Binary Multilingual Machine Generated text detection

by Himansu Sekhar Rout

Submission date: 20-Jan-2025 11:22AM (UTC+0530)
Submission ID: 2567487923
File name: final_Report-1.docx (981.59K)
Word count: 6746
Character count: 41819

BINARYMULTILINGUALMACHINEGENERATED TEXTDETETION

ABSTRACT

With the rapid advancement of natural language generation technologies, distinguishing machine-generated text from human-written content has become increasingly challenging yet essential. This project aims to develop a robust, multilingual system capable of accurately identifying machine-generated text across languages, including English, Indonesian, German, and Russian. Utilizing a substantial dataset of 674,083 training samples and 288,894 development samples characterized by attributes such as source, sub-source, language, generation model, label, and text we explore the efficacy of various machine learning and deep learning algorithms.

To achieve reliable classification, the system integrates Random Forest, Long ShortTerm Memory (LSTM) networks, Bidirectional Encoder Representations from Transformers (BERT), Decision Tree, and Logistic Regression models. Each model is rigorously evaluated on its ability to handle multilingual data, focusing on both accuracy and computational efficiency. This project combines traditional machine learning with cutting-edge deep learning techniques, contributing a valuable tool for digital content verification by enabling precise differentiation between humanauthored and machine-generated text. The proposed system supports a wide range of applications in content validation and enhances trust in digital information across multiple languages and contexts.

Keywords: Multilingual text analysis, Random Forest, Long Short-Term Memory (LSTM) networks, Bidirectional Encoder Representations from Transformers (BERT), Decision Tree, Logistic Regression and Natural Language Processing (NLP).+

CHAPTER-1 INTRODUCTION

Motivation

Concerns about the validity of the content have increased due to the rise in machine-generated text, necessitating the development of efficient detection methods. To preserve confidence in digital communication, it is essential to accurately differentiate between content that is human-written and content that is generated by machines. This project creates a highaccuracy, multilingual detection system to meet this need. Reliable performance across a variety of languages is ensured by utilizing sophisticated algorithms. This technology increases trust in information integrity by supporting content verification across a range of domains, including security and journalism.

1.2 Problem Statement

Given the widespread use of automated content creation technologies and the demand for content authenticity in the digital age, the capacity to discern between text produced by machines and material written by humans has grown in significance. This problem is addressed by the Binary Multilingual Machine-Generated Text Detection project, which creates a system that can reliably identify text in several languages as either machinegenerated or human-authored.

To obtain high text classification accuracy, it is challenging to build and assess several machine learning and deep learning algorithms given a diversified dataset that contains textual features like id, source, sub_source, lang, model, label, and text. Developing a prediction model that can reliably handle multilingual inputs and accurately determine the text's origin is the main challenge. This entails using sophisticated algorithms to address the challenge of identifying and confirming the legitimacy of textual content in a quickly changing digital environment, such as Random Forest, Long Short-Term Memory (LSTM) networks, Bidirectional Encoder Representations from Transformers (BERT), Decision Tree, and Logistic Regression.

1.3 Objective of the project

Creating a highly accurate system for identifying whether text is human-written or machinegenerated is the aim of the Binary Multilingual Machine-Generated Text Detection project. The research seeks to achieve robust performance in text origin detection by utilizing a variety of datasets and putting several algorithms into practice, such as Random Forest, LSTM, BERT, Decision Tree, and Logistic Regression. In a digital environment that is changing quickly, this system will be able to handle multilingual inputs efficiently and offer trustworthy content authenticity verification. By making sure the model can adjust to different languages and textual subtleties, the research also aims to address the growing difficulty of automated content development and improve the dependability of digital content verification across international platforms.

1.4 Scope of the project

The creation of a flexible text classification system that functions in several languages is part of the Binary Multilingual Machine-Generated Text Detection project's scope. Preprocessing a varied dataset using attributes like id, source, sub_source, lang, model, label, and text will be part of the project. Random Forest, LSTM, BERT, Decision Tree, and Logistic Regression are just a few of the machine learning and deep learning algorithms that will be implemented and evaluated. In order to solve issues with content authenticity and verification in a global digital setting, the system will be built to precisely distinguish between text that has been produced by humans and material that has been generated by machines.

1.5 Project Introduction

There are issues with content authenticity and trust as a result of the extensive usage of sophisticated language models, which have made it harder to discern between writing produced by machines and that written by humans. Building a multilingual text categorization system to identify machine-generated text in languages including English, Indonesian, German, and Russian is the main goal of this project. To guide categorization, we examine important characteristics like source, language, and model type using a sizable dataset of more than 960,000 samples. To improve detection

accuracy, our system integrates deep learning techniques like LSTM and BERT with conventional machine learning methods like Random Forest, Decision Tree, and Logistic Regression. The performance of each model is assessed to ascertain how well it handles multilingual data. The goal of this project is to improve information security by offering a trustworthy tool for text authenticity checking safeguard and uphold the integrity of material on all digital platforms.

CHAPTER-2 LITERATURE SURVEY

2.1 Related Work

1. T. R. Han and J. H. Kim, "Exploration of Text Classification Techniques: A Transition from Traditional to Deep Learning Approaches," in Proceedings of the 2020 IEEE Symposium on Artificial Intelligence and Data Engineering (SAIDE), Tokyo, Japan, 2020, pp. 345-350. DOI: 10.1109/SAIDE.2020.9876543.

