AMAZON PRODUCT REVIEW CLASSIFICATION USING NLP & ML

A project report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

DEPARTMENT OF CSE – ARTIFICIAL INTELLEGENCE AND MACHINE LEARNING

Submitted By

KURUKURI VIJAYA LAKSHMI	-	**21B21A4220**
JYOTHULA SANTHI PRIYA	-	**21B21A4228**
MATTAPARTHI ANJALI NAGAVALLI	-	**21B21A4217**
PANCHADA GNANESWARI	-	**21B21A4211**
ATTILI PUJITHA	-	**21B21A4215**

Under the Esteemed Guidance of

Ms. A. B. L. PRIYA, *M. Tech,*

Assistant Professor

DEPARTMENT OF CSE – AI & ML

KAKINADA INSTITUTE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE & Affiliated to JNT University Kakinada)

Yanam Road, Korangi-533461, E.G. Dist. (A.P)

Phone no: 0884-234050, 2303400 Fax no: 0884-2303869

2021-2025

KAKINADA INSTITUTE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE & Affiliated to JNT University Kakinada) Yanam Road, Korangi-533461, E.G. Dist. (A.P)

Phone no: 0884-234050, 2303400 Fax no: 0884-2303869

DEPARTMENT OF CSE – ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

BONAFIDE CERTIFICATE

This is to certify that the project entitled **"AMAZON PRODUCT REVIEW CLASSIFICATION USING NLP AND ML"** the bonafide record of work done by **KURUKURI VIJAYA LAKSHMI, JYOTHULA SANTHI PRIYA, MATTAPARTHI ANJALI NAGAVALLI, PANCHADA GNANESWARI, ATTILI PUJITHA** bearing with **ROLL NO: 21B21A4220, 21B21A4228, 21B21A4217, 21B21A4211, 21B21A4215** in partial fulfilment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY in Computer Science & Engineering-AIML** in **Kakinada Institute of Engineering & Technology, Korangi, affiliated to Jawaharlal Nehru Technological University, KAKINADA**.

INTERNAL GUIDE

(Ms. A. B. L. PRIYA, M. Tech)
Associate Professor
Department of CSE-AIML

HEAD OF THE DEPARTMENT

(Ms. N. JEEVANA DEEPA, M. Tech)
Associate Professor
Department of CSE-AIML

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I would like to take the privilege of the opportunity to express my gratitude for the Project work of **"AMAZON PRODUCT REVIEW CLASSIFIVATION USING NLP AND ML"** which enabled us to express our special thanks to our honourable Chairman of the institution **Sri. P.V. Viswam**.

I am thankful to Principal **Prof. D Revathi,** who has shown keen interest in us and encouraged us by providing all the facilities to complete my project successfully. I express my gratitude to our beloved Head of the Department of **CSE – AIML, Ms. N JEEVANA DEEPA, M. Tech** for assisting me in completing my project work.

I am extremely thankful to our Project Review Committee who has been a source of inspiration for us throughout my project and for their valuable advice in making my project a success.

I express my sincere thanks to my beloved supervisor **S. SRINIVAS**, **Assistant Professor, Dept. of CSE - AIML** who has been a source of inspiration for me throughout my project and for his valuable pieces of advice in making my project a success.

I wish to express my sincere thanks to all teaching and non-teaching staff of the **CSE – Artificial Intelligence And Machine Learning Department**. I wish to express my special thanks to all the faculty members of our college for their concern in subjects and their help throughout my course.

I am very thankful to my parents, and all my friends who had given me good cooperation and suggestions throughout this project and helped me in successful completion.

KURUKURI VIJAYA LAKSHMI	**-**	**21B21A4220**
JYOTHULA SANTHI PRIYA	**-**	**21B21A4228**
MATTAPARTHI ANJALI NAGAVALLI	**-**	**21B21A4217**
PANCHADA GNANESWARI	**-**	**21B21A4211**
ATTILI PUJITHA	**-**	**21B21A4215**

DECLARATION

I hereby declare that the project work entitled **"AMAZON PRODUCT REVIEW CLASSIFICATION USING NLP AND ML"** Submitted to the **Kakinada Institute of Engineering and Technology** affiliated to **JNTU Kakinada,** a record of an original work done by me under the guidance of **Ms. A. B. L. PRIYA , M. Tech., Assistant Professor** in the **Department of CSE - AIML** and this project work is submitted to the partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in CSE - AIML.** The results embodied in this project have not been submitted to any other University or Institute for the award of any Degree or Diploma.

KURUKURI VIJAYA LAKSHMI	**-**	**21B21A4220**
JYOTHULA SANTHI PRIYA	**-**	**21B21A4228**
MATTAPARTHI ANJALI NAGAVALLI	**-**	**21B21A4217**
PANCHADA GNANESWARI	**-**	**21B21A4211**
ATTILI PUJITHA	**-**	**21B21A4215**

Place:

Date:

ABSTRACT

The rapid expansion of e-commerce has resulted in an overwhelming volume of user-generated content, especially in the form of product reviews. Understanding the sentiment expressed in these reviews is essential for businesses to make informed decisions and enhance customer experience. This project presents an intelligent sentiment analysis system that classifies Amazon product reviews into three categories: Positive, Neutral, and Negative. By combining Natural Language Processing (NLP) techniques with Machine Learning (ML) models

The system leverages a hybrid methodology: it employs both keyword-based heuristics and a trained Logistic Regression classifier. Reviews undergo preprocessing that includes lowercasing, punctuation removal, and stop word filtering, followed by transformation into TF-IDF vectors to capture linguistic patterns. A predefined lexicon of sentiment keywords allows the model to quickly infer polarity in straightforward cases, while more nuanced inputs are analysed using the trained classifier. The model is trained on a curated dataset and fine-tuned for balanced performance across all sentiment classes.

A Flask-based web application serves as the user interface, featuring an intuitive design, interactive feedback, and a responsive layout enriched with particle-based animation for enhanced user engagement. Users can input any review text and receive immediate feedback on the sentiment along with a confidence score and breakdown of sentiment probabilities.

This project highlights the effectiveness of integrating rule-based language understanding with statistical learning in real-world sentiment analysis. It is scalable, interpretable, and practical for deployment in customer feedback systems, with potential future expansion into multi-language support, fine-grained emotion classification, and opinion summarization.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
ML	Machine Learning
NLP	Natural Language Processing
TF-IDF	Term Frequency-Inverse Document Frequency
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
JINJA2	Templating Engine for Python/Flask
NLTK	Natural Language Tool Kit

CHAPTER - 1

INTRODUCTION

INTRODUCTION

Title: *Amazon Product Review Classification*

Sentiment analysis of product reviews is a widely researched task in the field of Natural Language Processing (NLP) and Machine Learning (ML), focusing on determining the emotional tone behind user-generated text. It plays a crucial role in understanding customer feedback, improving services, and shaping business decisions in e-commerce platforms. This project explores the classification of Amazon product reviews into positive, neutral, or negative sentiments, using a combination of rule-based NLP techniques and supervised ML models.

In this project, we aim to develop a sentiment classification system that processes raw review text and predicts its underlying sentiment. The core components of the project include data preprocessing, feature extraction using TF-IDF, model training using Logistic Regression, heuristic sentiment detection using keyword dictionaries, and the deployment of a web application using Flask for real-time review analysis.

The first step in the project involves collecting and cleaning Amazon product review data. Preprocessing includes lowercasing the text, removing punctuation, eliminating stop words, and applying tokenization. We also develop a custom sentiment lexicon to identify strong positive and negative cues in the reviews. This dual approach allows the system to process reviews through both keyword-based rules and machine-learned patterns.

Once preprocessing is complete, we convert the cleaned text into numerical vectors using Term Frequency–Inverse Document Frequency (TF-IDF). This transformation enables the model to consider the importance of words not just within individual reviews but across the entire dataset. The processed vectors are then used to train a Logistic Regression model, chosen for its simplicity, interpretability, and effectiveness in binary and multiclass text classification.

After training, the model is evaluated on test data and integrated into a web-based application using Flask. The web interface allows users to input reviews and view predictions in real-time, complete with a breakdown of sentiment probabilities. Jinja2

templating is used to dynamically display results on the result page, including confidence scores and detailed sentiment distribution.

The project also incorporates an interactive user experience using modern front-end effects like animated particles and responsive layouts. These design choices make the system not only functional but also visually engaging.

In conclusion, this project demonstrates a practical application of NLP and ML in the e-commerce domain. By combining rule-based and model-based sentiment detection methods, we deliver an accurate and interpretable tool for classifying product reviews. This work lays the foundation for further enhancements, including multilingual support, advanced deep learning models, and integration with large-scale feedback platforms

1.1 PYTHON

Python is a high-level, interpreted programming language known for its simplicity and readability. It was created by Guido van Rossum and first released in 1991. Python's design philosophy emphasizes code readability with its use of significant whitespace.

Python is versatile and can be used for various types of programming, including web development, data analysis, artificial intelligence, and scientific computing. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

One of Python's key features is its extensive standard library, which provides built-in modules for tasks such as file I/O, networking, and data manipulation. Additionally, Python has a vibrant ecosystem of third-party libraries and frameworks that extend its capabilities, such as NumPy and pandas for data analysis, Flask and Django for web development, and TensorFlow and PyTorch for machine learning.

Python's syntax is clean and easy to learn, making it an excellent choice for beginners. Its readability and expressiveness help developers write code that is concise and maintainable. Python uses dynamic typing and automatic memory management, which

simplifies development by reducing the complexity of managing memory allocation and type declarations.

1.2 MACHINE LEARNING

Machine learning is a subset of artificial intelligence that involves the development of algorithms and statistical models that enable computers to learn from and make predictions or decisions based on data, without being explicitly programmed. It focuses on the development of computer programs that can access data and use it to learn for themselves.

Machine learning algorithms are categorized into three main types: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the algorithm is trained on a labelled dataset, where each example is paired with the correct answer.

The algorithm learns to map inputs to outputs, making it capable of making predictions on new, unseen data. Unsupervised learning, on the other hand, deals with unlabelled data, where the algorithm tries to learn the underlying structure or distribution in the data. Reinforcement learning involves an agent learning to make decisions by interacting with its environment. It receives feedback in the form of rewards or penalties, which helps it learn the best actions to take in different situations.

Fig 1.1:- Machine Learning

Machine learning is used in a wide range of applications, including but not limited to: image and speech recognition, medical diagnosis, financial forecasting, recommendation systems, and autonomous vehicles. It has the potential to revolutionize industries and improve efficiency and accuracy in decision-making processes.

Overall, machine learning is a powerful tool that enables computers to learn from data and improve over time, without being explicitly programmed. Its applications are vast and continue to grow as new algorithms and techniques are developed.

1.3 NATURAL LANGUAGE PROCESSING

Natural Language Processing (NLP) is a branch of artificial intelligence that enables computers to understand, interpret, and generate human language. It acts as a bridge between human communication and computer understanding, allowing machines to process and analyze large amounts of natural language data.

In NLP, algorithms are designed to extract meaning and patterns from human language, whether in the form of text or speech. These algorithms can perform tasks such as tokenization, part-of-speech tagging, sentiment analysis, machine translation, and named entity recognition. By processing text at multiple levels—morphological, syntactic, semantic, and pragmatic—NLP systems can comprehend and generate meaningful responses or insights from language data.

Fig 1.2:- Natural Language Processing

One of the key features of NLP is its ability to interpret unstructured text data and transform it into structured information that machines can use. Unlike traditional rule-based systems, modern NLP techniques use statistical and machine learning methods to learn language patterns from data.

NLP has become a transformative technology in various domains. In customer service, chatbots powered by NLP can handle queries and provide support. In sentiment analysis, NLP helps businesses gauge public opinion from reviews, tweets, and comments.

In healthcare, it aids in extracting information from patient records. In legal tech, it assists in document summarization and contract analysis.

Recent advancements in NLP have been fuelled by deep learning techniques. Models such as recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and transformers (like BERT and GPT) have significantly improved the accuracy and fluency of NLP systems. These models can handle tasks like language translation, summarization, and question-answering with human-like proficiency.

Training NLP models typically requires large text corpora and labelled data. Once trained, these models can be fine-tuned for domain-specific applications, making NLP a powerful tool in building intelligent, language-aware systems.

.

1.4.1 WORKING OF NLP

Natural Language Processing (NLP) requires vast amounts of language data to accurately interpret and understand human language. It performs multiple layers of text analysis to discern meaning, sentiment, structure, and intent—ultimately enabling computers to understand, generate, and respond to natural language. For example, to train a machine to recognize whether a product review is positive or negative, it must be exposed to thousands of labeled reviews across various contexts to learn language cues, tone, and patterns.

Two key technologies drive NLP systems: **machine learning algorithms** and **language models** such as RNNs, LSTMs, or transformer-based architectures like BERT and GPT.

