Google Colab, using Google Drive to access datasets, GloVe embeddings, and save/load models. All components run in the cloud environment for real-time predictions.

Uml Class diagram

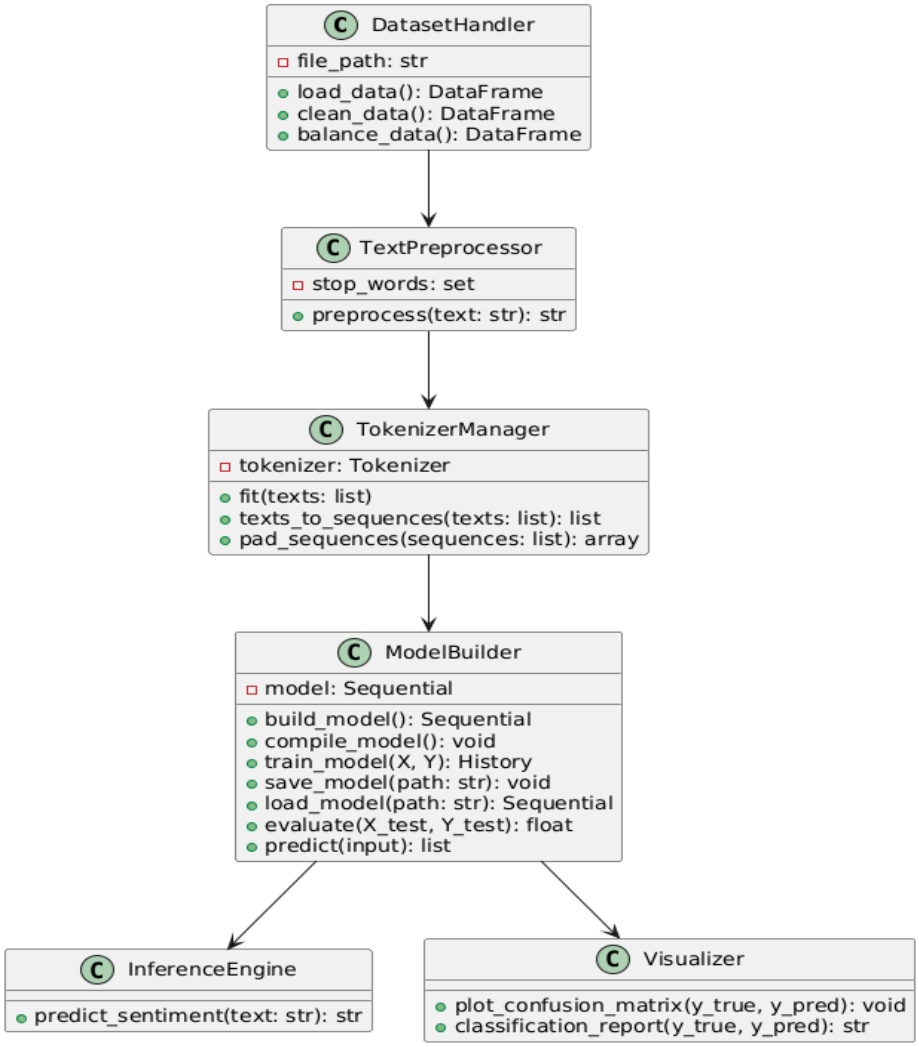


Fig 6.2.1: Uml class diagram for Proposed System

Explanation:

Dataset Handler class: It loads, cleans and balances the dataset.

Text Preprocessor class: It preprocesses the text (converts to lowercase, remove stopwords, tokenize).

Tokenizer Manager class; It converts text into sequences and applies padding.

Model Builder class: It builds, compiles, trains, saves and loads the CNN-LSTM model.

Visualizer class: It displays confusion matrix and classification report for evaluation.

Inference Engine class: It takes user input text and predicts the sentiment (positive/negative) using trained model.

Sequence Uml diagram

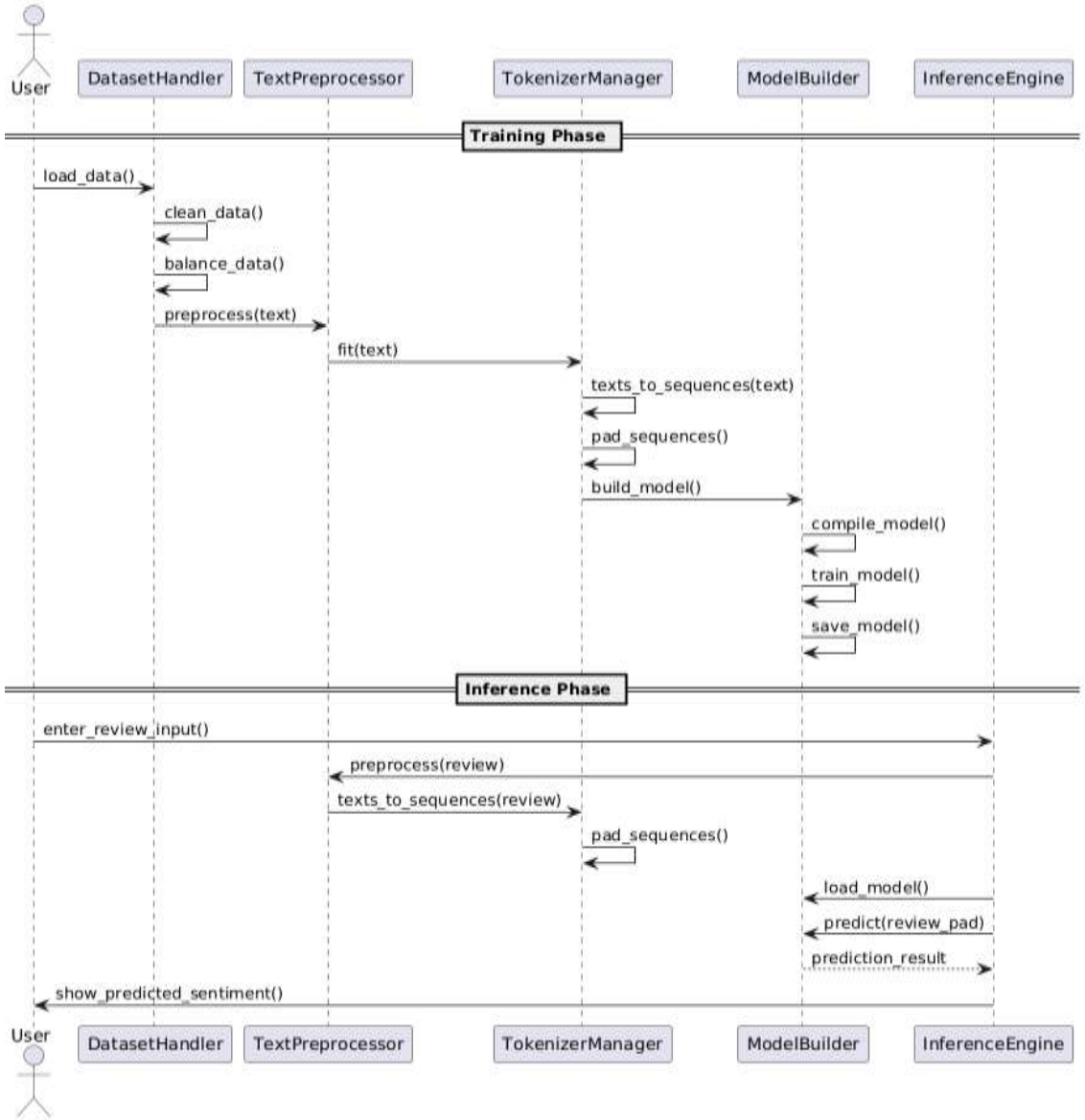


Fig 6.2.2 Sequence Uml diagram for Proposed System

CHAPTER 7
IMPLEMENTATION

7.1 IMPLEMENTATION

This chapter describes the practical implementation of the proposed sentiment analysis system, which is built using Python and state-of-the-art deep learning techniques. Python is a versatile and powerful programming language that has become a cornerstone in the field of data science and machine learning due to its simplicity, readability, and an extensive ecosystem of libraries. With its dynamic nature and strong support for numerical computation and text processing, Python serves as an ideal choice for developing machine learning models and natural language processing (NLP) applications.

The implementation of this project leverages multiple Python libraries such as Pandas for data handling, NumPy for numerical computation, NLTK for natural language preprocessing, Seaborn and Matplotlib for visualization, and TensorFlow/Keras for building and training deep learning models. The dataset used contains customer reviews and summary texts from an e-commerce platform, which are first cleaned, preprocessed, and balanced to improve training quality.

A hybrid deep learning model combining Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) is implemented to extract both local features and long-term dependencies from textual data. To enrich semantic understanding, the model uses pre-trained GloVe embeddings, which help convert text into meaningful numerical representations. The model also incorporates dropout and regularization techniques to prevent overfitting and improve generalization.

Google Colab is used as the execution environment, and Google Drive is integrated for dataset access and model persistence. The model is trained using techniques such as early stopping and learning rate reduction to ensure efficient and effective convergence. Finally, the system is equipped with an inference module that allows users to input custom product reviews and receive real-time sentiment predictions.

Overall, the implementation is designed to be modular, scalable, and robust, providing accurate sentiment classification with powerful deep learning techniques.