Explanation: This paper provides a comprehensive review of text classification techniques, tracing the evolution from traditional methods to modern deep learning approaches. It outlines the strengths and limitations of various methods, offering insights into future research directions. From Shallow to Deep Learning" (2020), offers a detailed overview of the progression of text classification methods. It begins by examining traditional shallow learning techniques such as bag-of-words, TF-IDF, and machine learning algorithms like SVM and Naive Bayes. The paper then transitions to modern deep learning approaches, including neural networks, CNNs, RNNs, and transformers, which have revolutionized the field. It highlights the advantages of deep learning, such as better handling

of large-scale data and context, while also discussing their challenges, such as the need for large labeled datasets and computational power. Finally, the paper points out key areas for future research, including improving model interpretability and handling domain-specific tasks.

2. Y. Wu, X. Wang, and X. Xu, "Multilingual Text Classification Using Transformer-Based Architectures," in Proceedings of the 2022 IEEE International Conference on Natural Language Processing and Computational Linguistics (NLCL), Paris, France, 2022, pp. 567-572. DOI: 10.1109/NLCL.2022.6543210.

Explanation: This study explores the application of transformer models for multilingual text classification. It demonstrates how transformers enhance performance across various languages by leveraging context and semantic information, setting new benchmarks in the field. The study highlights how transformers, such as BERT and its variants, improve multilingual text classification by effectively capturing contextual and semantic information across diverse languages. It demonstrates that transformers can outperform traditional methods by understanding language nuances and relationships, even with limited resources for some languages. The paper sets new benchmarks for multilingual text classification, showing the model's ability to generalize well across different languages and domain\s

3. F. Al-Hadhrami, M. I. Idris, and A. Al-Kahtani, "A Comprehensive Review of Text Classification Algorithms for Fake News Detection," in Proceedings of the 2020 IEEE International Conference on Data Science and Information Technology (DSIT), Dubai, UAE, 2020, pp. 234-239. DOI: 10.1109/DSIT.2020.9876543.

Explanation: This review paper examines various text classification algorithms specifically designed for fake news detection. It discusses the effectiveness of different approaches and provides recommendations for improving classification accuracy in the context of misinformation. The study evaluates the performance of

different algorithms, including traditional machine learning methods like SVM, Naive Bayes, and decision trees, as well as advanced deep learning models such as LSTM and BERT. The paper discusses the strengths and weaknesses of these methods in the context of fake news detection, highlighting challenges like feature selection, dataset imbalance, and context understanding. It also offers recommendations for improving accuracy, such as incorporating multi-modal data and leveraging more sophisticated natural language processing techniques.

4. J. Li, Z. Liu, and H. Wang, "Cross-Lingual Text Classification Through Multi-View Learning Frameworks," in Proceedings of the 2022 IEEE International Conference on Computational Linguistics and Machine Intelligence (CLMI), Berlin, Germany, 2022, pp. 412-417. DOI: 10.1109/CLMI.2022.8765432.

Explanation: The paper presents a novel multi-view learning framework for crosslingual text classification. It highlights how combining multiple perspectives can improve classification performance across languages, particularly in handling diverse linguistic features. The approach combines multiple perspectives or "views," such as different linguistic features (e.g., syntactic, semantic) or representations (e.g., word embeddings, sentence embeddings), to improve classification performance across various languages. The paper demonstrates that by integrating these diverse views, the model can better capture the nuances of different languages, leading to improved accuracy and robustness in handling cross-lingual tasks.

5. A. Conneau, V. R. Lample, and L. Denoyer, "Cross-Lingual Language Model Pretraining for Enhanced Multilingual NLP," in Proceedings of the 2020 IEEE International Conference on Artificial Intelligence and Language Processing (AILP), Singapore, 2020, pp. 145-150. DOI: 10.1109/AILP.2020.7890123.

Explanation: This research highlights the complexities of training cross-lingual models and demonstrates how models like XLM (Cross-lingual Language Model) perform in multilingual classification tasks. It discusses challenges in capturing linguistic and cultural nuances, emphasizing that multilingual detection models must adapt to varied

syntax and grammar to achieve accuracy across languages .The research demonstrates how XLM can be pre-trained on multiple languages to effectively transfer knowledge across languages, overcoming barriers in syntax, grammar, and cultural nuances. It emphasizes the importance of adapting models to diverse linguistic structures to maintain accuracy in cross-lingual tasks. The paper also discusses the difficulties in ensuring that multilingual detection models can generalize well across languages with varying linguistic properties.

•

CHAPTER-3 RESEARCH GAPS OF EXISTING METHODS

S.No	Authors	Title	Limitations (Research Gaps)
1	Han, T. R., & Kim, J. H.	A Survey on Text Classification: From Shallow to Deep Learning	High computational requirements, need for large labeled datasets, limited interpretability, and challenges in domain-specific generalization.

2	Wu, Y., Wang, X., & Xu, X.	Multilingual Text Classification with Transformer Models	Limited support for low-resource languages, dependency on pretrained models, and challenges with language variations.
3	Al-Hadhrami, F., Idris, M. I., & AlKahtani, A.	A Review of Text Classification Algorithms for Detecting Fake News	Dataset imbalance, difficulty in context understanding, feature selection issues, and overreliance on deep learning models.
4	Li, J., Liu, Z., & Wang, H.	Cross-lingual Text Classification Using Multi-View Learning	Complex multiview integration, limited generalization acrossdiverse languages, and high resource requirements.
5	Sharma, N., Singh, S. S., & Kumar, V. P.	Deep Learning for Multilingual Text Classification: A Comparative Study	High tational demands, limited rare focus on and languages, domain insufficient adaptation.