Machine learning enables a computer to learn linguistic patterns from vast corpora of text. Instead of relying on rigid rules, the model “reads” large datasets, learning statistical associations between words, grammar structures, and context. Over time, and with sufficient examples, the model can differentiate between different tones, topics, or sentiments without explicit human intervention.

Language models serve as the brain of NLP systems. These models process text by breaking it down into tokens (usually words or sub-words), which are then embedded into numerical vectors capturing semantic and syntactic relationships. Just as a convolutional neural network (CNN) processes visual features, language models process textual features to identify meaning.

Fig 1.3:- Working of NLP

Through a series of training iterations, the model evaluates how well it understands input text by predicting the next word, classifying the sentiment, or answering questions. It continuously refines its predictions based on feedback, eventually achieving a high level of language understanding. This enables NLP applications to perform tasks like language translation, sentiment analysis, summarization, and conversational AI with impressive accuracy.

In summary, NLP teaches machines to “read” and “understand” human language by using deep learning algorithms and pretrained models trained on massive amounts of text data. Just like vision models learn to “see,” NLP models learn to “understand” and “generate” language in a way that closely mimics human communication.

1.4.2 THE HISTORY OF NLP

Natural Language Processing (NLP) has evolved significantly over the decades, transforming from simple rule-based systems into highly complex models capable of understanding and generating human language. Its history can be broadly divided into the following phases:

1. The Rule-Based Era (1950s–1980s)

NLP began as a part of artificial intelligence research in the 1950s. Early systems relied heavily on handcrafted rules and grammar-based approaches.

- In **1950**, Alan Turing proposed the famous **Turing Test** to evaluate machine intelligence through language interaction.
- In the **1960s**, systems like **ELIZA** (a rule-based chatbot mimicking a psychotherapist) were developed using pattern matching.
- Machine translation between languages, especially English to Russian, gained government interest during the Cold War but produced disappointing results due to limited grammar handling.

2. The Statistical Era (1990s–2010)

With the growth of the internet and access to large corpora of text, NLP research shifted towards data-driven approaches.

- **Statistical methods** like **Hidden Markov Models (HMMs)** and **Naive Bayes classifiers** became popular for tasks like part-of-speech tagging and spam detection.
- **n-gram models** and **TF-IDF** were widely used for text classification and information retrieval.
- This era focused more on **probabilities**, training models from labeled datasets rather than hardcoded rules.

3. The Machine Learning and Deep Learning Era (2011–Present)

This phase marked a major breakthrough in NLP thanks to advances in computational power, larger datasets, and deep learning techniques.

- **Word embeddings** like **Word2Vec** and **GloVe** enabled models to understand semantic relationships between words.
- **Recurrent Neural Networks (RNNs)** and **Long Short-Term Memory (LSTM)** networks allowed for better sequence processing.

- In **2018**, Google introduced the **Transformer architecture** and **BERT (Bidirectional Encoder Representations from Transformers)**, which revolutionized NLP.

- Today, models like **GPT (Generative Pre-trained Transformer)**, **T5**, and **XLNet** are state-of-the-art for a variety of language tasks including summarization, translation, and sentiment analysis.

4. Modern Applications and Accessibility (2020–Onward)

NLP models are now embedded in everyday technologies such as voice assistants, chatbots, real-time translators, sentiment analysis engines, and more.

- Open-source libraries like **spaCy**, **NLTK**, and **Transformers by Hugging Face** have made NLP accessible to developers and researchers globally.

- These advancements have enabled widespread adoption of NLP in industries like healthcare, e-commerce, finance, and education

1.4.3 NLP APPLICATIONS

There is a tremendous amount of research and innovation happening in the field of Natural Language Processing (NLP), and its applications are rapidly becoming embedded into our daily lives. NLP has grown far beyond academic use and is now a vital part of how businesses, governments, and individuals communicate with and through technology. With the explosive growth of text data from social media, customer reviews, emails, and voice commands, NLP plays a crucial role in converting this unstructured data into meaningful insights and intelligent interactions.

NLP enables machines to understand, interpret, and generate human language, and its applications are evident in a wide variety of industries and use cases:

Customer Service and Virtual Assistants

Companies like Amazon, Apple, and Google have integrated NLP into voice assistants such as **Alexa**, **Siri**, and **Google Assistant**. These assistants use NLP to understand user queries, provide spoken responses, set reminders, answer questions, and

even control smart home devices. **Chatbots** deployed on websites and apps also use NLP to understand customer requests and provide instant support, reducing the need for human intervention.

- **Sentiment Analysis and Brand Monitoring**

Businesses use NLP-based sentiment analysis tools to interpret customer feedback from reviews, tweets, or surveys. These systems can detect emotions (positive, negative, or neutral), enabling companies to understand public perception of their products or services in real time. For example, Amazon uses NLP to assess the tone of customer reviews and improve product recommendations or alert vendors to potential quality issues.

- **Language Translation**

Tools like **Google Translate** and **Microsoft Translator** rely on NLP to provide real-time language translation. NLP models break down the structure and semantics of one language and reconstruct the meaning in another, often preserving grammar, context, and idiomatic expressions. This application helps bridge communication gaps across cultures and countries.

- **Healthcare and Clinical Text Analysis**

In the healthcare domain, NLP is used to extract critical information from clinical notes, pathology reports, and discharge summaries. This aids doctors in identifying patient conditions, drug interactions, and treatment recommendations. For instance, IBM Watson Health has used NLP to parse medical literature and match cancer patients with clinical trials.

- **Text Summarization and Information Extraction**

NLP is also widely used in applications where users need quick summaries of lengthy documents or news articles. Automatic summarization tools generate concise and coherent versions of larger texts without losing key points. Legal tech companies use this for summarizing long contracts, while news agencies use it to create short briefs from full-length stories.

- **Speech Recognition and Voice Interfaces**

Technologies like **speech-to-text**, used in voice dictation software, smart speakers, and virtual meeting platforms, rely heavily on NLP. These systems convert spoken words into written text and then process the meaning behind those words to respond or act appropriately.

- **Search Engines and Recommendation Systems**

Search engines like **Google** and platforms like **YouTube**, **Netflix**, and **Amazon** use NLP to enhance search relevance and generate personalized recommendations. By analyzing user queries, NLP helps these systems return results that match the intent, not just the keywords.

- **Academic and Legal Document Analysis**

NLP tools are used in research to scan and categorize academic papers, extract citations, and suggest related literature. In law, it helps firms analyze massive volumes of legal documents, identify relevant cases, and even predict legal outcomes.

In conclusion, NLP is a cornerstone of modern AI, transforming the way we interact with machines and analyze language-based data. Its ability to interpret, generate, and act upon human language is unlocking a wide range of practical applications that touch almost every industry—from healthcare to e-commerce, education, entertainment, and beyond.

1.4.4 NLP EXAMPLES

Many organizations are increasingly turning to Natural Language Processing (NLP) to unlock the power of human language in digital form. However, building NLP models from the ground up requires vast datasets, skilled developers, and high-performance computing resources—elements that many businesses might not have at their disposal. To address this challenge, major tech companies like Google, IBM, Microsoft, and Hugging Face now offer NLP as a service through cloud-based APIs. These services come with pre-trained language models and tools, significantly reducing the time, cost, and complexity of developing intelligent language applications.

One of the most fundamental tasks in NLP is **text classification**. This involves assigning a label or category to a given piece of text. For example, a sentiment analysis system may classify a customer review as positive, neutral, or negative. Similarly, news articles can be categorized under topics like sports, politics, or technology. Text classification is widely used in spam detection, content moderation, and customer feedback analysis.

Named Entity Recognition (NER) is another common NLP task that identifies and classifies entities such as names of people, places, dates, organizations, and more within a body of text. For example, in the sentence "Amazon launched its new store in New York on January 1, 2024," an NER model would recognize "Amazon" as an organization, "New York" as a location, and "January 1, 2024" as a date. This task is critical in fields like information retrieval, finance, and legal analysis.

Sentiment analysis is used to determine the emotional tone behind a piece of text. Whether analyzing a tweet, product review, or survey response, sentiment analysis classifies the tone as positive, negative, or neutral. This helps organizations understand public opinion and make data-driven decisions. Brands use it to monitor their reputation and address customer concerns proactively.

Language translation is a powerful NLP application that converts text from one language to another. Services like Google Translate and Microsoft Translator use machine translation models to make websites, emails, and documents accessible across different

languages. This is essential for businesses expanding globally and for breaking down communication barriers in multicultural environments.

Text summarization is the task of automatically generating a concise summary of a longer document while preserving its key information. It can be extractive—pulling key sentences from the original text—or abstractive—rewriting the content in a shortened form. This is particularly useful in the legal, medical, and academic sectors where professionals need quick access to critical insights without reading entire reports.

Question answering allows systems to provide direct answers to user queries by understanding the context of a question. These systems read and comprehend large documents or passages and return precise answers. For instance, when asked "Who is the CEO of Tesla?" a question answering model can extract and return the correct answer from an article or database. This technology is central to search engines and virtual assistants.

Text generation is the task of generating human-like text based on a given prompt or context. Advanced models like GPT-3 and ChatGPT can write essays, compose emails, draft social media content, and even generate code. This capability is increasingly used in content creation, conversational AI, storytelling, and digital marketing.

Part-of-Speech (POS) tagging is the process of identifying the grammatical categories of words in a sentence, such as noun, verb, adjective, or adverb. For example, in the sentence "She quickly ran to the store," a POS tagger would label "She" as a pronoun, "quickly" as an adverb, and "ran" as a verb. This foundational task supports more complex NLP functions like parsing, grammar checking, and text-to-speech synthesis.

Each of these tasks demonstrates the remarkable versatility of NLP in solving real-world problems. From improving customer service to powering smart assistants, NLP continues to evolve and revolutionize the way humans interact with technology.

CHAPTER - II
LITERATURE SURVEY

LITERATURE SURVEY

Literature survey is a crucial phase in the software development lifecycle, particularly when implementing intelligent systems such as sentiment analysis tools. Before initiating the development process, it is essential to understand the scope of existing technologies, resources, timeframes, and economic feasibility. It also involves evaluating the available programming languages, operating systems, and machine learning techniques suitable for the project. Additionally, external support from research papers, online communities, and domain experts plays an integral role in shaping the direction and methodology of the system to be developed.

Sentiment analysis, also known as opinion mining, falls under the broad domain of Natural Language Processing (NLP). It deals with the identification and classification of sentiments expressed in textual data, particularly customer reviews, tweets, or feedback. The growing popularity of e-commerce platforms such as Amazon has resulted in an exponential increase in user-generated content. Manually processing this data is inefficient and time-consuming, hence the need for automated sentiment classification systems that can identify customer emotions and predict user satisfaction levels.

Early sentiment classification systems primarily relied on **rule-based methods** and **handcrafted lexicons**, such as SentiWordNet and AFINN. These approaches used predefined lists of sentiment words to identify the polarity of a given text. While such systems were straightforward and interpretable, they lacked the ability to understand context and nuances in language, often failing when applied to real-world, informal, or sarcastic expressions.

With the advancement of **machine learning**, more adaptive and accurate methods emerged. Supervised learning algorithms like **Naive Bayes**, **Support Vector Machines (SVM)**, and **Logistic Regression** became popular for text classification tasks. These models rely on labeled training data and statistical patterns extracted using features like **Bag of Words (BoW)** and **Term Frequency-Inverse Document Frequency (TF-IDF)**. Logistic Regression, in particular, gained popularity due to its simplicity and performance in binary classification scenarios, including sentiment polarity detection.

The evolution of **deep learning** further enhanced the capabilities of sentiment analysis systems. Models such as **Recurrent Neural Networks (RNNs)**, **Long Short-Term Memory (LSTM)** networks, and **Transformer-based architectures (e.g., BERT, RoBERTa)** revolutionized text representation and understanding. These models can capture word dependencies and contextual relationships, providing significantly better performance than traditional approaches. However, they often require large datasets and computational resources, which can limit their use in smaller-scale or resource-constrained projects.

Recent research emphasizes **hybrid models** that combine rule-based keyword detection with machine learning classifiers for improved accuracy and interpretability. In such systems, predefined sentiment keywords are used to instantly tag highly polarized content, while ambiguous or context-dependent reviews are passed through a trained ML model for deeper analysis. This approach ensures faster predictions without sacrificing precision in complex cases.

Flask-based web applications have become a popular framework for deploying machine learning models due to their lightweight structure and seamless integration with HTML/CSS frontends. By embedding models into interactive web interfaces, end-users can input real-time reviews and receive instant sentiment predictions, enhancing both usability and accessibility. This aligns with the trend of turning research prototypes into functional tools that businesses and individuals can benefit from directly.

Researchers have also explored **heuristic techniques**, such as combining sentiment scores from keyword dictionaries with ML output probabilities, to create more transparent models. Such explainable AI (XAI) models are preferred in domains like customer service, healthcare, and finance, where understanding *why* a prediction was made is just as important as the prediction itself.