7.2 SAMPLE CODE:

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.model_selection import train_test_split
from sklearn.utils import resample
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, MaxPooling1D, Bidirectional, LSTM, Dropout, Dense
```

```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.regularizers import l2
from sklearn.metrics import confusion_matrix, classification_report
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')
# Load dataset
df = pd.read_csv("/content/drive/MyDrive/Dataset-SA.csv")
df.head()
df.info()
# Data Cleaning
# Drop duplicates and missing values (if any))
df=df.dropna().drop_duplicates()
print(df['Rate'].value_counts().sort_index())
# Clean invalid data in 'Rate'
invalid_rates = [
'Pigeon Favourite Electric Kettle??????(1.5 L, Silver, Black)',
'Bajaj DX 2 L/W Dry Iron',
'Nova Plus Amaze NI 10 1100 W Dry Iron?ÃƒÂ¿?ÃƒÂ¿(Grey & Turquoise)'
]
df = df[~df['Rate'].isin(invalid_rates)]
print(df['Rate'].value_counts().sort_index())
#Changing the datatype of the column
df = df[df['Rate'].str.isnumeric()]
df['Rate'] = pd.to_numeric(df['Rate'])
# Count the occurrences of each sentiment
sentiment_counts = df['Sentiment'].value_counts()
print("Sentiment Counts:")
print(sentiment_counts)
# Remove neutral sentiments
df = df[df['Sentiment'] != 'neutral'] # Data is too small
print(df['Sentiment'].value_counts())
# Balance dataset
# Oversampling (Random Resampling of Minority Class)
data_majority = df[df['Sentiment'] == 'positive']
data_minority = df[df['Sentiment'] == 'negative']
print("majority class before upsample:",data_majority.shape)
print("minority class before upsample:",data_minority.shape)
# Upsample minority class
data_minority_upsampled = resample(data_minority,
replace=True, # sample with replacement
```

```
n_samples=data_majority.shape[0], # to match majority class
random_state=42)
# Combine majority class with upsampled minority class
df_balanced = pd.concat([data_majority, data_minority_upsampled]).sample(frac=1, random_state=42)
# Display new class counts
print("After upsampling\n",df_balanced.Sentiment.value_counts(),sep = "")
# Downloading Necessary packages
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')
nltk.download('wordnet')
# Text Preprocessing
stop_words = set(stopwords.words('english')) # Set stopwords
def preprocess(text):
text = text.lower() # Convert to lowercase
text = re.sub(r'[^a-zA-Z\s]', '', text) # Remove all non-alphabetic characters
tokens = word_tokenize(text) # Tokenize text
filtered = [w for w in tokens if w not in stop_words] # Remove stopwords
return ' '.join(filtered) # Join tokens back into a single string
#Select Features and Target
#Concatenating 2 review columns and apply preprocessing
X = (df_balanced['Review'] + ' ' + df_balanced['Summary']).apply(preprocess)
print(X[0])
#Label Encoding
Y = np.array(df_balanced['Sentiment'].apply(lambda x: 1 if x == 'positive' else 0))
print(Y)
# Train-Test Split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
# Tokenization & Padding
# Preparing embedding layer
# Embedding layer expects the words to be in numeric form
# Using Tokenizer function from keras.preprocessing.text library
# Method fit_on_text trains the tokenizer
tokenizer = Tokenizer(num_words=10000)
tokenizer.fit_on_texts(X_train)
# Method texts_to_sequences converts sentences to their numeric form
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)
# Padding sequences
X_train_pad = pad_sequences(X_train_seq, maxlen=100, padding='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=100, padding='post')
print(X_train_pad.shape)
print(X_test_pad.shape)
```

```
# Embeddings Dictionary
# Load GloVe Embeddings and create an Embeddings Dictionary
from numpy import asarray
embeddings_index = {}
with open('/content/drive/MyDrive/glove.6B.50d.txt', encoding="utf8") as f:
for line in f:
values = line.split()
word = values[0]
coefs = np.asarray(values[1:], dtype='float32')
embeddings_index[word] = coefs
vocab_size = min(10000, len(tokenizer.word_index) + 1)
# Create Embedding Matrix
embedding_matrix = np.zeros((vocab_size, 50))
for word, i in tokenizer.word_index.items():
if i < vocab_size:
vec = embeddings_index.get(word)
if vec is not None:
embedding_matrix[i] = vec
# Check if model is already saved
model_path = "/content/drive/MyDrive/cnn_lstm_fast_model.keras"
if os.path.exists(model_path):
model = tf.keras.models.load_model(model_path)
print("Loaded pre-trained model.")
else: # CNN-LSTM Model Architecture
model = Sequential([
Embedding(vocab_size, 50, weights=[embedding_matrix], trainable=False),
Conv1D(64, 3, activation='relu'),
MaxPooling1D(2),
Bidirectional(LSTM(64)),
Dropout(0.5),
Dense(8, activation='tanh'),
Dense(1, activation='sigmoid', kernel_regularizer=l2(0.001))
])
model.compile(optimizer=Adam(0.0001),loss='binary_crossentropy', metrics=['accuracy']) # Compile Model
early_stop=EarlyStopping(monitor='val_loss',patience=5, restore_best_weights=True) # Define early stopping
reduce_lr = ReduceLROnPlateau(monitor='val_loss', patience=3, factor=0.3, verbose=1) # Create a ReduceLROnPlateau callback
# Train model
history = model.fit(
X_train_pad, Y_train,
batch_size=64,
```