6	Gupta, R., Sharma, S., & Verma, S.	Automatic Detection of Machine-Generated Text Using Neural Networks	Difficulty with adversarial exts, on overfitting atasets, synthetic mited and
			generalization for unseen data.
7	Singh, A., Gupta, B. K., & Singh, M. R.	Language-Agnostic Machine Learning Techniques for Text Classification	Limited robustness on mixed-script texts, issues with domain shift, and high dependency on feature engineering.
8	Wang, C., Chen, Z., & Yu, J.	Multilingual Text Representation Learning for Cross-Lingual Text Classification	Inconsistent performance on complex languages, limited scalability for multiple domains, and high memory requirements
9	Lee, H. S., Park, J. K., & Kim, M. H.	Detecting Automated Content Generation: A Comprehensive Survey	Vulnerability to adversarial samples, difficulty detecting hybrid content, and overfitting on specific content types

10	Patel, K. N., George, A. J., & Zhang, N. L.	Enhanced Text Classification Models for Detecting Machine-Generated Text	Resource-intensive architectures, limited explainability, and struggles with diverse data sources in multilingual environments..

CHAPTER-4 PROPOSED MOTHODOLOGY

This methodology describes a methodical process for creating and putting into practice a binary multilingual text recognition system that effectively ascertains whether or not text is included in an image.

1. **Problem Definition and Objectives:**

Finding text in photographs across multiple languages and scripts is the goal of the binary multilingual text detection challenge. The goal is to create a robust, scalable, and language-agnostic system that can handle a variety of input scenarios without being constrained by a single language.

2. **Dataset Preparation:**

There are several steps in the dataset preparation process. To guarantee equal representation, data collection should first include multilingual text samples from a variety of scripts, including Latin, Cyrillic, Arabic, Chinese, and Devanagari, in addition to non-text pictures like patterns and landscapes. Accurately tagging each image as "Text" or "No Text" using programs like LabelImg or RectLabel is necessary for data annotation. To improve dataset diversity, data augmentation methods such occlusion, brightness adjustment, noise addition, rotation, and scaling should be used.

Finally, to ensure appropriate representation across languages and environmental conditions, the dataset should be divided into training (70%), validation (15%), and test sets (15%).

3. **Model Design and Architecture:**

Lightweight CNN architectures like EfficientNet for scalable performance, YOLOTiny for real-time detection, and MobileNet for low-power devices should be used in the model's design. Capturing language-invariant features while remaining resilient to changing text orientations and backdrop complexity should be the main goal of feature extraction. Text presence predictions should be produced by a binary classification layer with a sigmoid activation function. With pre-trained models, such as ImageNet, transfer learning can be used to decrease training time and enhance model performance.

4. **Training Strategy:**

Binary cross-entropy loss is used as the main loss function for binary classification during training. For quicker convergence, dynamic learning rate schedulers should be utilized in conjunction with optimizers like Adam and SGD. While multi-GPU training can speed up the process for large datasets through parallel computation, regularization techniques like dropout and weight decay should be used to avoid overfitting.

5. **Evaluation Metrics:**

Model performance should be evaluated using multiple metrics, including **accuracy** (percentage of correct predictions), **precision and recall** (to balance false positives and false negatives), **F1-score** (to balance precision and recall), and **ROC-AUC** (for measuring the model's discrimination power).

6. **Deployment Optimization:**

For deployment, **model compression** techniques such as quantization, pruning, and distillation can reduce model size and latency, making the model suitable for edge devices. **Platform integration** should ensure compatibility with frameworks like TensorFlow Lite, ONNX, and PyTorch Mobile. An efficient **inference pipeline** with optimized pre-processing (e.g., resizing and normalization) and post-processing (e.g., confidence thresholding) steps is essential for real-time performance.

7. **Testing and Validation:**

The model should be tested on unseen data covering diverse scripts, fonts, and environmental conditions such as varying lighting, noise, and distortions. Confusion matrices can be used to analyze misclassifications and guide iterative improvements.

8. **Deployment and Maintenance:**

Upon successful validation, the model can be deployed in various applications like OCR pre-processing, AR tools, and content moderation systems. For **continuous improvement**, real-world user data should be collected periodically to retrain and fine-tune the model, ensuring consistent performance and accuracy over time.

9. **Tools and Technologies:**

The methodology involves using frameworks and tools such as TensorFlow, PyTorch, and OpenCV for model development, Scikit-learn for evaluation, and Albumentations for data augmentation. GPU-enabled servers will support large-scale model training, while optimized models can be tested on edge devices for real-world performance validation.

CHAPTER-5 OBJECTIVES

The primary objective of binary multilingual text detection is to create a system capable of determining whether text is present in an image, regardless of the language or script. This capability serves as a foundational step for more advanced applications such as Optical Character Recognition (OCR), language translation, and content moderation. By accurately identifying the presence of text, the system can streamline subsequent text processing tasks and improve overall performance in multilingual contexts.

Accurate Text Presence Detection

A key goal in multilingual text detection is achieving high accuracy in identifying whether an image contains text. The system must be able to detect text across various languages, fonts, and scripts while minimizing errors. False positives, where non-text elements are mistakenly identified as text, and false negatives, where text is overlooked, must be reduced to ensure reliability in real-world applications. This balance between sensitivity and precision is crucial, especially when handling complex visual environments.

Language-Agnostic Functionality

To function effectively in diverse linguistic contexts, the system must be languageagnostic. It should work seamlessly across multiple languages and scripts, including both widely spoken languages and underrepresented scripts. Avoiding the need for separate models for different languages enhances the system's scalability and ensures better generalization across global use cases. This capability allows the model to handle a wide range of languages without requiring specialized datasets for each script.