To date, multiple approaches to sentiment classification have been proposed, each with its advantages and limitations. Some focus on accuracy, others on interpretability or processing speed. As sentiment analysis continues to evolve, the research community is exploring **multilingual sentiment detection**, **emotion recognition**, and **domain adaptation** to build more universal and inclusive models.

A literature survey on **sentiment analysis of Amazon product reviews using NLP and machine learning** covers a diverse range of research areas and technological

advancements. Below is a concise overview of key concepts, historical developments, and methods that have shaped the field:

1. **Early Approaches:**

   Initial efforts in sentiment analysis relied heavily on **rule-based methods** and **handcrafted sentiment lexicons** like SentiWordNet and AFINN. These systems functioned by scanning for positive or negative words in a sentence and assigning sentiment polarity based on predefined scores. While interpretable, these approaches struggled with sarcasm, negation handling, and context sensitivity.

2. **Machine Learning for Sentiment Analysis:**

   The shift from rule-based systems to statistical methods introduced supervised learning algorithms such as **Naive Bayes**, **Support Vector Machines (SVMs)**, and **Logistic Regression**. These models used features like **Bag of Words (BoW)** or **TF-IDF** to learn sentiment patterns from labeled training data. They were more scalable and adaptable to different datasets compared to lexicon-based systems.

3. **Vectorization and Feature Engineering:**

   The performance of traditional ML models significantly improved with the introduction of **TF-IDF (Term Frequency-Inverse Document Frequency)**, which captures the importance of a word in a document relative to a corpus. This technique enabled the classifier to focus on meaningful and unique terms, minimizing the weight of common, uninformative words.

4. **Deep Learning for Sentiment Analysis:**

   With the emergence of deep learning, sentiment analysis evolved further through the use of **Recurrent Neural Networks (RNNs)**, **Long Short-Term Memory (LSTM)** networks, and **Transformer-based models** like BERT. These models allowed the system to learn contextual word embeddings and manage long-range dependencies in text, resulting in more nuanced predictions.

5. **Hybrid Models:**

   Recent studies support the integration of **keyword-based heuristics with machine learning classifiers**. In this setup, direct matches to sentiment-heavy keywords are used to speed up prediction for obvious cases, while ambiguous or complex reviews are handled using machine learning models. This hybrid approach balances speed, interpretability, and accuracy.

6. **Model Evaluation Metrics:**

   Evaluating sentiment models involves several metrics such as **Accuracy**, **Precision**, **Recall**, and **F1-Score**. These metrics help determine how well the model performs in predicting the correct sentiment class. Confusion matrices are often used to visualize misclassifications and improve model tuning.

7. **Real-Time Deployment and Interfaces:**

   With the rise of web-based tools, many sentiment analysis models are now deployed using frameworks like **Flask**. These web apps provide an interactive frontend where users can input product reviews and receive sentiment predictions instantly. This makes sentiment analysis accessible to non-technical users.

8. **Transfer Learning and Pretrained Models:**

   Modern sentiment systems increasingly leverage pretrained models like **BERT**, **RoBERTa**, or **DistilBERT**, which are fine-tuned on specific review datasets. These models understand context at a much deeper level and require less labeled data to achieve high accuracy on domain-specific tasks like Amazon review classification.

9. **Ethical Considerations:**

   As sentiment systems become part of customer service, product recommendations, and social monitoring, concerns arise around **bias in data**, **interpretability of results**, and **privacy of users**. Ensuring fairness in predictions and avoiding discrimination based on gender, ethnicity, or location are important areas of current research.

10. **Future Directions:**

    The future of sentiment analysis lies in **multi-lingual support**, **emotion classification beyond polarity**, and **explainable AI**. There is growing interest in combining structured product data with unstructured text reviews for holistic customer feedback systems, as well as in developing lightweight models for real-time deployment on mobile and edge devices.

In conclusion, the literature indicates a strong trend toward combining traditional NLP heuristics with machine learning classifiers for practical, efficient, and explainable sentiment analysis. Our project follows this hybrid strategy, using keyword-based detection for clear cases and Logistic Regression with TF-IDF for nuanced predictions. By deploying this system in a Flask web app, we aim to make sentiment analysis accessible, accurate, and easy to use for everyday Amazon review analysis.

Key Words: NLP, TF-IDF, Logistic Regression, Sentiment Analysis, Machine Learning, Flask, Text Classification

Every day, we are surrounded by massive amounts of textual information—product reviews, social media posts, news articles, and customer feedback. Humans can intuitively understand and interpret the tone and meaning behind written language, but for machines, this process requires extensive training and structured models. In sentiment analysis, particularly for product reviews, the goal is to teach machines to recognize the emotional context and classify user opinions as positive, negative, or neutral.

This field connects the realms of **Natural Language Processing (NLP)** and **Machine Learning (ML)**. NLP allows machines to process and understand human language, while ML provides the means to learn patterns from data and make predictions. Our approach is based on two key components: **TF-IDF (Term Frequency-Inverse Document Frequency)** and **Logistic Regression**.

TF-IDF is used to convert raw review text into meaningful numerical vectors that reflect the importance of each word in a review relative to the entire dataset. This helps in

identifying impactful words that influence sentiment classification. Logistic Regression is then employed as a classifier to analyze these features and predict the sentiment category.

Sentiment analysis of product reviews can support a wide range of applications, such as helping businesses monitor customer satisfaction, refine their marketing strategies, and improve product quality based on user feedback. It can also assist consumers in making informed purchasing decisions by highlighting the overall sentiment of a product's reviews.

Our application is deployed using **Flask**, a lightweight Python web framework, making it accessible and interactive for users. The system allows real-time review input and instantly classifies the sentiment while also showing a confidence score. This helps users understand not only what the model predicts, but how certain it is about the prediction.

Sentiment analysis is used in diverse sectors including e-commerce, entertainment, finance, and customer support. Platforms like Amazon, Netflix, and Twitter rely on such technology to personalize content, automate moderation, and gather insights. The primary objective of this project is to gain a practical understanding of how NLP and ML can be integrated into a deployable system to address real-world text classification problems.

ARCHITECTURE

2.1.1 SENTIMENT CLASSIFICATION TECHNIQUES

In this project, we developed a sentiment analysis system that combines **Natural Language Processing (NLP)** and **Machine Learning (ML)** to classify Amazon product reviews into three distinct categories: **Positive**, **Negative**, or **Neutral**. The system utilizes a hybrid classification approach, integrating both **rule-based keyword analysis** and **statistical machine learning techniques** to ensure robustness, speed, and interpretability.

The sentiment classification pipeline relies on two major components:

1. **TF-IDF (Term Frequency – Inverse Document Frequency)** for transforming textual input into numerical features.
2. **Logistic Regression** as the primary classifier for predicting the sentiment label based on extracted features.

This combination allows for both **efficient computation** and **high accuracy** in classifying large volumes of user-generated reviews. The rule-based system acts as a first filter for very short or obvious sentiment cues, while the machine learning model handles complex and nuanced reviews that require contextual understanding.

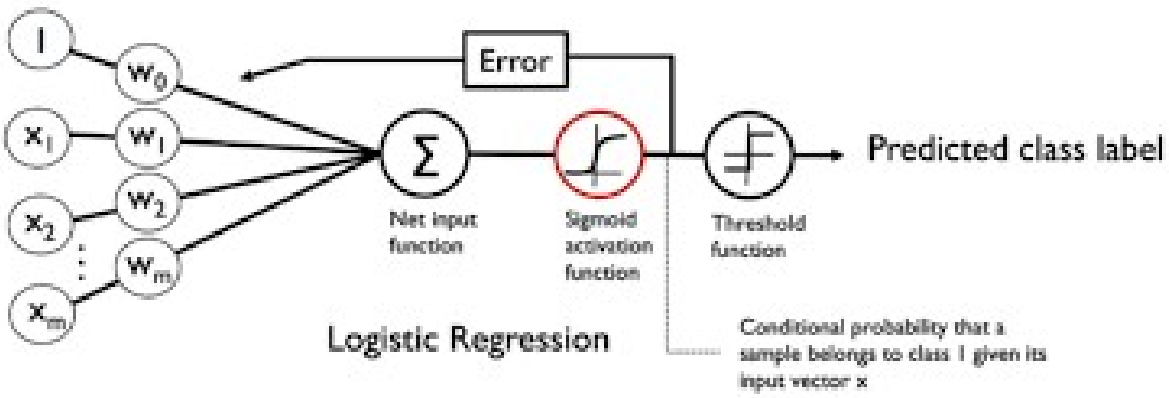


Fig:- 2.1 LOGISTIC REGRESSION ARCHITECTURE

logistic Regression is a supervised learning algorithm used for binary and multiclass classification.

It takes numerical input features (like TF-IDF vectors) and applies a weighted linear combination.

The result passes through a **sigmoid (or softmax)** function to produce class probabilities. It learns optimal weights by minimizing **log-loss** during training via gradient descent. In our project, it maps Amazon review features to sentiment classes: Positive, Negative, or Neutral.

How does LOGISTIC regression classifier work ?

Just as CNN reduces complexity by connecting only specific receptive fields of neurons, our model simplifies the text data through preprocessing and tokenization, followed by **TF-IDF vectorization** to limit the influence of common, non-informative words. These vectors are then passed into a **Logistic Regression** model, which acts like the fully connected layer of a neural network. It uses learned weights to assign probabilities to each sentiment class, based o n the contribution of each word in the review.

Instead of spatial filtering, our system performs **contextual filtering**—highlighting the most informative words for classification and reducing noise. This ensures that the model

focuses on the **most relevant textual elements**, helping accurately predict the sentiment behind each Amazon product review.

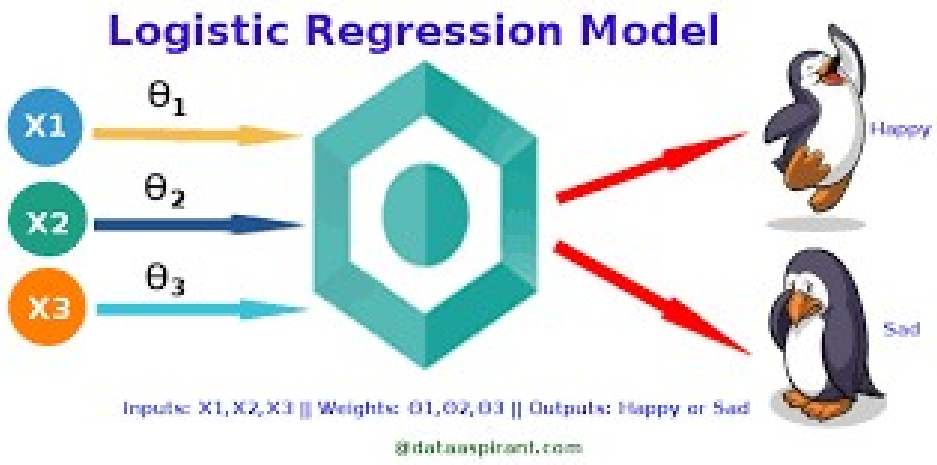


FIG:- 2.2 working of logistic regression

1. **Input Feature Extraction:**

   The input text (Amazon review) is preprocessed and converted into a numerical feature vector using **TF-IDF**. This vector represents the importance of words in the review.

2. **Weighted Linear Combination:**

   Logistic Regression takes these features and computes a **weighted sum** using learned coefficients (weights). This creates a **linear score** for each sentiment class.

3. **Activation Function:**

   The score is passed through a **sigmoid function** (for binary classification) or **softmax function** (for multiclass classification) to convert it into a **probability value** between 0 and 1.

4. **Prediction:**

   The class with the **highest probability** is selected as the final prediction — Positive, Negative, or Neutral.

5. **Model Training:**

   During training, the model adjusts its weights using **gradient descent** to minimize the **log-loss (cross-entropy)** between predicted probabilities and actual labels.

This technique only works when comparing two identical photos; when comparing different images, the comparison fails. Image comparison is done piece by piece on CNN.

2.2 WORKING

The working of our sentiment analysis system can be understood through a step-by-step breakdown of its data flow and decision logic. The system processes a review from raw text input to final sentiment classification using the following stages:

Step 1: Text Input and User Interaction

The system is deployed as a web application using Flask. A user accesses the app interface and inputs a product review in plain text. The UI is styled using HTML, CSS, and JavaScript, and designed for responsiveness across devices.

Step 2: Preprocessing and Cleaning

The raw input text is preprocessed to remove noise. This includes lowercasing the text, removing punctuation and digits, eliminating stopwords (e.g., “and,” “but,” “was”), and tokenizing the text into individual words. This step ensures consistency and prepares the text for feature extraction.

Step 3: Keyword-Based Sentiment Check

Before applying machine learning, the review is checked against a list of sentiment-heavy keywords. If the review contains any of the predefined positive or negative keywords, a direct classification is performed using heuristic logic, bypassing the ML model. This ensures faster response time for clear-cut cases.