```
epochs=5,
validation_data=(X_test_pad, Y_test),
callbacks=[early_stop, reduce_lr],
verbose=1
)
# Save model
model.save(model_path)
print("Model trained and saved.")
# Evaluate Model
loss, acc = model.evaluate(X_test_pad, Y_test)
print(f"Test Accuracy: {acc:.2f}")
# Predict & Evaluate
y_pred = model.predict(X_test_pad)
print(y_pred)
y_pred_class = (np.array(y_pred) >= 0.5).astype(int).ravel()
print(y_pred_class)
result=pd.DataFrame({'Actual output': Y_test,'Predict Output': y_pred_class})
print(result)
# Plot confusion matrix
cm = confusion_matrix(Y_test, y_pred_class)
labels = list(['Negative', 'Positive']) # Define labels based on encoded values
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.title('Confusion Matrix')
plt.show()
#Classification report
print("Classification Report:")
print(classification_report(Y_test, y_pred_class))
# Inference
user_review = input("Enter a product review: ") # Take review as input from user
pre_review = preprocess(user_review) #Preprocess
review_seq = tokenizer.texts_to_sequences([pre_review]) # Convert into sequences
review_pad = pad_sequences(review_seq, maxlen=100, padding='post') # pad sequences
pred = model.predict(review_pad)[0][0] # Predict sentiment
print("Predicted Sentiment:", "Positive" if pred >= 0.5 else "Negative") #Display predicted sentiment
```

CHAPTER 8
TESTING
8.1 TESTING INTRODUCTION

Software testing is the process of finding defects in the software so that these can be debugged and the defect-free software can meet the customer needs and expectations. Software testing is one of the

important phases in software development lifecycle. A quality software can be achieved through testing. Software testing is a crucial phase in the development of any machine learning-based system to ensure that it meets the required specifications and delivers accurate results. In this project, testing is performed to verify that the sentiment analysis model behaves as expected, classifies sentiments correctly, handles various types of input, and performs consistently. Effective testing increases the reliability of the model, reduces future maintenance, and ensures the delivery of a user-ready solution.

8.2 TESTING PROCESS

The testing process involves systematic activities to validate different stages of the sentiment analysis pipeline. A test plan was prepared that included testing of preprocessing modules, model performance, prediction accuracy, and user interaction. Test cases were designed to check each functional block — from data input to final output. Testing was performed both during development and after integration to ensure a robust and accurate system. Testcases include the input, output, and the conditions. Software tester runs the program using test cases and observes results.

8.3 LEVELS OF TESTING

Testing is a defect detection technique that is performed at various levels. Testing begins once a module is fully constructed. Although software engineers test source codes after it is written, but it is not an appealing way that can satisfy customer's needs and expectations.

Software is developed through a series of activities, i.e., customer needs, specification, design, and coding. Each of these activities has different aims. Therefore, testing is performed at various levels of development phases to achieve their purpose.

8.3.1 UNIT TESTING

Unit testing was performed on individual functions such as text cleaning, tokenization, label encoding, and padding. Each function was tested with various inputs to ensure correct behavior and output. Both black-box and white-box testing techniques were applied to verify functionality and internal logic.

8.3.2 INTEGRATION TESTING

Integration testing is performed after unit testing to ensure that individual modules of the sentiment analysis system work correctly when combined. In this project, modules such as data preprocessing, tokenization, word embedding, CNN-LSTM model, and prediction output were integrated step by step to verify the flow of data and correctness of intermediate outputs. The bottom-up approach was followed, starting from the lower-level modules such as preprocessing and embedding, then integrating with the deep learning model and finally linking to the user interface for prediction. This level of testing focused on identifying interface mismatches, data inconsistencies, or integration errors between components. Any issues arising from module dependencies or unexpected input-output behavior were identified and resolved during this phase.

8.3.3 SYSTEM TESTING

The unit and integration testing are applied to detect defects in the modules and the system as a whole. Once all the modules have been tested, system testing is performed to check whether the system satisfies the requirements (both functional and nonfunctional). In this project, system testing ensured that the model correctly loads the dataset, preprocesses the reviews, performs sentiment prediction, and presents results accurately to the user. Both positive and negative test scenarios were executed, including edge cases such as empty reviews or inputs with special characters. Additionally, non-functional aspects like

response time, model accuracy, usability, and reliability were validated. This comprehensive testing confirmed that the sentiment analysis system is robust, user-friendly, and ready for deployment.

Some of the non-functional system tests that are being used to test various features are:

- Performance testing
- Volume testing
- Security testing
- Recovery testing
- Compatibility testing
- Configuration testing
- Installation testing
- Documentation testing

8.3.4 ACCEPTANCE TESTING

Acceptance testing is a kind of system testing, which is performed before the system is released into the market. It is performed with the customer to ensure that the system is acceptable for delivery. Acceptance testing was conducted from the end-user's perspective by providing real-world product reviews through the user interface. The sentiment prediction results were observed for correctness and consistency. The system met the desired expectations in terms of ease of use and result accuracy.