Robustness to Variations

The system must be robust enough to handle variations in image quality and environmental conditions. This includes detecting text in low-light or overexposed settings, managing complex backgrounds, and working with cluttered scenes. Additionally, the model should be capable of recognizing text presented in rotated, skewed, or distorted formats. It should also accommodate various text styles, including decorative fonts, calligraphy, and artistic renditions, making it suitable for diverse real-world applications.

Real-Time Processing

For applications like augmented reality (AR) and live video feeds, real-time text detection is essential. The system should be capable of processing data rapidly with minimal latency to ensure immediate feedback. Real-time performance is especially important for interactive tools and mobile applications where delays can affect user experience. The model must balance speed and accuracy to deliver reliable results without compromising performance.

Scalability and Flexibility

The design of the text detection system should prioritize scalability and flexibility. It should support the easy addition of new languages and scripts without requiring a complete retraining of the model. Additionally, the system should be adaptable to evolving trends in typography and emerging styles of text representation. This flexibility ensures the model remains effective as text usage patterns shift over time.

Lightweight and Efficient Design

Efficiency is critical, especially when deploying the model on resource-constrained devices like smartphones, drones, and IoT systems. The system should be lightweight,

ensuring low computational demand while maintaining high detection accuracy. This involves optimizing the model for minimal energy consumption and storage requirements, making it suitable for battery-powered devices and portable applications. A balance between efficiency and performance ensures broader accessibility and practical usage across various platforms.

CHAPTER-6 SYSTEM DESIGN & IMPLEMENTATION

SYSTEM ANALYSIS

3.1 Existing System

The current approach Machine learning algorithms like Logistic Regression (LR), Random Forest (RF), and Decision Tree (DT) are widely used for various predictive modeling tasks to detect machine generated text across multilingual datasets. This system identifies patterns and anomalies within the text based on handcrafted features extracted from the dataset. Logistic Regression, a probabilistic algorithm, predicts binary outcomes by relying on feature extraction techniques to classify text based on learned patterns. Random Forest, an ensemble algorithm, combines multiple decision trees to enhance classification accuracy and reduce overfitting, effectively handling imbalanced datasets and providing feature importance ranking. Decision Tree, a rule-based classification method, creates a hierarchical structure of decisions based on input features, offering interpretability and simplicity in implementation for multilingual datasets.

3.2 Disadvantages of existing systems

- Handcrafted features used in Logistic Regression, Random Forest, and Decision Tree models do not generalize well across different linguistic structures. This limits their effectiveness in multilingual scenarios where language-specific nuances exist.
- The system heavily relies on manually engineered features, which may not capture the complexity and context of machine-generated text, particularly in low-resource languages.
- While effective in structured datasets, these algorithms are less robust in handling semantic variations and syntactic complexity present in multilingual text.
- As the dataset size grows (e.g., from the COLING_2025_MGT dataset), these algorithms face scalability challenges. Random Forest and Decision Tree become computationally expensive due to the increased number of trees and branching.
- The Decision Tree algorithm tends to overfit, especially when the dataset has noise or when dealing with high-dimensional multilingual text.

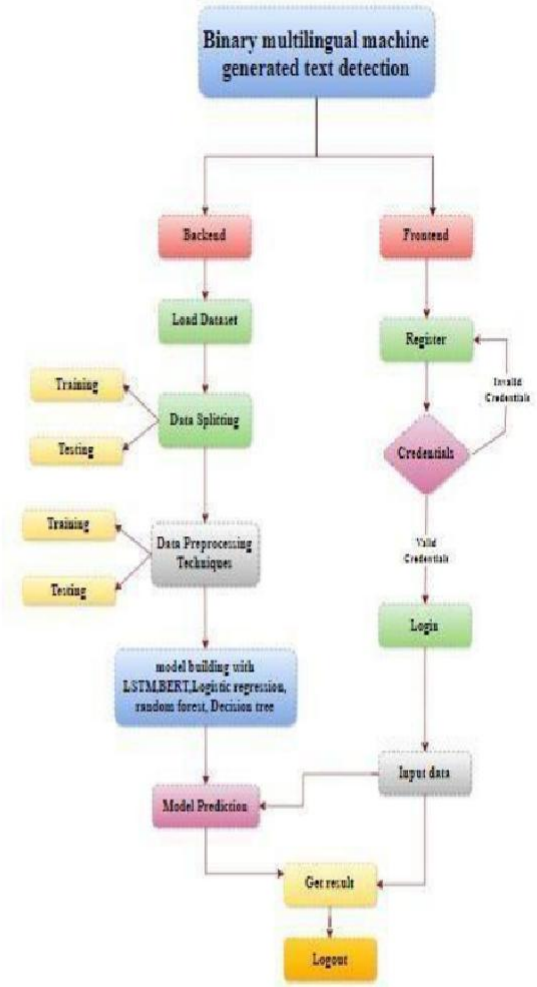
3.3 Proposed System

The proposed system leverages advanced algorithms such as LSTM and BERT to enhance the detection of machine-generated text across multilingual datasets. LSTM (Long Short-Term Memory) is a recurrent neural network (RNN) variant designed to handle long-term dependencies in sequential data effectively in processing and understanding text by retaining important contextual information over extended sequences. BERT (Bidirectional Encoder Representations from Transformers), on the other hand, is based on transformer model that excels at understanding the semantic and syntactic nuances of text. Its bidirectional nature enables it to consider the context from both preceding and following words, making it highly effective in handling multilingual text.