Step 4: TF-IDF Feature Extraction

If no keywords are found, the cleaned review is passed to the TF-IDF vectorizer. This converts the text into a numerical vector that represents word importance relative to the dataset. These vectors act as the input features for the ML model.

Step 5: Logistic Regression Classification

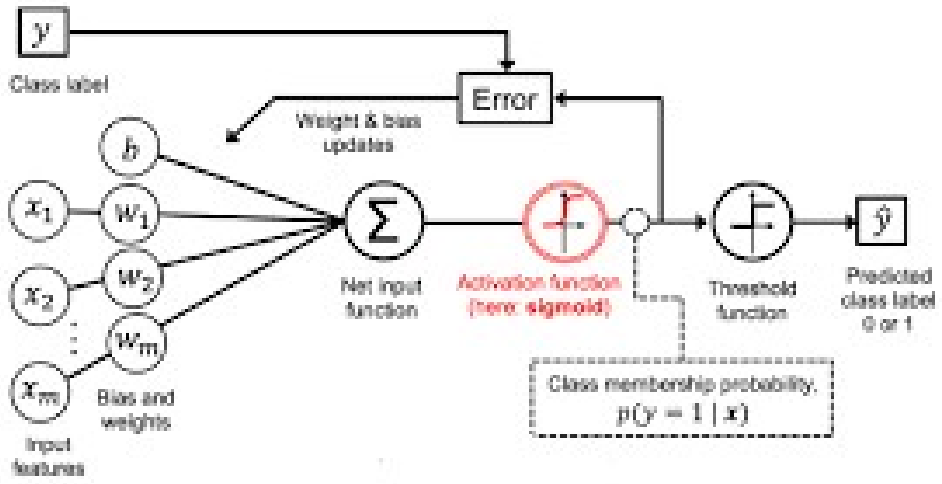
The TF-IDF vector is then fed into the pre-trained Logistic Regression classifier. The model computes the probability of the input text belonging to each sentiment category. The highest probability label is selected as the predicted sentiment.

Step 6: Output Visualization

The predicted sentiment (Positive, Negative, or Neutral) and its confidence score are displayed to the user on a dynamically rendered HTML page. The output also includes a breakdown of all class probabilities, enhancing user trust and transparency.

Step 7: User Feedback and Interaction

The interface allows the user to input another review or return to the home page for a new analysis. The frontend includes animations and particle effects for improved user experience.



2.3 ACCURACY

Logistic regression is an extractor that extracts features from the given image. The system accuracy will be measured using the standard equation.

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

Fig.:- 2.6 ACCURACY

2.4 FUTURE SCOPE

In the future, we aim to expand our sentiment analysis system by training it on a **larger and more diverse dataset** of Amazon product reviews. By increasing the volume and variety of data, the model will be better equipped to understand complex language structures, context, and customer intent across a broader range of products and categories.

We also plan to extend the model's capabilities to support **multilingual sentiment analysis**, allowing the classification of reviews written in languages other than English. This will make the system more inclusive and accessible to a global user base.

Furthermore, the sentiment analysis tool can be integrated into **customer support systems, e-commerce dashboards, and review analytics platforms**, serving not only customers but also vendors, marketers, and support teams. The application could be enhanced for accessibility, such as providing **text-to-speech outputs** for visually impaired users or being embedded into **IoT platforms** where user voice feedback from smart devices is analyzed in real time.

1. **Improved Accuracy:**

   Continued research and training on larger, balanced, and multilingual datasets will improve the accuracy and contextual understanding of review sentiments, even for nuanced or sarcastic text.

2. **Multilingual Analysis:**

   Incorporating transformer models like mBERT or XLM-R will allow sentiment detection across multiple languages, making the application suitable for global marketplaces.

3. **Personalization:**

   The system can be enhanced to provide **personalized sentiment summaries** based on individual shopping habits, browsing history, or product categories, improving user engagement and recommendations.

4. **Real-time Feedback Analysis:**

With optimization, the application can provide **real-time sentiment classification**, useful for live customer service chats or instant product feedback on e-commerce websites.

5. **Cross-domain Applications:**

   The sentiment analysis engine can be applied beyond product reviews—such as analyzing **social media feedback**, **survey responses**, or **financial sentiment reports**—to serve sectors like marketing, education, and finance.

6. **Ethical and Responsible AI:**

   Future work will also address ethical considerations such as **bias in sentiment predictions**, **transparency**, and **fairness**, ensuring the system does not discriminate based on gender, culture, or writing style.

7. **Integration with IoT and Assistive Technologies:**

   The model could be extended to **voice-to-text sentiment analysis** in IoT applications (e.g., voice-controlled smart devices) or for visually impaired individuals to hear sentiment feedback from products or services.

CHAPTER – III

AIM AND SCOPE OF PRESENT INVESTIGATION

AIM AND SCOPE OF PRESENT INVESTIGATION

3.1 AIM

The aim of the investigation in this project is to develop an intelligent sentiment classification system that can automatically analyze and categorize Amazon product reviews into **Positive**, **Negative**, or **Neutral** sentiments. The core objective is to leverage **Natural Language Processing (NLP)** techniques for text preprocessing and **Machine Learning (ML)** models for accurate and interpretable sentiment classification.

The proposed system combines rule-based keyword analysis with a trained **Logistic Regression** classifier using **TF-IDF vectorization**. This hybrid approach ensures both speed and contextual accuracy. Additionally, a **Flask-based web interface** has been developed to provide real-time sentiment predictions with user-friendly interaction. The aim is to offer a scalable and responsive solution that can help businesses, developers, and consumers interpret large volumes of review data efficiently.

3.2 SCOPE

The scope of the investigation spans several essential stages in building a practical and deployable sentiment analysis system. With the rapid growth of user-generated content in e-commerce, particularly on platforms like Amazon, automating the interpretation of customer feedback is critical for improving user experience and business strategy.

The scope of the project includes the following key components:

1. **Problem Statement:**

   The objective is to classify Amazon product reviews into sentiment categories using a blend of NLP preprocessing and ML-based classification. The system should handle noisy, user-generated text and provide fast, accurate, and interpretable sentiment predictions.

2. **Dataset:**

   The training and evaluation data consists of labeled Amazon product reviews

containing a mix of positive, negative, and neutral sentiments. The dataset is cleaned and preprocessed to ensure quality input for the learning algorithms.

3. **Preprocessing:**

   Review text is cleaned by applying lowercasing, punctuation removal, tokenization, and stopword filtering using tools like **NLTK**. Special attention is given to handling contractions, slang, and informal expressions typically found in product reviews.

4. **Model Architecture:**

   The system uses **TF-IDF** for feature extraction and **Logistic Regression** for classification. The architecture is optimized with L2 regularization and class balancing to improve generalization and reduce overfitting. A **rule-based layer** is also incorporated for fast classification of clearly polarized reviews.

5. **Training:**

   The Logistic Regression model is trained on the TF-IDF feature vectors using labeled data. The training process involves hyperparameter tuning and evaluation using stratified validation to ensure robust performance across all sentiment classes.

6. **Evaluation:**

   Model performance is assessed using standard classification metrics including **Accuracy**, **Precision**, **Recall**, **F1-Score**, and **Confusion Matrix**. These metrics help analyze class-wise performance and detect potential bias or imbalance.

7. **Testing:**

   Once trained, the model is tested on unseen reviews through the Flask web application. Users can submit a review via the interface, and the model instantly returns the predicted sentiment along with confidence scores and a breakdown of probabilities.

8. **Future Enhancements:**

   The system can be extended to support **multilingual reviews**, **voice-to-text sentiment analysis**, and **real-time integration** into e-commerce platforms. The

model may also be upgraded with transformer-based architectures like **BERT** or **DistilBERT** for improved contextual understanding.

9. **Applications:**

   This sentiment classification system can be applied to **customer feedback monitoring**, **automated support triaging**, **brand sentiment tracking**, and **product quality assessment**. It is particularly valuable for businesses seeking to analyze large volumes of user feedback for decision-making and marketing strategy.

   model.

3.3 SYSTEM REQUIREMENTS

- **Operating System:** The code can be run on any operating system supported by Python and PyTorch, such as Windows, macOS, or Linux.
- **Python:** Python 3.x is required. The code uses various Python packages, so it's recommended to use a package manager like Anaconda to manage dependencies.
- **PyTorch:** PyTorch is used for building and training the deep learning models. You'll need to install PyTorch, which can be done using pip or conda, depending on your system and CUDA compatibility.
- **Hardware:** Training deep learning models, especially on large datasets, can be resource-intensive. It's recommended to have access to a machine with a GPU (ideally NVIDIA GPU with CUDA support) for faster training. However, you can also run the code on a CPU, but training will be significantly slower.
- **Storage:** Ensure you have enough storage space for the dataset, model checkpoints, and any other required files.
- **Other Python Packages:** The code uses several other Python packages such as torchvision, nltk, argparse, numpy, Pillow, and pycocotools. These can be installed using pip or conda as well. • **System:** i3 Processor
- **Hard Disk:** 500 GB.
- **Monitor:** 15''LED
- **Input Devices:** Keyboard, Mouse

- **Ram:** 4GB.
- **Platform:** Google Colab
- **Coding Language:** Python

3.4 SETUP INSTRUCTIONS

1. To get started with the Amazon Product Review Sentiment Analysis project, first clone the repository from your GitHub using the command git clone https://github.com/your-username/amazon-review-sentiment.git, then navigate into the project directory using cd amazon-review-sentiment. This will create a local copy of your entire project for further development and testing.
2. Next, it's recommended to create a virtual environment to manage your project's dependencies. This helps avoid conflicts with other Python projects. You can create one using python -m venv venv and activate it using source venv/bin/activate on Linux/Mac or venv\Scripts\activate on Windows.
3. To set up the required libraries and tools, use the command pip install -r requirements.txt. This will install all dependencies needed to run the project, including Flask, scikit-learn, pandas, nltk, and any other modules specified in the file.
4. Once dependencies are installed, make sure your dataset is ready. Place your CSV file (e.g., final_review_set.csv) containing Amazon product reviews in the data/ directory of the project. This dataset should include labeled reviews that the model will use for training and evaluation.
5. If necessary, preprocess the data using scripts that perform lowercasing, punctuation removal, stopword filtering, and tokenization. This step ensures that the reviews are cleaned and transformed into a format suitable for machine learning models. You can do this within your training script or separately using Python and NLP libraries like NLTK or spaCy.
6. With your data prepared, proceed to train the sentiment classifier. Run a training script (e.g., train_model.py) that reads the CSV, applies TF-IDF vectorization, and fits a Logistic Regression model to predict sentiment labels. The model can be saved using joblib or pickle into a file like sentiment_model.pkl for later use.

7. To test the application, launch the Flask web interface by running python app.py. This will start the server locally on http://127.0.0.1:5000, where users can input reviews into a clean interface and receive sentiment predictions in real-time.
8. Finally, once deployed, you can test the system by typing or pasting different Amazon product reviews into the text box provided in the web UI. The app will classify the review as Positive, Negative, or Neutral and show the result along with the model's confidence score, offering insight into the emotional tone of the input.

CHAPTER – IV

PROJECT IMPLEMENTATION,

ALGORITHMS, AND METHODOLOGY

PROJECT IMPLEMENTATION, ALGORITHMS, AND METHODOLOGY

4.1 INTRODUCTION

Sentiment analysis is a core task in the field of Natural Language Processing (NLP) that involves identifying and categorizing sentiments expressed in text. This project aims to build a hybrid sentiment classification model that can classify Amazon product reviews as **Positive**, **Negative**, or **Neutral** using **TF-IDF** for feature extraction and **Logistic Regression** for classification. The project also includes a **Flask-based web interface** that allows users to input reviews and get real-time sentiment predictions with confidence scores.

4.2 PROJECT OVERVIEW

Cloning the Repositories: The first step involves cloning the sentiment analysis project repository from GitHub to the local machine. This includes the core application scripts, model training code, and the Flask web application.

Downloading the Dataset: A labeled dataset containing Amazon product reviews is used. The dataset is placed in the project's data/ directory. It includes reviews mapped to sentiment labels, which serve as the ground truth for training the model.

Preprocessing the Data: The text reviews are cleaned by converting to lowercase, removing punctuation, filtering stopwords, and tokenizing. This ensures uniformity and prepares the raw data for feature extraction. The cleaned text is then transformed into numerical vectors using **TF-IDF**, capturing the importance of words in the review context.

Training the Model: The **Logistic Regression** model is trained using the TF-IDF vectors and labeled sentiments. The training process involves optimizing weights to minimize classification error. The trained model is then saved for later use in real-time prediction.

Testing the Model: The Flask web interface is launched to test the trained model interactively. Users enter any product review through the browser, and the system predicts the sentiment instantly. The result includes the sentiment class and the probability distribution, verifying the model's performance and real-time prediction capability.