8.3.5 BLACK-BOX TESTING

Black box testing focuses on evaluating the functionality of the system without considering the internal structure or code logic. In this project, black box testing was applied to verify the sentiment analysis system based solely on the inputs and outputs. For example, different types of product reviews (positive, negative, empty, or ambiguous) were entered into the system to test whether it produced the correct sentiment classification. The internal implementation of the CNN-LSTM model or preprocessing logic was not examined during this process. The goal was to ensure that the system behaves as expected from an end-user's perspective, identifying functional errors such as incorrect predictions or handling of invalid inputs.

8.3.6 WHITE-BOX TESTING

White box testing involves testing the internal logic, structure, and flow of the code. In this project, white box testing was applied to modules such as text preprocessing, tokenization, label encoding, and the data flow between layers of the neural network. Each function was reviewed and tested to ensure that all possible branches and paths were executed correctly, especially for handling special conditions like empty input, stopword removal, and unknown tokens. This helped identify logical errors in the code and confirm that each module processed the data as intended before passing it to the next component in the pipeline.

8.4 TEST CASE

We now know, test cases are integral part of testing. So, we need to know more about test cases and how these test cases are designed. The most desired or obvious expectation from a test case is that it should be able to find most errors with the least amount of time and effort.

CHAPTER 9
RESULT AND DISCUSSION
9.1 EVALUATION METRICS
ACCURACY SCORE:

- It measures the proportion of correctly predicted instances out of the total instances in the dataset.
- The accuracy score is calculated using the following formula: Accuracy = Number of Correct Predictions/Total Number of predictions.
- The accuracy score is expressed as a percentage and represents the overall correctness of the model's predictions.
- A higher accuracy score indicates better performance, as it means that a large proportion of the predictions made by the model are correct.

$$Accuracy = \frac{(TP + TN)}{(\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN})}$$

Where,

TP = True Positive

In a confusion matrix, a true positive (TP) denotes instances where the model correctly predicts the positive class, indicating accurate identification of the target outcome. Specifically, in the context of sentiment analysis, TP represents the number of product reviews that are actually positive and have been correctly classified as positive by the model. For instance, a TP would occur when the model accurately identifies a customer review like “This product is amazing and exceeded my expectations” as expressing positive sentiment. Understanding true positives helps evaluate the model’s ability to correctly recognize positive feedback, thereby contributing to the overall assessment of its precision in capturing customer satisfaction.

TN = True Negative

In a confusion matrix, a true negative (TN) signifies instances where the model correctly predicts the absence of the positive class, reflecting precise identification of the absence of the target outcome. Specifically, in binary sentiment classification, TN represents the count of product reviews that are genuinely negative and have been correctly classified as such. For example, a TN occurs when the model accurately identifies a review like “The quality was poor and not worth the price” as expressing negative sentiment. The true negative value provides insight into the model’s capability to detect dissatisfaction and helps assess its reliability in filtering unfavorable reviews from the positive ones.

FP = False Positive

In a confusion matrix, a false positive (FP) denotes instances where the model incorrectly predicts the presence of the positive class, indicating a misclassification of negative instances as positive. Specifically, in sentiment analysis, FP represents the number of reviews that are actually negative but have been erroneously classified as positive by the model. For example, an FP occurs when the model predicts a review like “Completely disappointed with this product” as positive. False positives can result in misleading insights about customer satisfaction and reduce the trustworthiness of automated sentiment reporting. Therefore, understanding and minimizing false positives is crucial for improving the model’s precision.

FN = False Negative

In a confusion matrix, a false negative (FN) signifies instances where the model erroneously predicts the absence of the positive class, reflecting a failure to identify instances that belong to the positive class.

Specifically, in binary sentiment classification, FN represents the count of reviews that are actually positive but have been incorrectly classified as negative by the model. For instance, a false negative would occur when the model labels a review like "Great value for money and excellent service" as negative. False negatives can lead to underestimation of positive feedback, affecting business strategies that rely on accurate sentiment insights. Hence, recognizing false negatives is essential for assessing the model's recall and its effectiveness in capturing true customer satisfaction.

9.2 RESULTS

This project is coded using Google Colab and below are the code and output screens with comments.

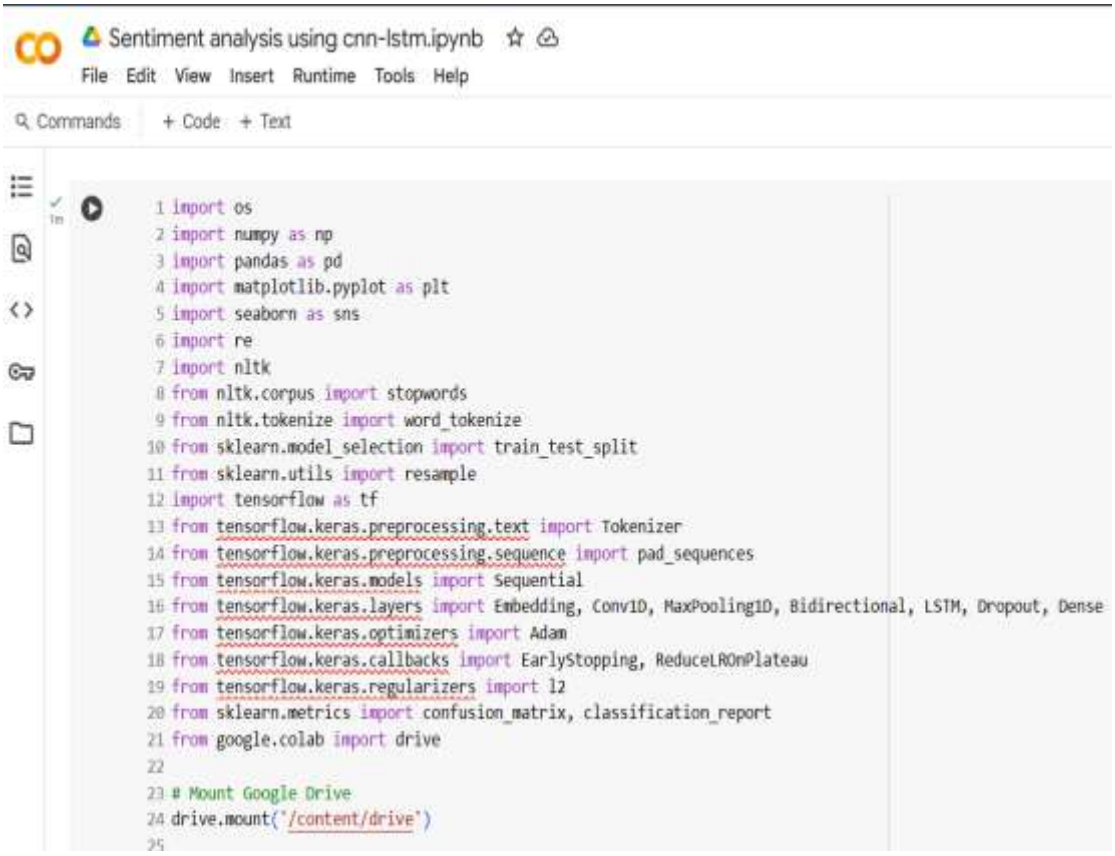
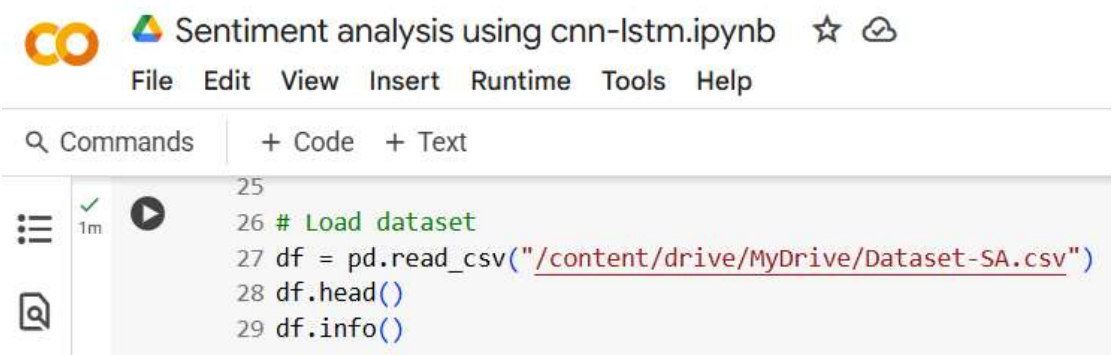


Fig 9.2.1: Importing Packages

In above screen importing require packages and classes for our proposed model to predict sentiment of customer reviews.



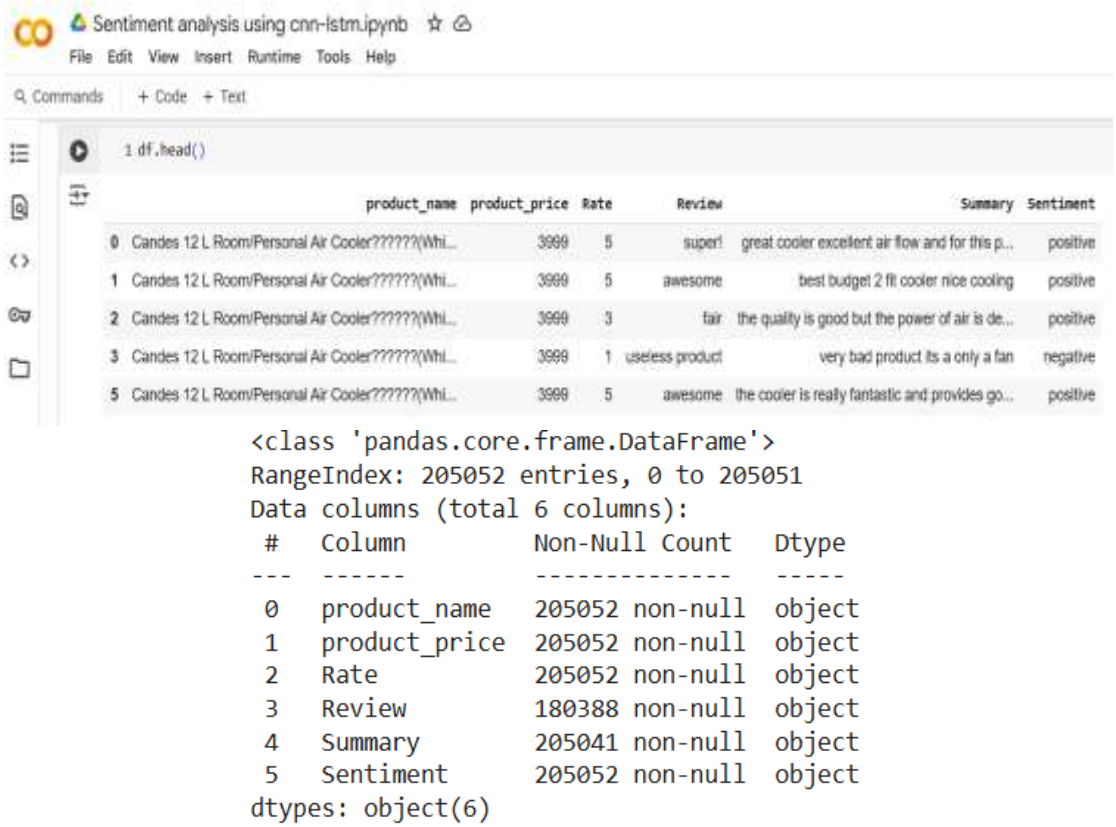
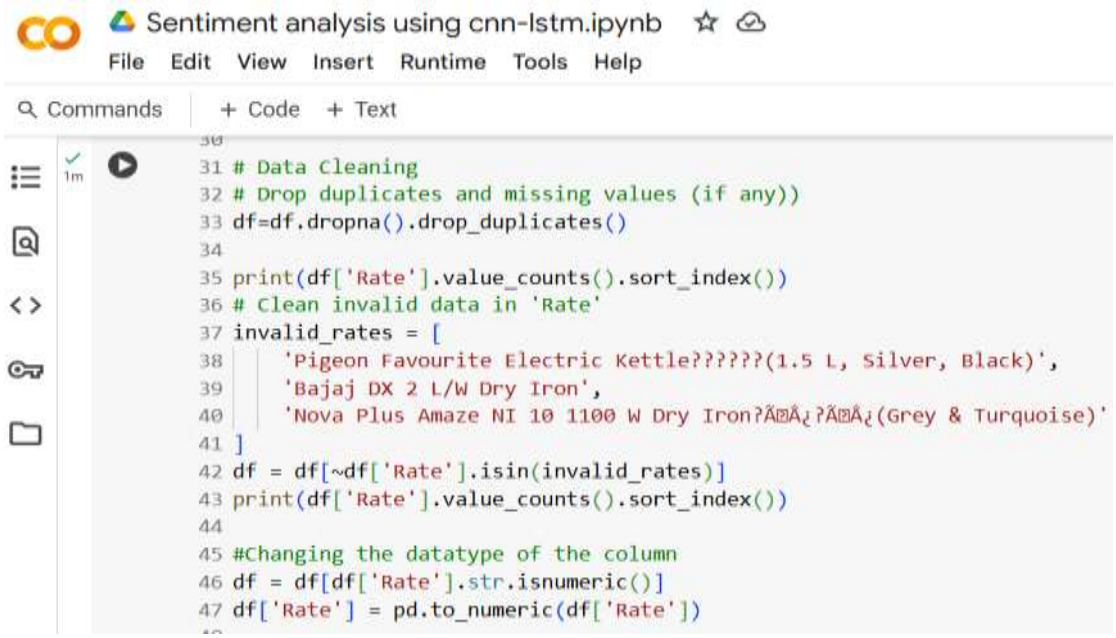


Fig 9.2.2: Loading Dataset

In the above screen the dataset containing flipkart reviews is loaded into google colab notebook and first 5 rows and information about the dataset are displayed.