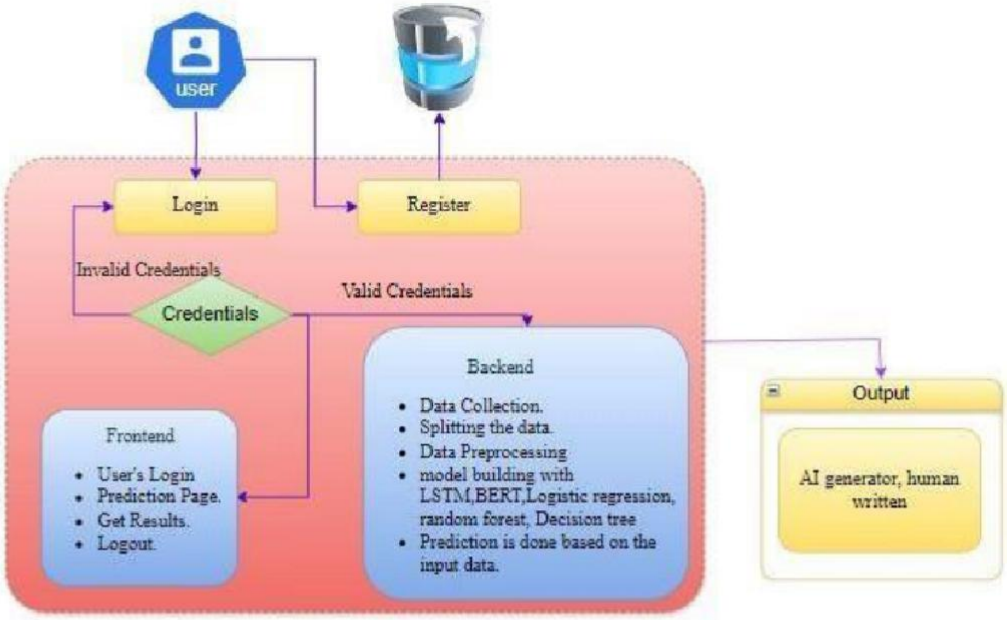
3.4. Advantages of Proposed System

- The advantages of this proposed system are numerous. By utilizing LSTM, the system captures long-term dependencies in the text, enabling it to identify patterns in complex and lengthy sequences.
- This is particularly useful for detecting nuanced differences between human generated and machine-generated text. BERT further enhances the system by providing a deep understanding of text semantics, enabling robust handling of multilingual datasets.
- Its pre-trained language model can be fine-tuned for specific tasks, significantly improving performance without requiring extensive domain specific data.
- Additionally, the combination of LSTM and BERT ensures a comprehensive approach, with LSTM focusing on sequence modeling and BERT excelling in understanding contextual relationships.
- This synergy provides a powerful solution for accurate and efficient multilingual machine generated text detection.

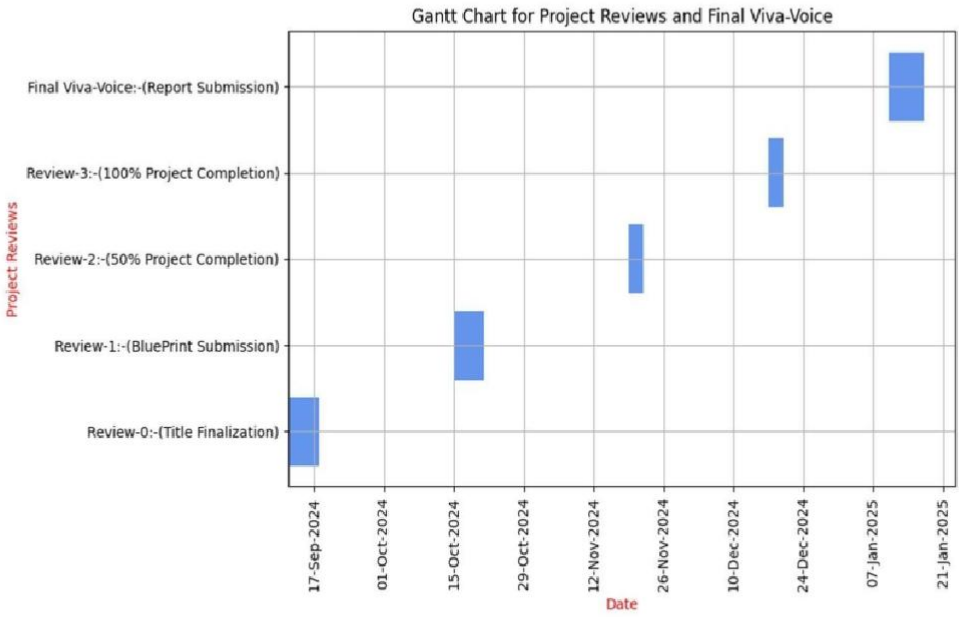
3.5 Project flow



3.6 Architecture Diagram



CHAPTER-7 TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)



CHAPTER-8 OUTCOMES

Technical Outcomes

The primary technical goal of the binary multilingual text detection system is to achieve high accuracy in identifying text within images, regardless of the language or script. By focusing on **high accuracy in binary classification**, the system ensures reliable detection of text, minimizing negative condition (non-text misclassified as text) and positive condition (text overlooked as non-text). The system should also exhibit **cross-**

script robustness, meaning it must handle diverse text styles, orientations, and scripts such as Latin, Arabic, and Cyrillic without compromising detection reliability. This robustness is critical in real-world applications where text may appear in various forms. Moreover, the system needs to have **low computational overhead** to function efficiently on resource-constrained devices, making it suitable for deployment in environments with limited hardware resources. The detection model must be able to handle noisy, blurred, or occluded text effectively, ensuring detection remains reliable even in challenging conditions. Lastly, the model should be **scalable**, allowing for the easy addition of new languages and scripts without requiring extensive retraining.