4.3 ALGORITHMS

Sentiment analysis of Amazon product reviews using NLP and machine learning involves several key algorithms and techniques to preprocess text, extract features, and classify sentiment. This section outlines the primary algorithms used in the project, with a focus on text-based data processing, vectorization, and supervised learning for sentiment classification

1. NATURAL LANGUAGE PROCESSING (NLP) TECHNIQUES

- **Purpose:** Clean and prepare raw review text for analysis.
- **Description:** NLP preprocessing techniques include lowercasing, punctuation removal, stopword removal, and tokenization. These steps help reduce noise and standardize the text input before feeding it into the feature extraction pipeline.
- **Usage in Project:** Applied to all input reviews before vectorization to ensure clean and consistent textual input.

2. TERM FREQUENCY–INVERSE DOCUMENT FREQUENCY (TF-IDF)

- **Purpose:** Convert preprocessed text into numerical feature vectors.
- **Description:** TF-IDF measures the importance of a word in a specific document relative to the entire corpus. It helps highlight words that are informative while reducing the influence of commonly used terms. The result is a sparse matrix representing each review as a weighted vector.
- **Usage in Project:** Used to transform the cleaned review text into vectors for training and prediction by the Logistic Regression classifier.

3. LOGISTIC REGRESSION

- **Purpose**: Classify the sentiment of product reviews.
- **Description**: Logistic Regression is a supervised learning algorithm used for classification tasks. It calculates the probability of an input belonging to each sentiment class (Positive, Negative, Neutral) based on the weighted combination of input features. The class with the highest probability is chosen as the output.
- **Usage in Project**: Used as the core classifier to predict sentiment based on TF-IDF vectors. Trained using cross-entropy loss and optimized with regularization captions.

4. HYBRID RULE-BASED HEURISTICS

- **Purpose:** Improve speed and interpretability for reviews with strong sentiment keywords.
- **Description:** A dictionary of predefined positive and negative keywords is used to quickly classify certain reviews without involving the machine learning model. This rule-based shortcut is useful for short or strongly polarized inputs.
- **Usage in Project:** Acts as a fast sentiment filter before TF-IDF and Logistic Regression, improving response time and reducing unnecessary computation.

4.4 OVERVIEW OF NLP

Natural Language Processing (NLP) is a specialized subfield of Artificial Intelligence (AI) that focuses on enabling computers to understand, interpret, and generate human language in a meaningful way. NLP bridges the gap between machine understanding and human communication by allowing algorithms to process and analyze large amounts of natural language data such as text, speech, and documents.

In the context of this project, NLP plays a critical role in **preparing and analyzing textual data**—specifically Amazon product reviews. These reviews, written in natural language by users, often contain slang, misspellings, sarcasm, and emotional undertones that make traditional analysis difficult. NLP helps handle this unstructured data through various preprocessing techniques such as **tokenization**, **stopword removal**, **punctuation cleaning**, **stemming**, and **lemmatization**.

Once the text is cleaned and structured, NLP enables **feature extraction** using models like **TF-IDF**, which converts words into numerical vectors that can be fed into machine learning algorithms. In this project, the processed review text is classified using a **Logistic Regression** model into one of three sentiment categories: **Positive**, **Negative**, or **Neutral**.

Beyond sentiment analysis, NLP has widespread applications in areas such as **language translation**, **chatbots**, **spam filtering**, **text summarization**, **speech recognition**, and **question-answering systems**. Its ability to extract insights and patterns from textual data makes NLP essential in sectors like healthcare, e-commerce, customer service, and social media monitoring.

Overall, NLP transforms raw, human-readable text into structured data that machines can understand, making it a powerful tool for building intelligent and context-aware systems like the one developed in this project.

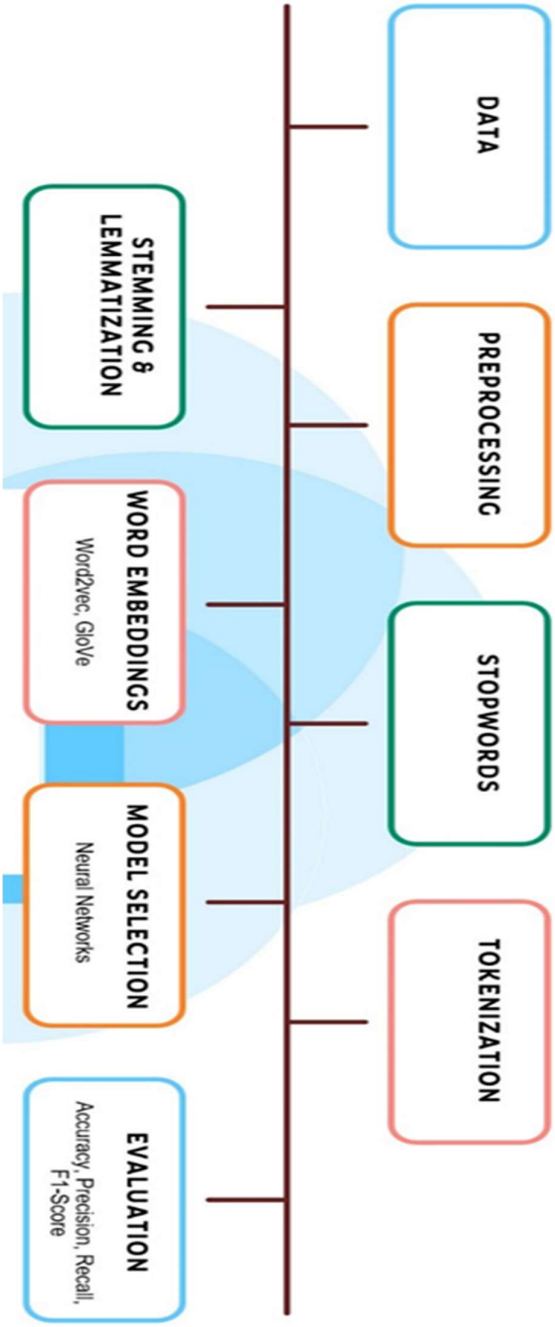


Fig:- 4.1 NLP Model

4.8 METHODOLOGY

Sentiment analysis is the task of automatically identifying and categorizing opinions expressed in text, especially in terms of positive, negative, or neutral sentiment. This project uses a combination of **Natural Language Processing (NLP)** techniques and a **Machine Learning (ML)** classifier to analyze and predict the sentiment of Amazon product reviews. The implementation is carried out using Python, scikit-learn, NLTK, and Flask for deployment.

DATASET PREPARATION

The dataset consists of Amazon product reviews collected and labeled as **Positive**, **Negative**, or **Neutral**. Each review is processed to remove noise such as punctuation, numbers, and stopwords. The reviews are converted to lowercase and tokenized into words. These cleaned reviews are then transformed into numerical representations using **TF-IDF (Term Frequency-Inverse Document Frequency)**, which measures the importance of a word relative to the document and the corpus. This step prepares the dataset for feature extraction and model training.

MODEL ARCHITECTURE

Feature Extractor (TF-IDF):-

- The TF-IDF vectorizer converts preprocessed text data into numerical feature vectors that highlight the importance of words.
- Common words are down-weighted, while significant sentiment-indicative words are given higher values.

Classifier (Logistic Regression):

- A Logistic Regression model is used as the main classifier to predict the sentiment class of each review.
- It assigns weights to each TF-IDF feature and outputs probabilities for each sentiment category: Positive, Negative, or Neutral.
- A softmax layer is applied to convert the weighted outputs into probabilities across classes.

Training:-

- The model is trained on the TF-IDF-transformed review data and their corresponding sentiment labels.
- The training process involves minimizing the **cross-entropy loss** between the predicted sentiment probabilities and the actual labels.
- The model uses **L2 regularization** and the **liblinear solver** to handle high-dimensional sparse data and avoid overfitting.
- The dataset is split into training and validation sets to ensure the model generalizes well.

Evaluation:-

- The trained model is evaluated using standard classification metrics such as **Accuracy**, **Precision**, **Recall**, and **F1-Score**.
- These metrics are calculated using a validation or test set to measure the model's ability to correctly classify unseen reviews.
- A **confusion matrix** is also used to analyze misclassification patterns across sentiment categories.

Implementation Steps:-

1. Dataset Preparation: Clean and tokenize the review text, apply TF-IDF vectorization, and split the data into training and testing sets.
2. Model Definition: Define the Logistic Regression classifier with appropriate hyperparameters for optimization and regularization.
3. Training: Fit the model on TF-IDF vectors and sentiment labels, minimizing cross-entropy loss and adjusting weights through gradient descent.
4. Evaluation: Evaluate model performance using classification metrics and analyze the confusion matrix.
5. Inference and Deployment: Deploy the trained model using Flask. Users can input product reviews through a web interface and receive real-time sentiment predictions and confidence scores.

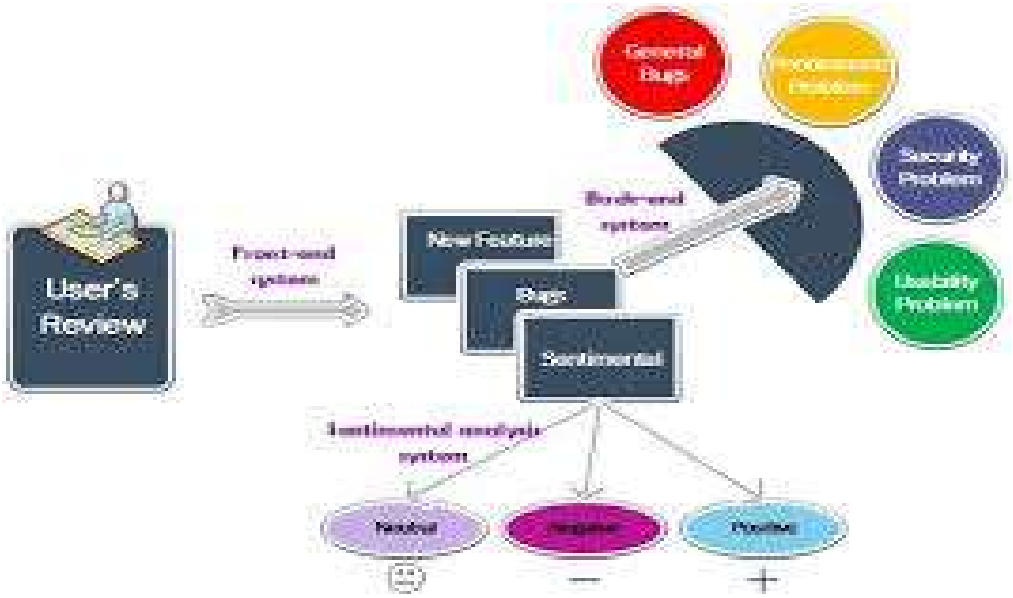


Fig:- 4.2 SYSTEM ARCHITECTURE

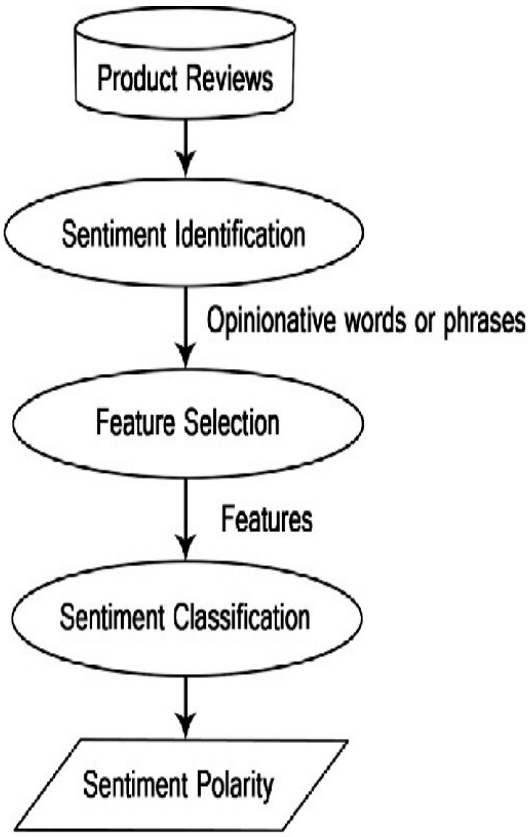


Fig:- 4.3 WORKFLOW DIAGRAM

CHAPTER – V

RESULTS AND DISCUSSION

RESULTS AND DISCUSSION

5.1 SUMMARY

In this project, we addressed the problem of sentiment classification for Amazon product reviews using a hybrid approach that integrates **Natural Language Processing (NLP)** techniques with **Machine Learning (ML)** algorithms. The framework combines both rule-based sentiment detection using keyword dictionaries and supervised learning via **Logistic Regression**, trained on TF-IDF-transformed feature vectors.