```
Rate
1                                                                    17475
2                                                                     5279
3                                                                    12500
4                                                                    30484
5                                                                    88726
Bajaj DX 2 L/W Dry Iron                                                  1
Nova Plus Amaze NI 10 1100 W Dry Iron?ÃÂ¿?ÃÂ¿(Grey & Turquoise)        1
Pigeon Favourite Electric Kettle??????(1.5 L, Silver, Black)             1
Name: count, dtype: int64
Rate
1    17475
2     5279
3    12500
4    30484
5    88726
Name: count, dtype: int64
```

Fig 9.2.3: Data Cleaning

In the above screen data cleaning is performed like dropping duplicates, missing values and invalid rate values in the dataset.

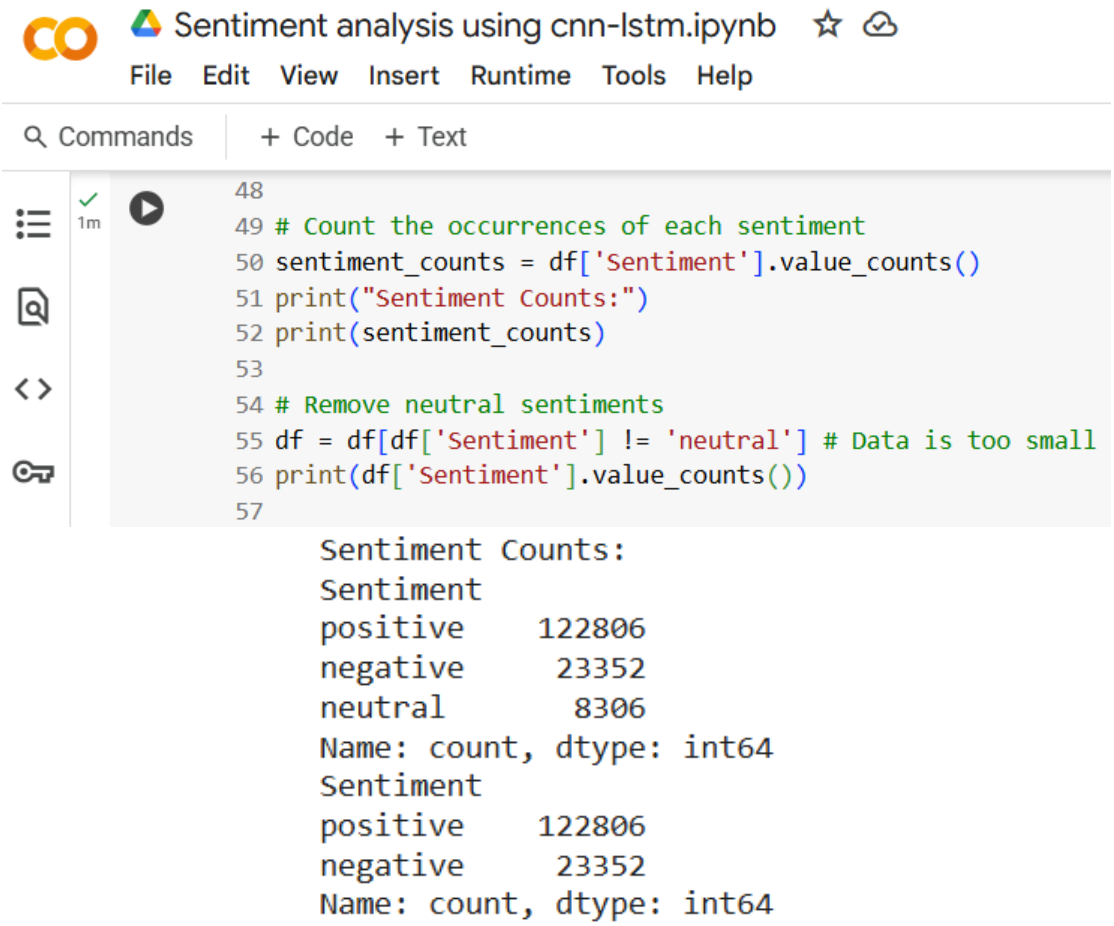


Fig 9.2.4: Removing Neutral Sentiments

In the above screen neutral sentiments are removed from the dataset because the data for neutral sentiment is too smaller than positive and negative sentiments.

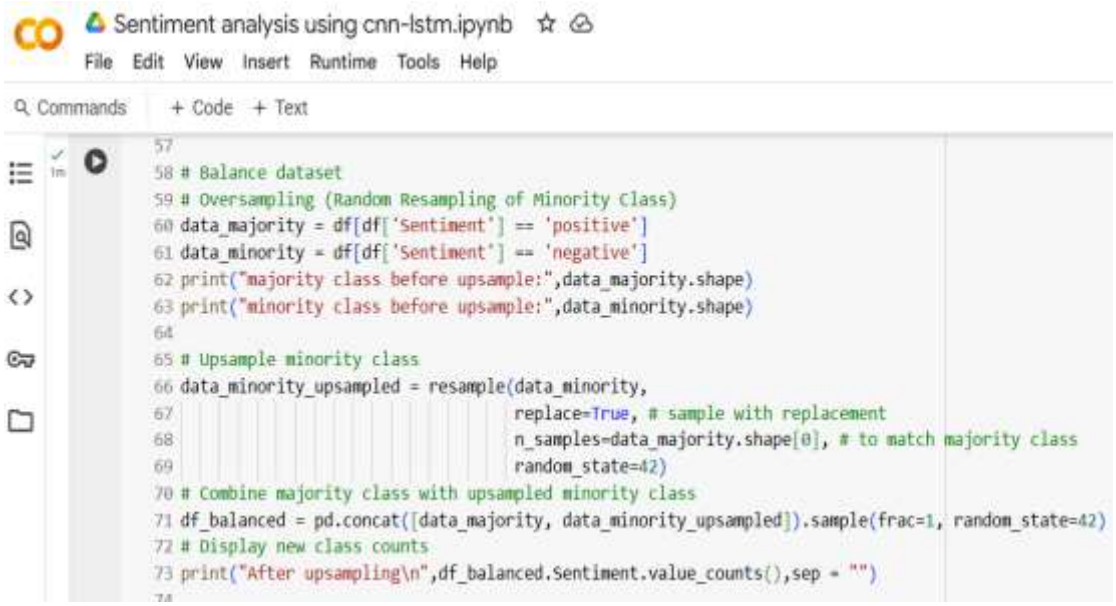


Fig 9.2.5: Balancing dataset

In the above screen the imbalanced dataset is balanced by using random resampling technique to upsample minority class in the dataset.

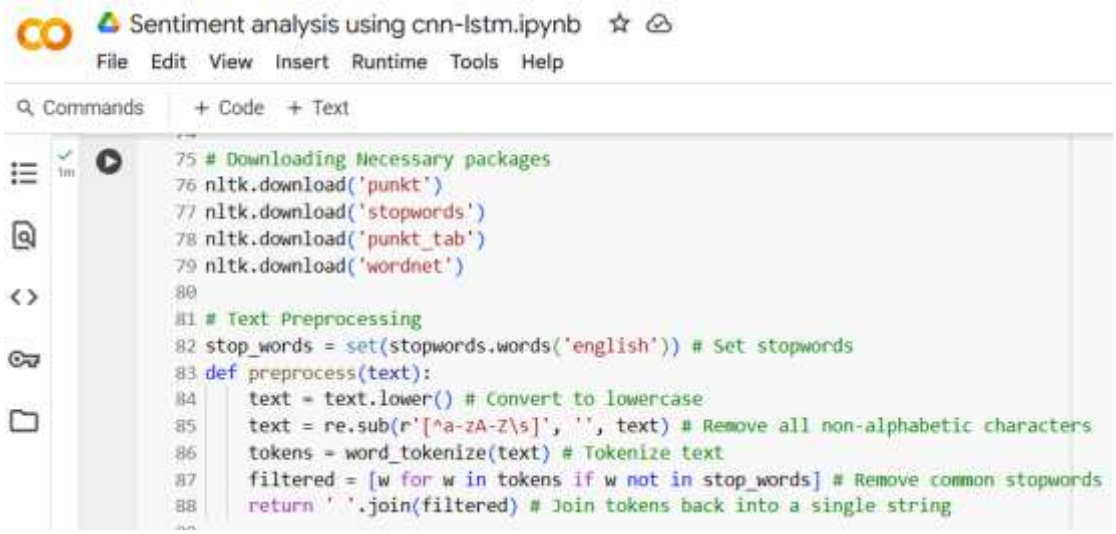


Fig 9.2.6: Text Preprocessing