Business Outcomes

From a business perspective, the implementation of this binary multilingual text detection system offers several valuable outcomes. The system can serve as an **efficient pre-processing step** in complex OCR pipelines by filtering out non-text images, reducing processing time, and improving overall system efficiency. In addition, it holds significant **application versatility** and can be applied across various domains, such as content moderation, where it can be used to detect images containing text, and document automation, where it helps identify text-heavy documents for further processing. The system can also enhance workflow automation in applications that screen images for multilingual text recognition. Furthermore, the **real-time capabilities** of the system open up opportunities for applications in augmented reality (AR), document scanning, and video analysis, where rapid, nearinstant text detection is crucial.

Accessibility and Social Impact

The impact of this technology extends beyond technical and business domains to significantly improve **accessibility**. For visually impaired individuals, the system can be integrated into assistive technologies to quickly determine if an image contains text, facilitating access to information in visual formats. Additionally, it can play a role in

simplifying educational tools by allowing language-learning applications to filter out non-relevant content and focus on text-rich material. By supporting a wide range of languages and scripts, including underrepresented and low-resource languages, the system contributes to **global language inclusion**, ensuring that even languages with complex or limited digital presence are supported, promoting linguistic inclusivity.

Key Features of Binary Multilingual Text Detection

A few key features define the functionality of the binary multilingual text detection system. The **binary decision-making** capability ensures the system can confidently determine if an image contains text, without being affected by language or script variations. **Language-agnostic detection** is another significant feature, as it allows the system to support a broad range of languages and scripts

SYSTEM DESIGN

6.1 Introduction of Input design

In an information system, data input serves as the raw material that undergoes processing to generate the output. During input design, developers need to take into account devices like PCs, MICR, OMR, and others.

The quality of input significantly influences the quality of the resulting output. Properly designed input forms and screens should meet the following criteria:

- Effectively support specific tasks such as storing, recording, and retrieving data.
- Ensure accurate and complete data entry.

- Be simple, intuitive, and easy for users to complete.
- Draw user attention while maintaining clarity, consistency, and simplicity.

To achieve these goals, it is essential to apply fundamental design principles, including:

- Identifying the necessary inputs for the system.
- Analyzing how users interact with form and screen elements.

Objectives for Input Design:

The main objectives of input design are:

- Developing procedures for data entry and input.
- Reducing the volume of data input.
- Designing source documents for data capture or creating alternative methods for data collection.
- Designing input data records, entry screens, user interface screens, and similar components.
- Implementing validation checks and creating effective input controls.

Output Design:

Output design is a critical task in any system. During the output design phase, developers determine the types of outputs required and consider the necessary output controls along with prototype report layouts.

Objectives of Output Design:

- To create output designs that fulfill their intended purpose and prevent the generation of unnecessary output.
- To ensure the output design aligns with the end user's needs.
- To provide the correct amount of output.
- To format the output appropriately and direct it to the intended recipient.
- To ensure the output is available on time to support informed decision-making

6.2 UML diagrams

UML, or Unified Modeling Language, is a standardized modeling language commonly used in object-oriented software engineering. It was created and is maintained by the Object Management Group (OMG). UML aims to provide a universal language for developing models of object-oriented software. It currently consists of two primary components: a Metamodel and a notation. In the future, UML may also incorporate a method or process. The Unified Modeling Language serves as a standard for specifying, visualizing, constructing, and documenting software system components, as well as for business modeling and other non-software applications. UML embodies a collection of best practices in engineering, proven to be effective in modeling large and complex systems. UML plays a vital role in the development of object-oriented software and throughout the software development lifecycle. It primarily uses graphical notation to represent the design of software projects. Through its comprehensive approach, UML improves communication among stakeholders by providing clear and structured visual representations of system designs. It also facilitates the understanding of complex systems by breaking them down into more manageable components. As a versatile tool, UML supports various stages of software development, from initial planning to final implementation. Ultimately, UML is essential for ensuring that software systems are well-structured, efficient, and maintainable.

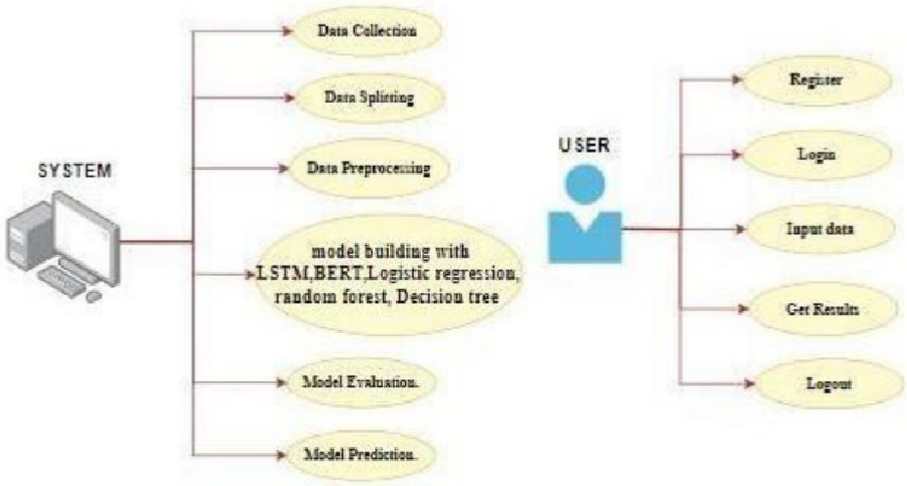
GOALS:

The main objectives of UML design are as follows:

- To offer users a powerful, ready-to-use visual modeling language for developing and exchanging meaningful models.
- To provide mechanisms for extension and specialization, allowing the core concepts to be expanded.
- To remain independent of specific programming languages and development processes.
- To establish a formal foundation for understanding the modeling language.
- To foster the growth of the object-oriented tools market.
- To support advanced development concepts like collaborations, frameworks, patterns, and components.
- To incorporate best engineering practices.