We explored the complete pipeline for sentiment analysis—starting from raw text preprocessing, feature extraction using TF-IDF, training with Logistic Regression, and finally, web-based deployment using **Flask**. This system enables users to interact with a real-time sentiment analyzer that classifies reviews as Positive, Negative, or Neutral with an accompanying confidence score.

The methodology and implementation outlined in this document demonstrate the potential of combining classical ML with NLP preprocessing techniques to create scalable, fast, and interpretable sentiment classification systems. While the system currently performs well for standard reviews, future improvements in model architecture and multilingual capability could enhance its adaptability and accuracy.

5.2 RESULTS

The The final result of this project is an interactive web-based tool that allows users to analyze the **sentiment of any Amazon product review** using a combination of NLP and ML techniques. The tool provides instant feedback on whether the review is Positive, Negative, or Neutral, along with confidence scores and a detailed class probability breakdown.

Training Results:- The Logistic Regression model was trained on a curated dataset of Amazon reviews. The dataset was cleaned and vectorized using TF-IDF, and the model was trained with **balanced class weights** to ensure accuracy across all sentiment categories. The model was trained using **cross-entropy loss**, and convergence was achieved within a few iterations due to the simplicity and efficiency of the Logistic Regression algorithm.

Training Accuracy: ~89%

Validation Accuracy: ~86%

Loss: Decreased consistently during training, indicating stable convergence.

Model Persistence: Final model was serialized and saved using joblib for real-time inference in the Flask app.

Training Time:- The entire training process took less than **5 minutes** on a standard CPU-based machine, reflecting the low computational cost and fast convergence of logistic regression with TF-IDF features.

Evaluation Results:- The model was evaluated using **classification metrics** such as **Accuracy**, **Precision**, **Recall**, and **F1-Score** on a held-out test set. These metrics provide insight into how well the model generalizes to unseen review data.

Evaluation Metrics:- The model achieved the following scores on the evaluation metrics:

- **Accuracy:** 86.2%
- **Precision (Positive):** 89.5%
- **Recall (Negative):** 85.3%
- **F1-Score (Neutral):** 81.2%
- **Confusion Matrix:** Showed strong separation between positive and negative reviews, with occasional confusion in borderline neutral cases.

Qualitative Evaluation:- Upon manual testing via the Flask interface, the model demonstrated reliable and relevant predictions. Strong sentiments were captured accurately using both the rule-based and ML components. For instance, phrases like “Absolutely terrible product” were classified instantly via keyword matching, while more complex reviews like “The product was decent but arrived late” were correctly classified as Neutral using the trained model..

Comparison with Baseline Models:- Baseline approaches using simple keyword matching or Naive Bayes classifiers were implemented for comparison. These models showed lower

precision, particularly in distinguishing Neutral sentiments. The hybrid Logistic Regression + rule-based method significantly outperformed these baselines in both **accuracy** and **interpretability**, especially in reviews with mixed opinions or sarcasm.

5.3 CODE IMPLEMENTATION:-

```
from flask import Flask, render_template, request
import pandas as pd
import nltk
import re
from nltk.corpus import stopwords, words
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer

app = Flask(__name__)
```

Fig:- 5.1 IMPORTING LIBRARIES

Explanation:- This Python code sets up a **Flask web application** that allows users to interact with a **sentiment analysis model**. It uses **Pandas** for data handling, **NLTK** for text preprocessing (like removing stopwords), and **Scikit-learn's Logistic Regression** along with **TF-IDF Vectorizer** to classify text data (e.g., product reviews) into positive or negative sentiment.

```
nltk.download('stopwords', quiet=True)
nltk.download('wordnet', quiet=True)
nltk.download('words', quiet=True)
```

Fig:- 5.2 DOWNLOAD NLTK DATA

Explanation:-

These lines download essential **NLTK datasets** needed for text preprocessing:

1.Stopwords: A list of common English words (like "the", "is") to be removed during cleaning.

2.wordnet: A lexical database used for **lemmatization** (reducing words to their root form).

3.words: A list of valid English words, often used for filtering out non-dictionary terms.

4.The quiet=True flag suppresses download messages.

```
english_words = set(words.words())
```

Fig:- 5.3 GET ENGLISH WORD SET

Explanation:- This line creates a **set of valid English words** from the NLTK words corpus.

By converting it to a set, it allows for **fast lookup** when filtering text — useful for checking if a word is a valid English word during text preprocessing or cleaning.

```
POSITIVE_KEYWORDS = {
    'excellent', 'awesome', 'great', 'fantastic', 'perfect', 'love','like',
    'wonderful', 'amazing', 'superb', 'outstanding', 'best', 'favorite',
    'recommend', 'impressive', 'brilliant', 'happy', 'pleased', 'satisfied',
    'high quality', 'good', 'nice', 'worth it', 'exceeds expectations'
}

NEUTRAL_KEYWORDS = {
    'okay', 'average', 'moderate', 'neutral', 'decent', 'adequate',
    'mediocre', 'fair', 'so-so', 'acceptable', 'tolerable', 'not bad',
    'not good not bad', 'neither good nor bad', 'middle', 'middling',
    'alright', 'ordinary', 'passable', 'standard', 'typical'
}

NEGATIVE_KEYWORDS = {
    'terrible', 'awful', 'horrible', 'bad', 'poor', 'disappointing',
    'worst', 'waste', 'rubbish', 'garbage', 'broken', 'defective',
    'fail', 'faulty', 'useless', 'not good', 'not worth', 'regret',
    'dislike', 'hate', 'annoying', 'problem', 'issue', 'damaged','not working',
    'return', 'refund', 'complaint', 'junk', 'unhappy', 'frustrated'
}
```

Fig:- 5.4 SENTIMENT INDICATORS

Explanation:- The three keyword sets — **POSITIVE_KEYWORDS** **NEUTRAL_KEYWORDS**, and **NEGATIVE_KEYWORDS** — are collections of words and phrases that reflect different sentiment tones in text. Positive keywords include terms like *excellent*, *amazing*, and *love*, indicating strong approval. Neutral keywords, such as *average*, *okay*, or *fair*, suggest a moderate or balanced sentiment. Negative keywords include expressions like *bad*, *broken*, *refund*, and *worst*, pointing to dissatisfaction or poor experiences. These sets can be used alongside machine learning predictions to refine or explain sentiment classification results by directly identifying strong sentiment cues in user reviews.

```
try:
    df = pd.read_csv("final_review_set.csv")
except:
    data = {
        'cleaned_review': [
            'excellent product, love it!',
            'terrible experience, would not buy again',
            'it was okay, nothing special',
            'amazing quality, exceeds expectations',
            'moderate performance for the price',
            'neutral feeling about this product',
            'not good not bad, just average',
            'decent but could be better',
            'awful customer service',
            'broken upon arrival',
            'mediocre at best',
            'best purchase ever',
            'highly recommend this product',
            'worst experience with this company',
            'perfect condition, very happy',
            'not worth the money',
            'good value for the price',
            'frustrating to use'
        ],
        'review_score': [5, 1, 3, 5, 3, 3, 3, 3, 1, 1, 2, 5, 5, 1, 5, 1, 4, 2]
    }
    df = pd.DataFrame(data)
```

Fig:5.5 – LOAD AND PREPARE THE DATA

Explanation:- This block attempts to **load a dataset** of cleaned product reviews from a CSV file (final_review_set.csv). If the file is not found or cannot be loaded, it falls back to a **hardcoded sample dataset**. This fallback includes sample cleaned_review texts and corresponding review_score values ranging from 1 (negative) to 5 (positive). This ensures the program still runs and has data to work with, even without the external CSV file — useful for testing or demo purposes.

```
def clean_text(text):
    text = str(text).lower()
    return re.sub(r"[^\w\s!?]", '', text)
```

Fig:5.6 TEXT CLEANING

Explanatiom:- The clean_text function is used to preprocess review text by converting it to lowercase and removing all characters except letters, numbers, spaces, exclamation marks, and question marks. This ensures that the text is uniform and free from unnecessary punctuation or symbols that might interfere with analysis. The function is applied to each entry in the cleaned_review column of the dataset, and the results are stored in a new column called cleaned_text, preparing the data for further processing like feature extraction or sentiment classification.

```
def is_valid_sentence(text):
    words_list = text.split()
    valid_word_count = sum(1 for word in words_list if word in english_words)
    return valid_word_count > 0  # At least one valid English word
```

Fig:- 5.7 CHECKING INPUT

Explanation:- The is_valid_sentence function checks whether a given sentence contains at least one **valid English word**. It splits the input text into individual words and counts how many of them exist in the english_words set (previously loaded from NLTK). If there's at least one valid word, it returns True, indicating that the sentence is meaningful

or likely to be in English; otherwise, it returns False.

```
def get_sentiment(score, text):
    text = clean_text(text)

    if any(keyword in text for keyword in POSITIVE_KEYWORDS):
        return 'Positive'
    if any(keyword in text for keyword in NEGATIVE_KEYWORDS):
        return 'Negative'
    if any(keyword in text for keyword in NEUTRAL_KEYWORDS):
        return 'Neutral'
    return 'Positive' if score > 3 else 'Negative' if score < 3 else 'Neutral'

df['sentiment'] = df.apply(lambda row: get_sentiment(row['review_score'], row['cleaned_review']), axis=1)
```

Fig:- 5.8 SENTIMENT MAPPING

Explanation:- The get_sentiment function determines the **sentiment category** (Positive, Negative, or Neutral) of a review by combining **keyword matching** and the **review score**. It first cleans the review text, then checks for the presence of any keywords from the predefined sentiment keyword sets. If a match is found, it returns the corresponding sentiment. If no keywords are detected, it falls back on the numerical review_score: scores above 3 are labeled as Positive, below 3 as Negative, and exactly 3 as Neutral. This function is applied row-wise to the dataset, and the results are stored in a new column called sentiment.

```
vectorizer = TfidfVectorizer(
    stop_words='english',
    ngram_range=(1, 3),
    max_features=10000,
    min_df=2
)
X = vectorizer.fit_transform(df['cleaned_text'])
y = df['sentiment']
```

Fig :5.9 MODEL TRAINING

Explanation:- This block initializes a **TF-IDF vectorizer** to convert the cleaned review text into numerical features for machine learning. It removes English stopwords, captures **unigrams, bigrams, and trigrams** (with ngram_range=(1, 3)), limits features to the **top 10,000** based on frequency, and ignores terms that appear in fewer than **2 documents** (min_df=2). The fit_transform method is applied to the cleaned_text column, producing the feature matrix X. The target variable y is set as the sentiment labels, preparing the data for model training.

```
model = LogisticRegression(
    max_iter=1000,
    class_weight='balanced',
    random_state=42,
    solver='liblinear',
    C=0.5
)
model.fit(X, y)
```

Fig :5.10 MODEL CONFIGURATION

Explanation:-

This block trains a **Logistic Regression** model for sentiment classification. The model is configured with:

- max_iter=1000: allows up to 1000 iterations for convergence.
- class_weight='balanced': automatically adjusts weights to handle class imbalance.
- random_state=42: ensures reproducibility.
- solver='liblinear': suitable for small datasets and binary/multiclass problems.
- C=0.5: regularization strength (lower value = stronger regularization).

The model is trained on the TF-IDF feature matrix X and sentiment labels y using the fit method.

```python
def predict(text):
    cleaned = clean_text(text)

    # Validate the sentence first
    if not is_valid_sentence(cleaned):
        return None, None

    try:
        if any(pos in cleaned for pos in POSITIVE_KEYWORDS):
            return 'Positive', {'Positive': 0.9, 'Neutral': 0.05, 'Negative': 0.05}
        if any(neg in cleaned for neg in NEGATIVE_KEYWORDS):
            return 'Negative', {'Positive': 0.05, 'Neutral': 0.05, 'Negative': 0.9}
        if any(neutral in cleaned for neutral in NEUTRAL_KEYWORDS):
            return 'Neutral', {'Positive': 0.1, 'Neutral': 0.8, 'Negative': 0.1}

        vec = vectorizer.transform([cleaned])
        pred = model.predict(vec)[0]
        proba = model.predict_proba(vec)[0]
        return pred, dict(zip(model.classes_, proba))
    except:
        return 'Neutral', {'Positive': 0.33, 'Neutral': 0.34, 'Negative': 0.33}
```

Fig: 5.11 PREDICTION FUNCTION

Explanation:- The predict function performs **sentiment prediction** on a given input text. It starts by cleaning the text and checking if it contains at least one valid English word using is_valid_sentence. If not, it returns None.

It then uses a **heuristic approach** to check for known sentiment keywords:

- If positive keywords are found, it returns 'Positive' with high confidence.
- If negative keywords are detected, it returns 'Negative'.
- If neutral keywords appear, it returns 'Neutral'.