In the above screen downloading necessary packages for stopwords and preprocessing like conversion of text into lowercase, removing non-alphabetic characters, tokenization are performed and then defining function to clean review text.

Fig 9.2.7: Splitting the dataset

In above screen we split the dataset into training set and testing set using train_test_split method.

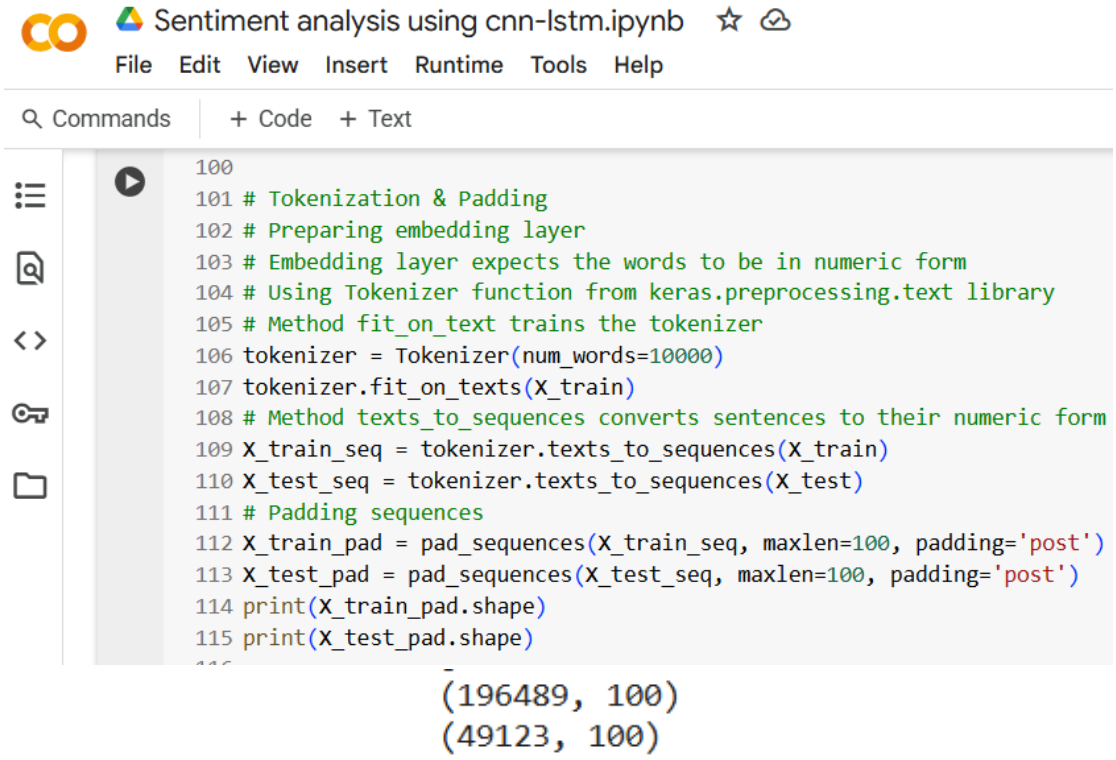


Fig 9.2.8: Tokenization & Padding

In above screen tokenizing text into sequences of integers and padding the sequences to max_length.

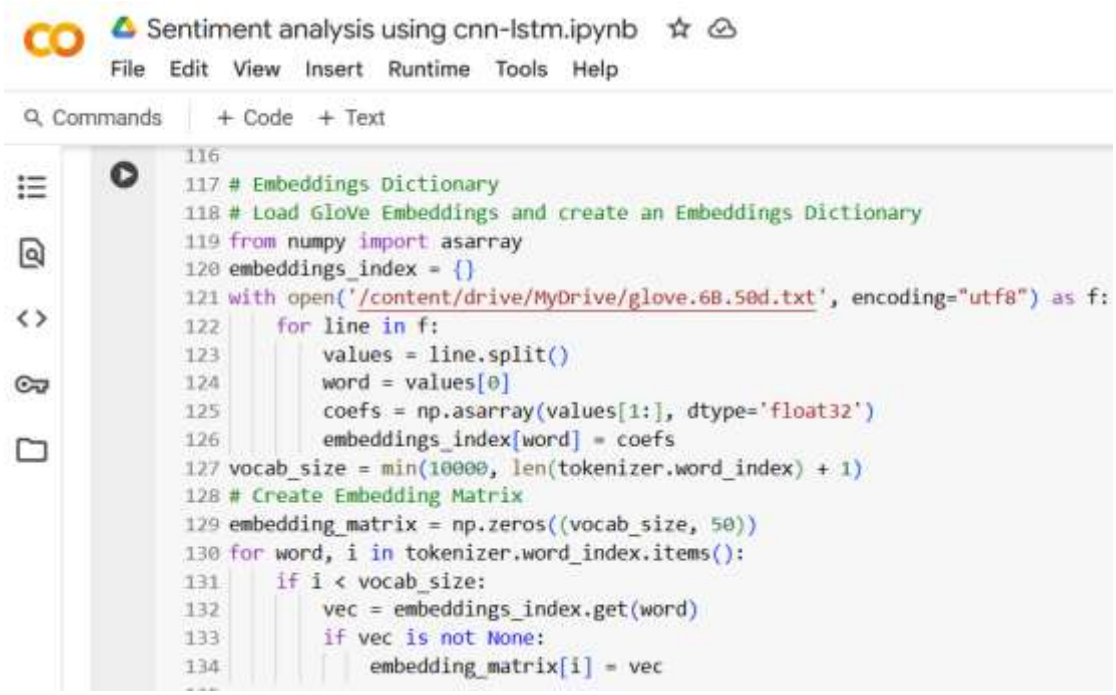


Fig 9.2.9: Load GloVe embeddings and create matrix

In the above screen we load GloVe embeddings and create an embedding matrix for deep learning model.

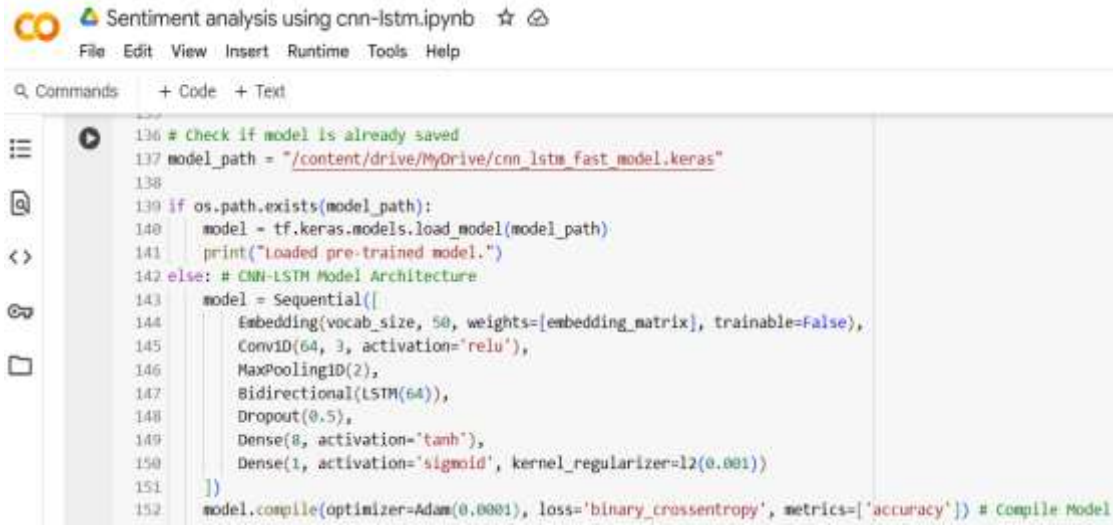


Fig 9.2.10: Using CNN-LSTM Model

In the above screen we build a deep learning model using techniques which are CNN and LSTM. The model is compiled using Adam optimizer.

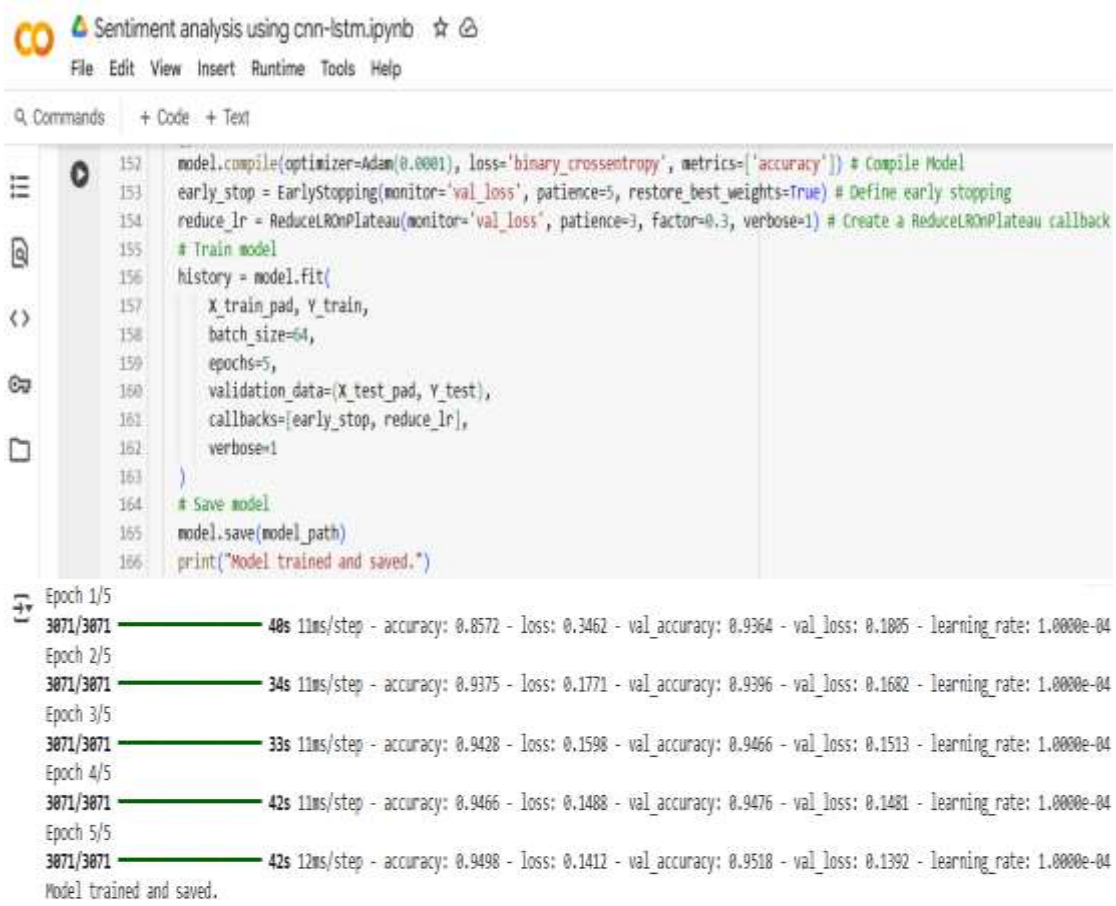


Fig 9.2.11: Train and save model

In above screen early stopping is defined to stop training when model stops improving on validation set and to prevent overfitting and ReduceLROnPlateau is created to reduce learning rate based on loss of model. The model is trained and saved in keras format.

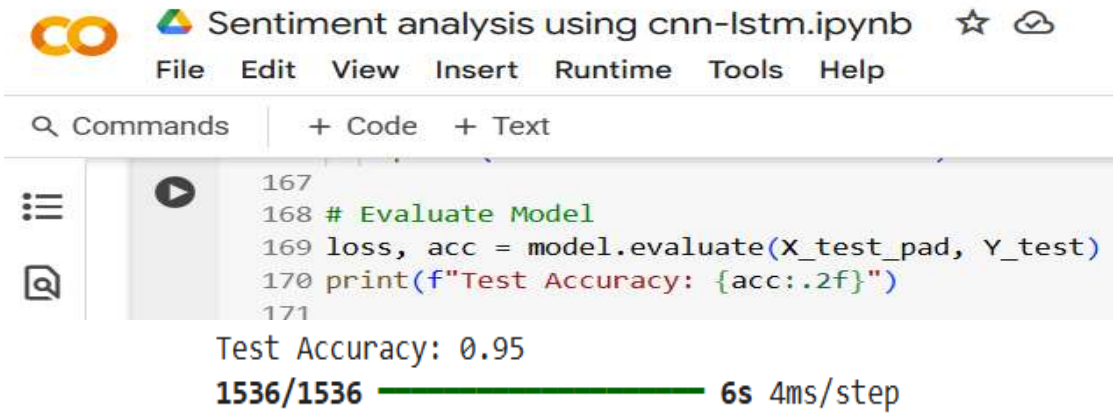
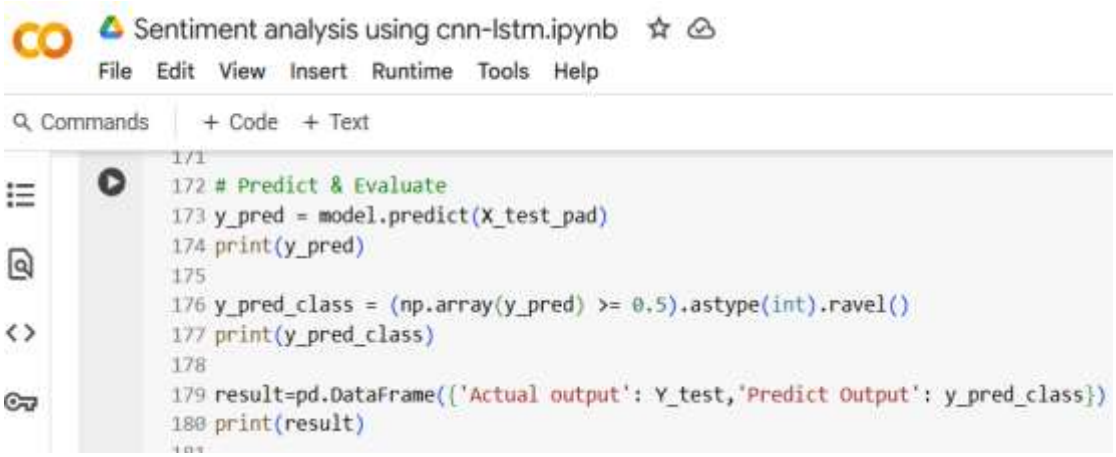


Fig 9.2.12: Evaluation of model

In the above screen the accuracy and loss of model is evaluated and test accuracy is displayed.