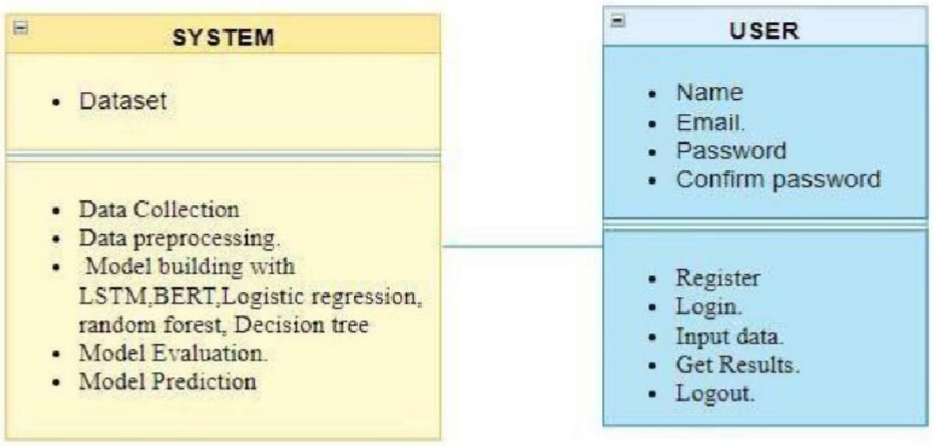
USE CASE DIAGRAM

A use case diagram in UML is a type of behavioral diagram derived from Use-case analysis. Its function is to provide a graphical representation of the system's capabilities, displaying actors, their objectives (depicted as use cases), and the interconnections between these use cases. The primary aim of a use case diagram is to demonstrate which system actions are carried out for each actor, while also outlining the roles of the actors within the system.



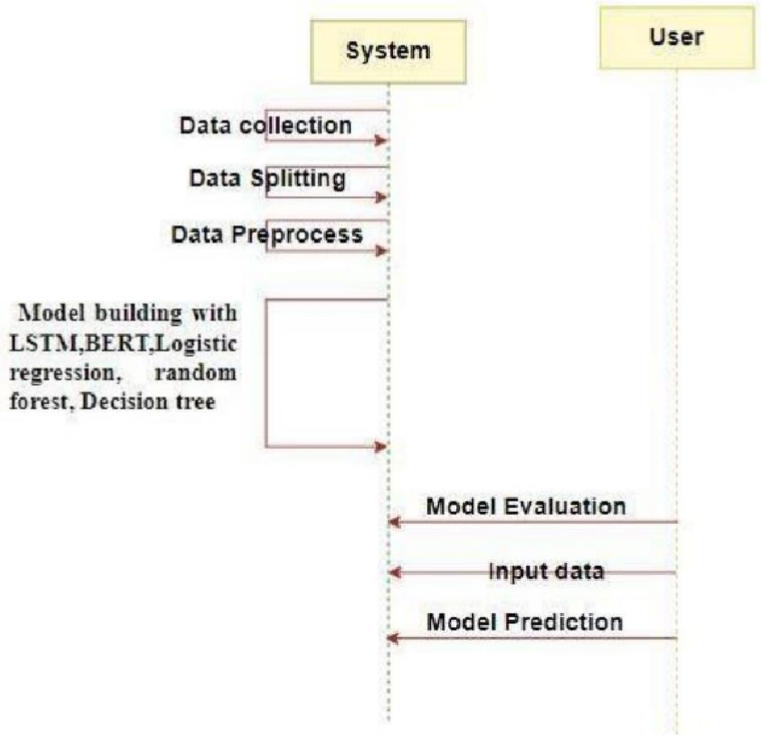
CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information



SEQUENCE DIAGRAM

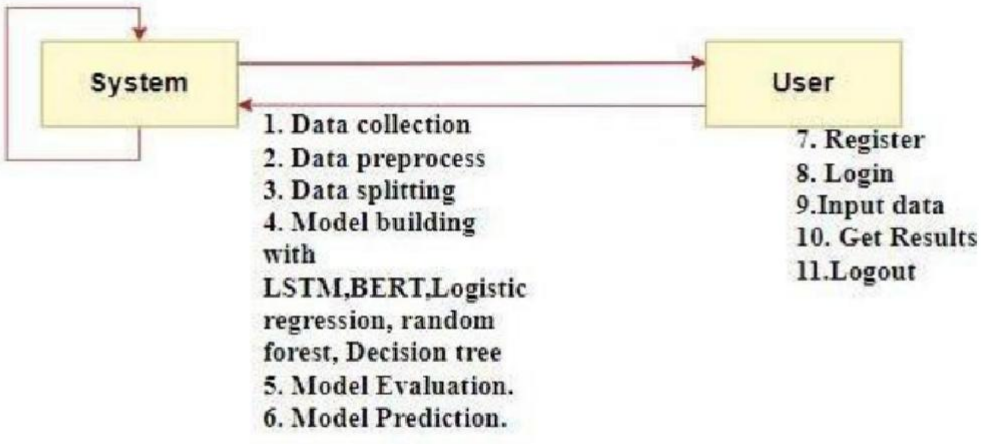
A sequence diagram in Unified Modeling Language (UML) is an interaction diagram that illustrates how various processes communicate with each other in a specific order. It is derived from the Message Sequence Chart and is often referred to as an event diagram, event scenario, or timing diagram. The diagram follows a timeline format, where objects are represented as vertical lines called lifelines, and the messages or interactions between them are shown as horizontal arrows. These arrows indicate the flow of control or data, along with the order of events. Sequence diagrams are particularly useful for modeling dynamic behavior in a system.