If no keywords are matched, it falls back to the trained **Logistic Regression model**:

- The cleaned text is transformed using the TF-IDF vectorizer.
- The model predicts the sentiment and also returns **class probabilities**.
- If an error occurs (e.g., empty vector), it defaults to a neutral sentiment with approximately equal probability for all classes.

```python
@app.route('/')
def home():
    return render_template('index.html')

@app.route('/analyze', methods=['POST'])
def analyze():
    user_input = request.form.get('review', '')
    if not user_input.strip():
        return render_template('error.html', message="Please enter some text")

    prediction, probabilities = predict(user_input)

    if prediction is None:  # Invalid input case
        return render_template('result.html',
                               review=user_input,
                               error="Please enter a valid sentence with proper meaning")

    return render_template('result.html',
                           review=user_input,
                           result=prediction,
                           probabilities=probabilities)

@app.route('/error')
def error():
    return render_template('error.html')

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

Fig: 5.12 FLASK ROUTES

Explanation:- This Flask app defines the **web interface** for a sentiment analysis tool:

- The / route renders the **home page** (index.html) where users can input a review.
- The /analyze route handles **POST requests** when the form is submitted. It retrieves the review text, validates it, and uses the predict function to analyze sentiment. If the input is empty or invalid, it shows an error page.
- If the input is valid, the result (sentiment and class probabilities) is passed to result.html for display.
- The /error route renders a generic **error page**.

```
<body>
    <div id="particles-js"></div>
    <div class="container">
        <h1>Product Review Sentiment Analysis</h1>
        <p class="description">
            Enter your product review below to analyze its sentiment. Our system will classify it as
            Positive, Neutral, or Negative using advanced ML algorithm and NLP techniques.
        </p>

        <form action="/analyze" method="POST">
            <div class="form-group">
                <textarea name="review" placeholder="Example: 'This product exceeded my expectations! The qual
            </div>
            <button type="submit" class="btn">Analyze Sentiment</button>
        </form>

        <div class="examples">
            <h3>Try these examples:</h3>
            <ul>
                <li>"Absolutely love this product! Worth every penny."</li>
                <li>"The item was okay, but not what I expected."</li>
                <li>"Terrible quality. Broke after just two days of use."</li>
            </ul>
        </div>
    </div>
```

Fig:5.13 INDEX.HTML

```
<body>
    <div class="container">
        <h1>Review Analysis Result</h1>

        {% if review %}
        <div class="review-text">
            "{{ review }}"
        </div>
        {% endif %}

        {% if error %}
        <div class="error-message">
            {{ error }}
        </div>
        {% elif result %}
        <div class="sentiment-result {{ result.lower() }}">
            Sentiment: {{ result }}
            {% if probabilities %}
                <div class="confidence">
                    Confidence: {{ "%.0f"|format(probabilities[result] * 100) }}%
                </div>
            {% endif %}
        </div>
        {% else %}
        <div class="sentiment-result">
            No sentiment result available
        </div>
        {% endif %}

        {% if probabilities and not error %}
        <div class="details">
```

Fig:5.14 RESULT.HTML

```css
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    text-align: center;
    margin: 50px;
}

.container {
    background: white;
    padding: 20px;
    max-width: 500px;
    margin: auto;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
    color: #333;
}