```
[[0.8696091 ]
 [0.9913966 ]
 [0.01684811]
 ...
 [0.00888741]
 [0.01177241]
 [0.00945278]]
[1 1 0 ... 0 0 0]
       Actual output  Predict Output
0                  1               1
1                  1               1
2                  0               0
3                  0               0
4                  0               0
...              ...             ...
49118              1               1
49119              0               0
49120              0               0
49121              0               0
49122              0               0

[49123 rows x 2 columns]
```

Fig 9.2.13: Prediction & Evaluation on test data

In the above screen predicting and evaluating test data to compare between actual and predicted outputs.

Fig 9.2.14: Code for Confusion Matrix

In the above screen we use confusion_matrix() function to get the confusion matrix and plot the confusion matrix by defining labels based on encoded values.

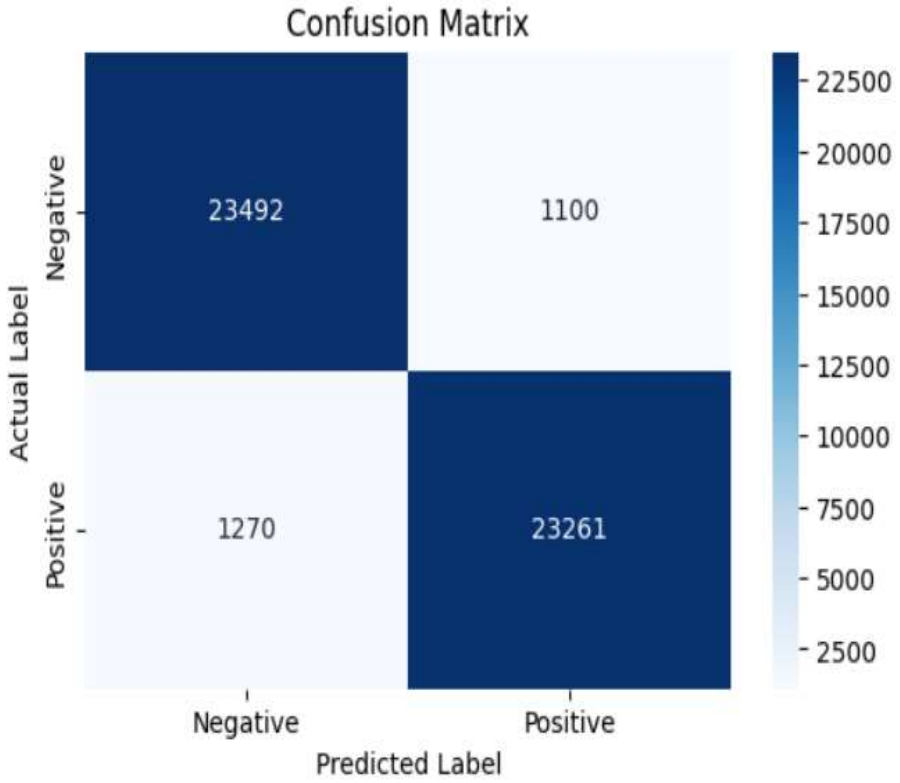
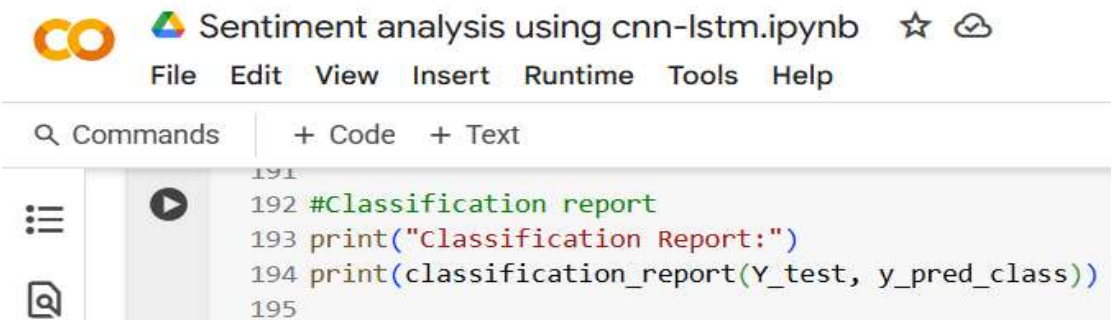


Fig 9.2.15: Confusion Matrix of proposed system