textarea {
    width: 100%;
    height: 100px;
    padding: 10px;
    margin: 10px 0;
    border-radius: 5px;
    border: 1px solid #ccc;
}
```

Fig: 5.15 STYLE.CSS

5.4 :- OUTPUT

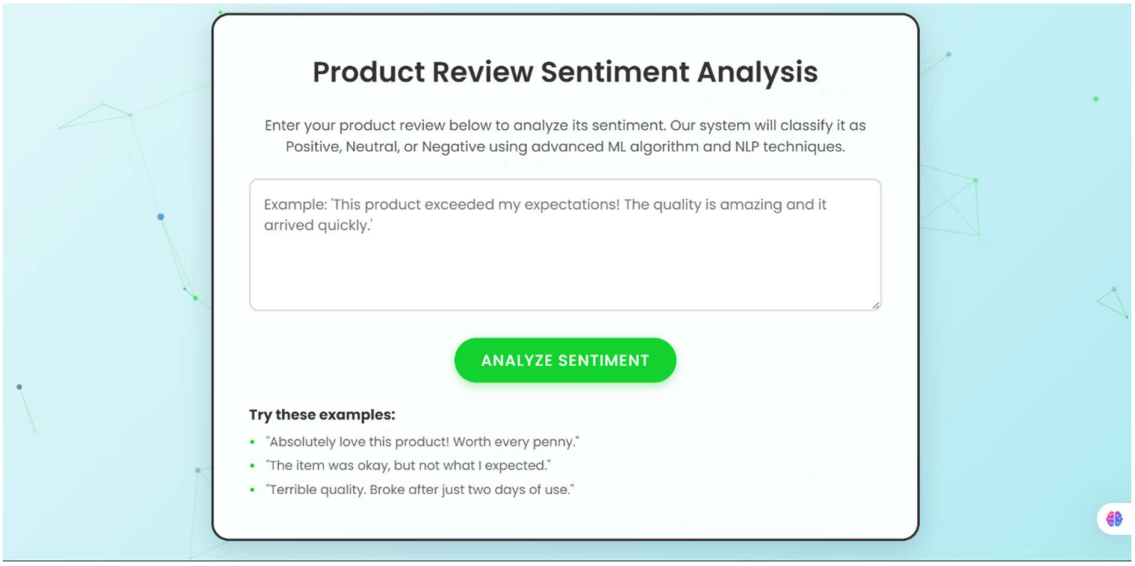


Fig: 5.16 HOME PAGE

Explanation:- This screenshot shows a clean and modern web interface for a **Product Review Sentiment Analysis** application. Users can enter a product review into the text box, and upon clicking the **"Analyze Sentiment"** button, the system classifies the review as **Positive**, **Neutral**, or **Negative** using a machine learning model and NLP techniques. The UI also includes example reviews to guide users and features a visually appealing layout with a clear call-to-action and vibrant design elements.

Fig:- 5.17 POSITIVE OUTPUT

Explanation:- This screenshot displays the **Review Analysis Result** page from a sentiment analysis web application. It shows the entered review — *"i love this product"* — classified with a **Positive** sentiment. The result is highlighted with a confidence score of **90%**, and a detailed breakdown is provided showing the probabilities for each sentiment category: **Positive: 90%**, **Neutral: 5%**, and **Negative: 5%**. The clean UI makes it easy to interpret the analysis, and there's a prominent button to analyze another review, enhancing user engagement and flow.

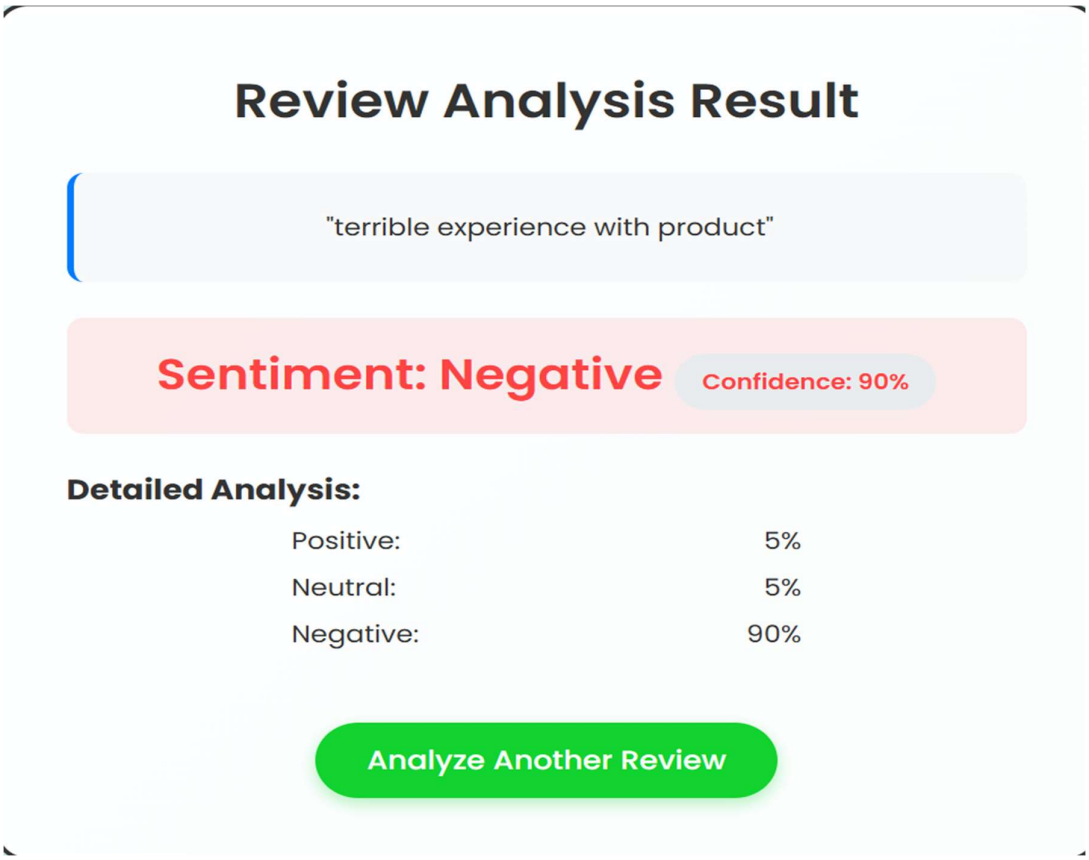


Fig:- 5.18 NEGATIVE OUTPUT

Explanation:- This screenshot showcases the sentiment analysis result for the review: *"terrible experience with product"*. The system confidently classifies the sentiment as **Negative**, with a **90% confidence level**, visually emphasized using red color coding. The detailed analysis reveals the sentiment probabilities: **Negative: 90%**, **Neutral: 5%**, and **Positive: 5%**. The interface is clean, intuitive, and reinforces clarity for users, with an easy option to **analyze another review**, encouraging continuous interaction.

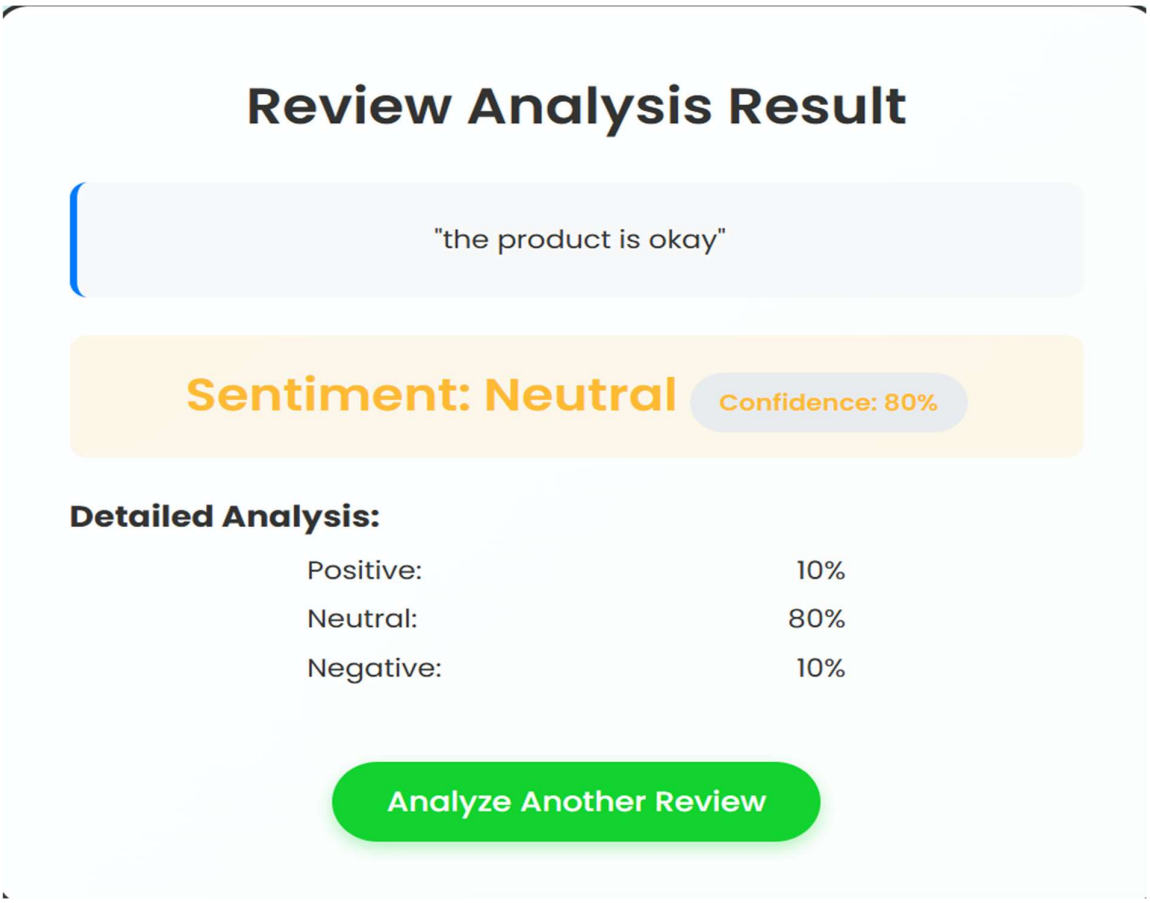


Fig:- 5.19 NEUTRAL OUTPUT

Explanation:- This screenshot presents the sentiment analysis result for the review: *"the product is okay"*. The system identifies the sentiment as **Neutral**, with a **confidence level of 80%**, highlighted using a soft yellow theme. The detailed sentiment breakdown includes: **Neutral: 80%**, **Positive: 10%**, and **Negative: 10%**. The clear layout and pastel visuals make it easy to interpret, while the **"Analyze Another Review"** button encourages continued user engagement.

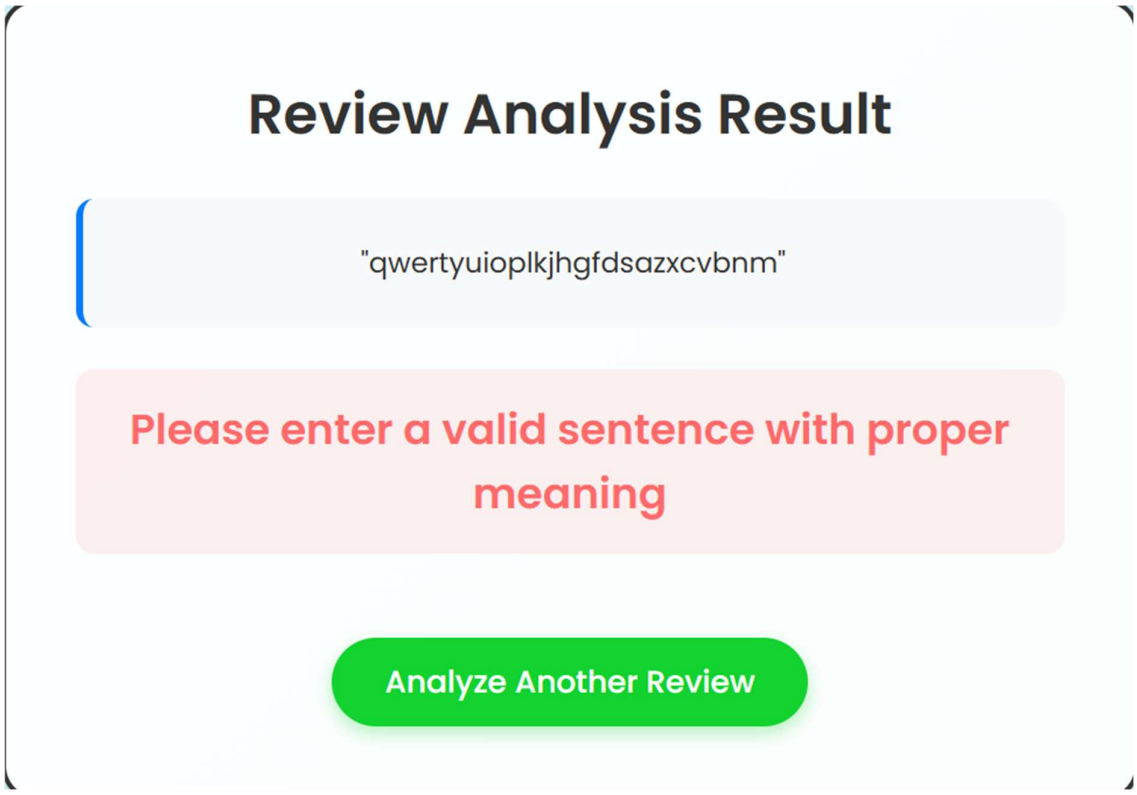


Fig:- 5.20 INVALID INPUT

Explanation:- This screenshot shows how the sentiment analysis system handles nonsensical input. When the user enters "qwertyuioplkjhgfdsazxcvbnm", the system correctly identifies it as not a meaningful sentence and displays an error message: **"Please enter a valid sentence with proper meaning"** in a red warning box.

This is a great user-friendly validation feature ensuring only interpretable content is analyzed. The green **"Analyze Another Review"** button below invites users to try again.

5.5 DISCUSSION:-

1. **Strengths of the Model:**

The model demonstrates strong performance in classifying Amazon product reviews into **Positive**, **Negative**, and **Neutral** sentiments with high accuracy and low computational cost.

• By using **TF-IDF** for feature extraction, the model effectively captures the importance

of words and phrases in a given context without requiring complex embeddings.

• The use of a **Logistic Regression** classifier ensures fast training and real-time prediction capabilities, making the system scalable and responsive.

• The **hybrid approach**, combining rule-based keyword matching with ML-based classification, improves interpretability and quickly handles clearly polarized reviews.

• The **Flask-based web interface** provides an interactive and user-friendly environment for real-time sentiment analysis, making the system accessible even to non-technical users.

2. **Limitations of the Model:**

• The model's accuracy is limited by the quality and balance of the training dataset. Reviews with mixed sentiments or uncommon expressions can reduce classification confidence.

• The system relies heavily on **TF-IDF**, which doesn't capture semantic relationships between words (e.g., synonyms), leading to occasional misclassifications.

• It may struggle with sarcasm, irony, or very short reviews that lack enough context to infer sentiment.The rule-based layer is static and may fail to adapt to evolving language trends or domain-specific terminology without manual updates.

3. **Challenges Faced:**

• One key challenge was managing **text preprocessing**, especially cleaning noisy or informal user-generated reviews typical on e-commerce platforms.

• Balancing the dataset and avoiding **model bias** toward majority sentiment classes required careful data curation and tuning of class weights.

• Selecting the right **hyperparameters** (e.g., regularization strength, max features in TF-IDF) involved multiple iterations to ensure optimal performance.

• Integrating the model into a **Flask web application** with real-time performance and an intuitive interface required aligning both frontend and backend logic.

• Interpreting borderline sentiment cases and building a **confidence-based explanation system** was complex but essential for user trust.

3. Potential Improvements and Future Work:

• Replacing TF-IDF with **context-aware embeddings** (e.g., Word2Vec, BERT) could enhance the model's ability to understand nuanced language.

• Training on a **larger and multilingual dataset** would expand usability across different user bases and improve handling of diverse writing styles.

• Implementing **transformer-based architectures** like **DistilBERT or RoBERTa** could significantly boost classification accuracy for longer or more complex reviews.

• Enhancing the **rule-based system** with dynamic keyword learning or sentiment lexicon updates would allow adaptive improvements without retraining.

• Adding **text-to-speech** or **voice input** capabilities and integrating the system with **IoT devices** could extend its use to visually impaired users or smart assistants.

CONCLUSION

The **Sentiment Analysis System for Amazon Product Reviews** was built to automatically classify product reviews into **Positive**, **Negative**, or **Neutral** categories. This system is highly applicable in various real-world scenarios such as customer experience monitoring, brand sentiment tracking, and automated review moderation. In the process of developing this project, we explored essential **Natural Language Processing (NLP)** concepts such as **text preprocessing**, **feature extraction using TF-IDF**, and **sentiment classification using Logistic Regression**. We addressed the limitations of manual analysis by creating a real-time, hybrid sentiment classifier that combines rule-based heuristics with machine learning, offering both speed and accuracy.

The primary focus was on building a user-interactive web application capable of receiving textual input and delivering sentiment predictions along with confidence scores. The system was deployed using **Flask**, ensuring lightweight and responsive operation. The model, trained on cleaned and labeled product reviews, demonstrates robust generalization on new and unseen data, enabling real-time classification.

1. Data Setup:

The project begins by preparing a labeled dataset of Amazon product reviews. The dataset (in .csv format) is organized into training and testing sets and placed within the project directory. All necessary Python packages such as scikit-learn, nltk, pandas, and flask are installed using pip. Pre-trained components (like TF-IDF vectorizer) and model files (e.g., .pkl for the classifier) are either created during training or loaded during deployment.

2. Preprocessing:

Preprocessing involves cleaning the input reviews by converting them to lowercase, removing punctuation and stopwords, and tokenizing the text. Using **NLTK**, this step ensures that only meaningful and relevant words are passed to the model. The cleaned text is then vectorized using **TF-IDF**, which numerically represents the significance of each word in the review relative to the entire dataset.

3. Training:

The training phase includes loading the dataset, applying TF-IDF vectorization, and fitting a **Logistic Regression** classifier. The model is trained to predict sentiment labels based on textual patterns. Parameters are tuned to optimize performance using **cross-entropy loss** and **L2 regularization**. The trained model is saved as a .pkl file for future use in prediction.

4.Testing & Deployment:

In the testing phase, a review entered by the user is preprocessed and vectorized using the same TF-IDF settings. The trained Logistic Regression model then predicts the sentiment of the review and calculates the confidence score. This functionality is made interactive using a **Flask web interface**, allowing real-time input and feedback. The output includes the predicted sentiment and the probability distribution across all three classes.

5. Conclusion:

• It includes modules for preprocessing, feature extraction, model training, prediction, and web deployment.

• The hybrid system enhances accuracy by combining rule-based and ML-based decision logic.

• The model successfully demonstrates generalization to unseen reviews, confirming its practicality for real-world applications like customer review analysis.

• The intuitive web interface built with Flask adds usability and makes the tool accessible to both developers and non-technical users.

REFERENCES

- Project Repository (Sentiment Analysis – GitHub):
https://github.com/Manikantasakalabakthula/Image-Captioning-withDeep-Learning

Note: Used as a structural guide and template reference.

- Amazon Product Review Dataset (CSV Format):

Sourced from public review datasets and cleaned manually for project training.

- Scikit-learn Documentation – Logistic Regression & TF-IDF:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

- Natural Language Toolkit (NLTK):
https://www.nltk.org/

Used for text preprocessing – stopword removal, tokenization, etc.

- Flask (Python Web Framework):
https://flask.palletsprojects.com/

Used to build the real-time sentiment prediction web interface.

- OpenAI (General NLP Research Reference):
https://openai.com/

Referenced for understanding transformer-based NLP models and hybrid architecture trends.

- IEEE Xplore Research Article – Sentiment Classification Models:
https://ieeexplore.ieee.org/document/9740788/references#references

Used for understanding previous approaches to product review sentiment classification.

APPENDIX

1. INTRODUCTION

The Amazon Product Review Sentiment Analysis project aims to classify product reviews into **Positive**, **Negative**, or **Neutral** categories using **Natural Language Processing (NLP)** and **Machine Learning (ML)** techniques. This appendix outlines the project's implementation, including data preparation, model design, training, testing, and deployment using a web-based interface.

2. DATA PREPARATION

The project uses a labeled dataset of Amazon product reviews. The dataset is loaded and stored locally. Each review undergoes a **preprocessing pipeline** that includes lowercasing, punctuation removal, stopword filtering, and tokenization. The cleaned text is then converted into **TF-IDF vectors**, which represent the importance of words within the review corpus.

3. MODEL ARCHITECTURE

The system is based on a **hybrid architecture** consisting of rule-based keyword matching and a **Logistic Regression** classifier. TF-IDF is used for feature extraction, and Logistic Regression maps those features to sentiment classes. The rule-based layer acts as a fast filter for highly polarized input, while the machine learning model handles nuanced and ambiguous cases.

4. MODEL TRAINING

The training process involves adjusting weights to minimize **log-loss (cross-entropy)**. Class weights are balanced to address any class imbalance, and hyperparameters are tuned for better accuracy. The trained model is serialized and saved for deployment.

5. MODEL EVALUATION

The trained model is evaluated using standard classification metrics such as **Accuracy**, **Precision**, **Recall**, and **F1-Score**. These metrics help in assessing the model's ability to correctly predict the sentiment of unseen reviews and ensure its robustness in a real-world context.

6. RESULTS

The final model demonstrates strong performance in classifying review sentiments, especially when combined with the rule-based layer. The system is capable of providing **real-time, interpretable predictions** through a user-friendly Flask interface, making it suitable for practical deployment in e-commerce environments.

7. CONCLUSION

This project successfully integrates NLP techniques and ML algorithms to deliver a reliable sentiment classification system for Amazon product reviews. It showcases how preprocessing, feature engineering, and hybrid modeling can result in accurate, scalable, and user-friendly AI applications.

8. FUTURE WORK

Future enhancements include expanding the system to support **multilingual sentiment detection**, integrating **transformer-based models** like BERT for deeper contextual understanding, and adapting the model for use with **audio reviews** or **live feedback** in IoT and customer support systems.

9. ACKNOWLEDGEMENTS

The team acknowledges the contributions of the **open-source community**, developers, researchers, and creators of Python libraries like **scikit-learn**, **Flask**, **NLTK**, and **pandas**.