```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.95     24592
           1       0.95      0.95      0.95     24531

    accuracy                           0.95     49123
   macro avg       0.95      0.95      0.95     49123
weighted avg       0.95      0.95      0.95     49123
```

Fig 9.2.16: Classification Report

Fig 9.2.17: User input & output

In the above screen when a user gives a product review as input, the predicted sentiment is returned in the output as Positive. The proposed system takes a product review as an input from user, predicts its sentiment and gives either Positive or Negative as output.

EVALUATION METRICS	RESULTS
ACCURACY	95.18%
PRECISION	95.48%
RECALL	94.82%
F1-SCORE	95.15%

Table 9.2.1: Overview of results

GRAPHICAL REPRESENTATION

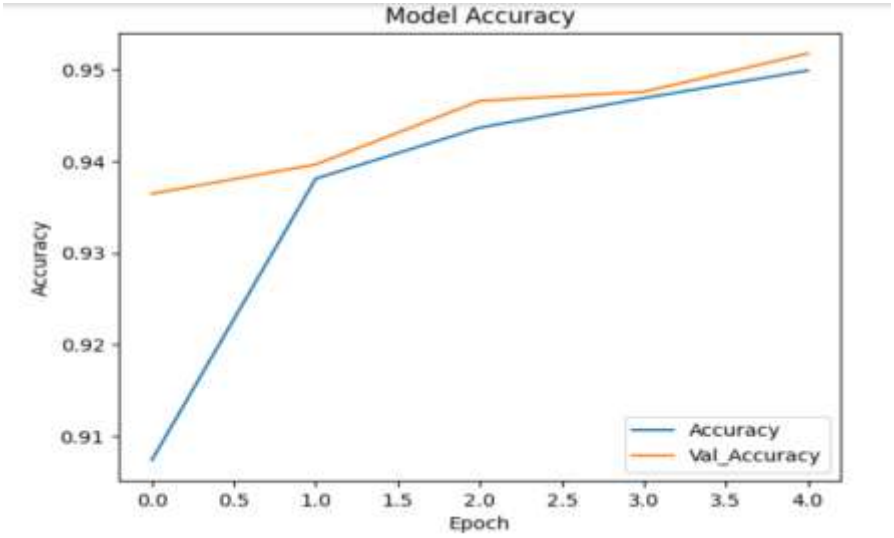


Fig 9.2.18: Accuracy of model

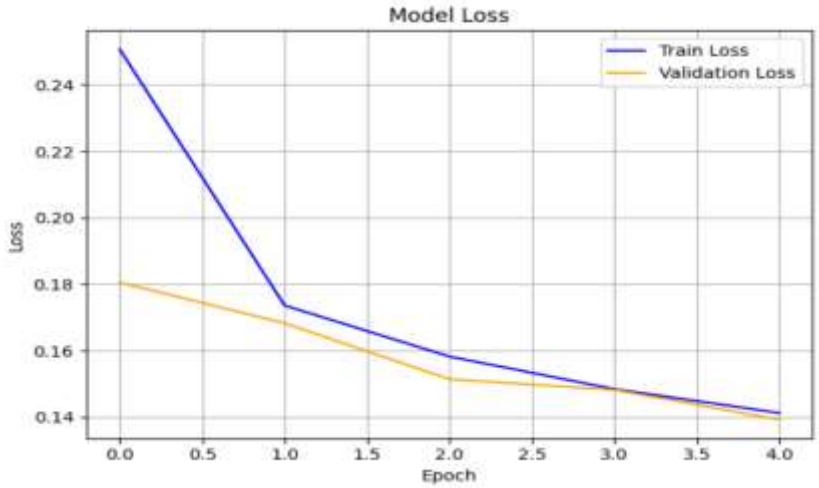


Fig 9.2.19: Model loss

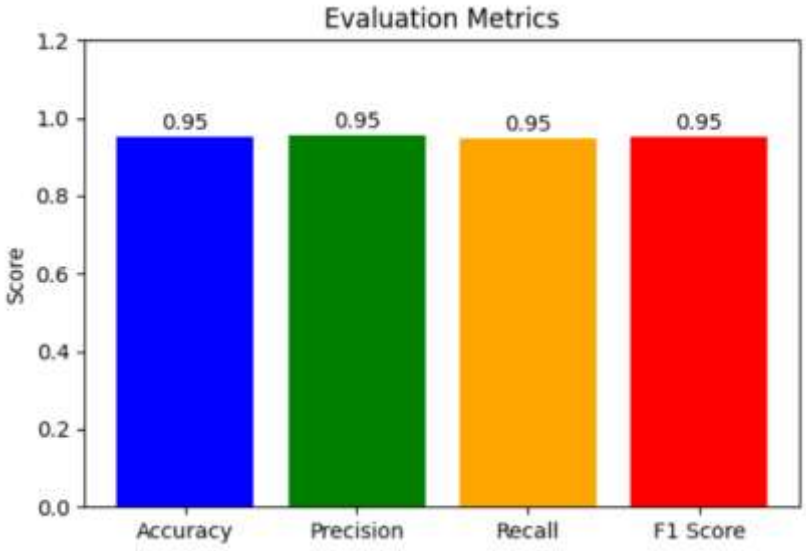


Fig 9.2.20: Performance Metrics of proposed system (Sentiment Analysis on E-commerce product reviews using Deep Learning)

In the proposed system, we used CNN-LSTM classifier which got results as:

- The Accuracy of the model is 95.18% (approx.).
- The Precision of the model is 95.48% (approx.).
- The Recall of the model is 94.82% (approx.).
- The F1-score of the model is 95.15% (approx.).

OUTPUT:

When we entered the input text it is going to classify whether the data is Positive or Negative.

CHAPTER 10
CONCLUSION AND FUTURE WORK

CONCLUSION:

Sentiment Analysis is a rapidly growing research area that provides valuable insights into customer opinions and experiences, especially in the e-commerce domain. Our proposed system performs sentiment analysis on e-commerce product reviews using deep learning techniques. The model is trained and evaluated on the Flipkart product reviews dataset.

Sentiment Analysis is a research area that has a lot of scopes and also has a large dataset. Our model is run against the existing dataset. From Table 2, we conclude CNN-LSTM algorithms show a Maximum Accuracy of up to 95%. So, using this classifier we built our classification model for sentiment classification and prediction. The user can enter the text or keyword and check the reliability of the review.

In this system, users can directly enter their product reviews, and the sentiment (Positive, Negative) is predicted using the trained deep learning model in Python without using any web API.

This proposed system helps e-commerce platforms, sellers, and customers to understand product feedback efficiently, improving product quality and customer satisfaction.

RECOMMENDATIONS FOR FUTURE STUDIES:

In our future work, we aim to create our own customized and updated dataset for e-commerce product reviews across various categories and platforms. The dataset will include the latest customer reviews to reflect recent trends, product updates, and user preferences.

Further, we plan to enhance our model by applying advanced deep learning architectures such as Transformer-based models (BERT, RoBERTa) for better accuracy and sentiment understanding. We also look forward to incorporating aspect-based sentiment analysis, which will allow identifying sentiment for specific product features.

Additionally, real-time review analysis and integration with e-commerce platforms could be developed, enabling instant feedback analysis for new products.

The feasibility of this project has been analyzed during the system design phase, and the implementation was carried out within the available resources. Since most of the tools and libraries used in the development of this system are open-source and freely available, the cost incurred was minimal.

This makes the proposed system economically feasible, efficient, and scalable for future improvements.

REFERENCES

1. R. Harika, M. Rajasekhar, and S. V. S. Ramakrishna, "Sentiment Analysis on Customer Reviews of E-commerce Sites Using Deep Learning," *International Journal of Advanced Research in Science, Communication and Technology* (IJARSCT), vol. 4, no. 6, pp. 723–728, 2023.
2. T. Manikandan, R. Venkatesan, and A. Ranjith, "Sentiment Analysis of E-commerce Product Reviews Using NLP Techniques," *International Journal of Scientific & Technology Research*, vol. 10, no. 5, pp. 1001–1006, 2021.
3. K. Rajalakshmi, B. Anuradha, and V. Prabha, "Sentiment Analysis of Online Product Reviews Using a Hybrid Approach," *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 10, pp. 2345–2351, 2021.
4. R. Priya and A. G. Ramakrishnan, "A Comparative Study of Machine Learning Techniques for Sentiment Analysis on Product Reviews," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 11, no. 1, pp. 111–117, 2021.
5. Sowmiya G., R. Deepika, and M. R. Sangeetha, "Sentiment Analysis on Flipkart Product Reviews Using Machine Learning," *International Journal of Innovative Research in Technology*, vol. 8, no. 3, pp. 567–572, 2021.
6. A. Sharma and D. Dey, "Sentiment Analysis on Product Reviews Using Machine Learning Techniques," *Procedia Computer Science*, vol. 152, pp. 202–207, 2019.
7. P. K. Rout, A. Singh, and S. Tripathy, "Sentiment Analysis of Customer Product Reviews Using Machine Learning," *International Journal of Computer Sciences and Engineering*, vol. 6, no. 11, pp. 724–730, 2018.
8. V. N. Patil and P. K. Bharne, "Sentiment Analysis on E-commerce Reviews Using SVM and Naïve Bayes," *International Journal of Engineering Research and Applications*, vol. 10, no. 3, pp. 45–50, 2020.
9. S. K. J. Al Amrani and A. M. Niyaz, "Performance Evaluation of Machine Learning Algorithms in Sentiment Analysis," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 2, pp. 361–368, 2020.
10. M. S. Khan, M. A. Khan, and F. Ali, "A Machine Learning Approach for Sentiment Analysis on Product Reviews," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 9, pp. 384–391, 2019.
11. A. Singh and P. Kumar, "Sentiment Analysis on E-commerce Reviews Using Machine Learning," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 6, no. 1, pp. 175–179, 2020.
12. N. Jain and S. R. Kolhe, "Sentiment Analysis Using Supervised Machine Learning Algorithms," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 6, pp. 1927–1932, 2018.
13. A. Kaur and A. Sharma, "Sentiment Analysis of Customer Reviews Using Machine Learning," *International Journal of Computer Applications*, vol. 182, no. 6, pp. 12–17, 2018.
14. M. J. Nirmal and T. Praveen, "Sentiment Analysis Using Naïve Bayes and SVM Classifiers," *International Journal of Computer Applications*, vol. 179, no. 11, pp. 7–12, 2018.
15. P. Joshi, "Sentiment Analysis of Product Reviews Using Random Forest Algorithm," *International Journal of Advanced Research in Computer Science*, vol. 9, no. 5, pp. 169–172, 2018.

16. V. Sharma and R. Sharma, "Sentiment Classification Using Hybrid Machine Learning Model," *Procedia Computer Science*, vol. 167, pp. 2318–2327, 2020.
17. A. Vaswani, N. Shazeer, and N. Parmar, "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.
18. S. Yadav and V. Vishwakarma, "Sentiment Analysis Using Machine Learning for Online Reviews: A Survey," *Journal of Information and Optimization Sciences*, vol. 40, no. 6, pp. 1235–1245, 2019.
19. R. Gupta and D. Gupta, "Comparative Study of Sentiment Analysis Using NLP Techniques," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, no. 4, pp. 7456–7463, 2017.
20. S. Patel and D. Patel, "Sentiment Analysis on Flipkart Reviews Using Machine Learning Algorithms," *International Journal of Engineering Research & Technology (IJERT)*, vol. 8, no. 6, pp. 231–235, 2019.
21. H. Chauhan and J. Nanda, "Sentiment Analysis Using Logistic Regression," *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 5, pp. 123–126, 2020.
22. S. Roy and M. Bhattacharya, "Sentiment Analysis on Customer Reviews Using Ensemble Learning," *International Journal of Scientific & Engineering Research*, vol. 11, no. 4, pp. 1112–1116, 2020.
23. D. Sharma and A. S. Chauhan, "A Survey on Sentiment Analysis Algorithms and Applications," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 3, pp. 424–429, 2019.
24. A. Maroof, S. Wasi, S. I. Jami, and M. S. Siddiqui, "Aspect Based Sentiment Analysis for Service Industry," *IEEE Access*, vol. 12, pp. 109702–109713, 2024.
25. M. P. Dave and R. H. Jhaveri, "Sentiment Analysis with Machine Learning: A Review," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 6, no. 6, pp. 343–347, 2017.
26. P. K. Soni and R. Rambola, "A Survey on Implicit Aspect Detection for Sentiment Analysis: Terminology, Issues, and Scope," *IEEE Access*, vol. 10, pp. 63932–63957, 2022.
27. X. X. Ye and Y. Xu, "ALBERT-C-CNN Based Aspect-Level Sentiment Analysis," *IEEE Access*, vol. 9, pp. 94748–94755, 2021.
28. S. Hu, A. Kumar, F. Al-Turjman, S. Gupta, and S. Seth, "Reviewer Credibility and Sentiment Analysis Based User Profile Modelling for Online Product Recommendation," IEEE Access, vol. 8, pp. 26172–26189, 2020.
29. A. Noor and M. Islam, "Sentiment Analysis for Women's E-Commerce Reviews Using Machine Learning Algorithms," IEEE Access, vol. 7, pp. 2870–2879, 2019.
30. Y. H. Ko, P.-Y. Hsu, Y.-C. Liu, and P.-C. Yang, "Confirming Customer Satisfaction With Tones of Speech," *IEEE Access*, vol. 10, pp. 83236–83248, 2022.