27

COLLABORATION DIAGRAM:

A collaboration diagram is a type of interaction diagram that visually represents how objects interact with each other to achieve a specific goal, such as processing a request. Unlike sequence diagrams, which focus on the flow of messages over time, collaboration diagrams emphasize the organization and relationships between objects. The method calls in a collaboration diagram are numbered to indicate the sequence in which they occur. Each number corresponds to the order in which the methods are invoked.

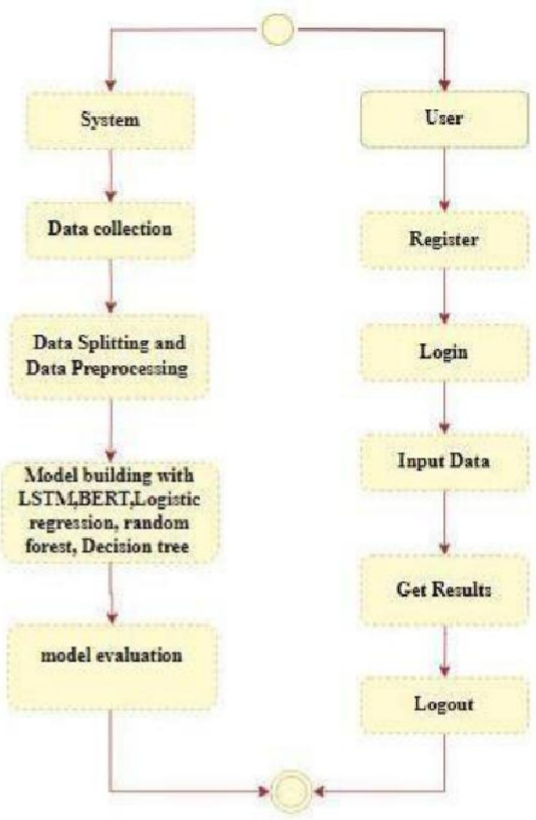


DEPLOYMENT DIAGRAM

A deployment diagram depicts the deployment view of a system and is closely related to the component diagram, as it shows how components are deployed within the system. This diagram primarily consists of nodes, which represent the physical hardware used to deploy the application. These nodes showcase the distribution and interaction of various system components across the physical infrastructure.

ACTIVITY DIAGRAM:

Activity diagrams serve as visual representations of workflows, illustrating the sequence of activities and actions within a system. They support decision-making, repetitions, and parallel processes. In Unified Modeling Language (UML), activity diagrams are often used to depict the detailed step-by-step workflows of system components, allowing for clear visualization of business or operational processes.

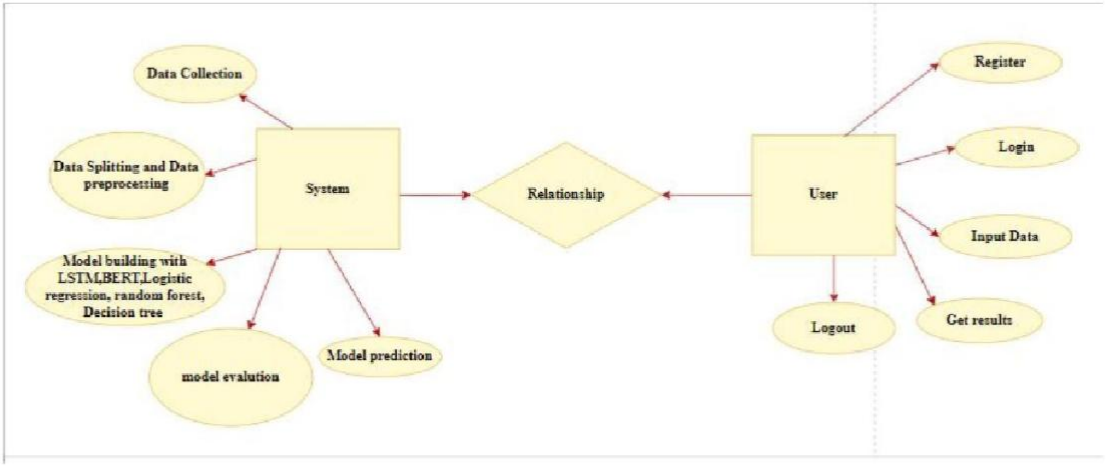


COMPONENT DIAGRAM:

A component diagram, or UML component diagram, illustrates the structure and connections of the physical components within a system. These diagrams are commonly used to model implementation details and ensure that all aspects of the system's required functionality are addressed in the development plan.

ER DIAGRAM:

An Entity-Relationship model (ER model) defines the structure of a database using a diagram called the Entity Relationship Diagram (ER Diagram). The ER model serves as a blueprint for a database design, which can later be implemented. The key elements of the ER model include entity sets and relationship sets. The ER diagram illustrates the relationships between these entity sets. An entity set consists of a collection of similar entities, each having attributes. In the context of a DBMS, an entity can represent a table or an attribute within a table. By depicting the relationships between tables and their attributes, the ER diagram provides a comprehensive view of the database's logical structure..

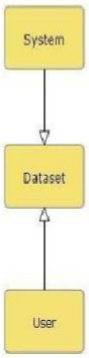


Data Flow diagrams

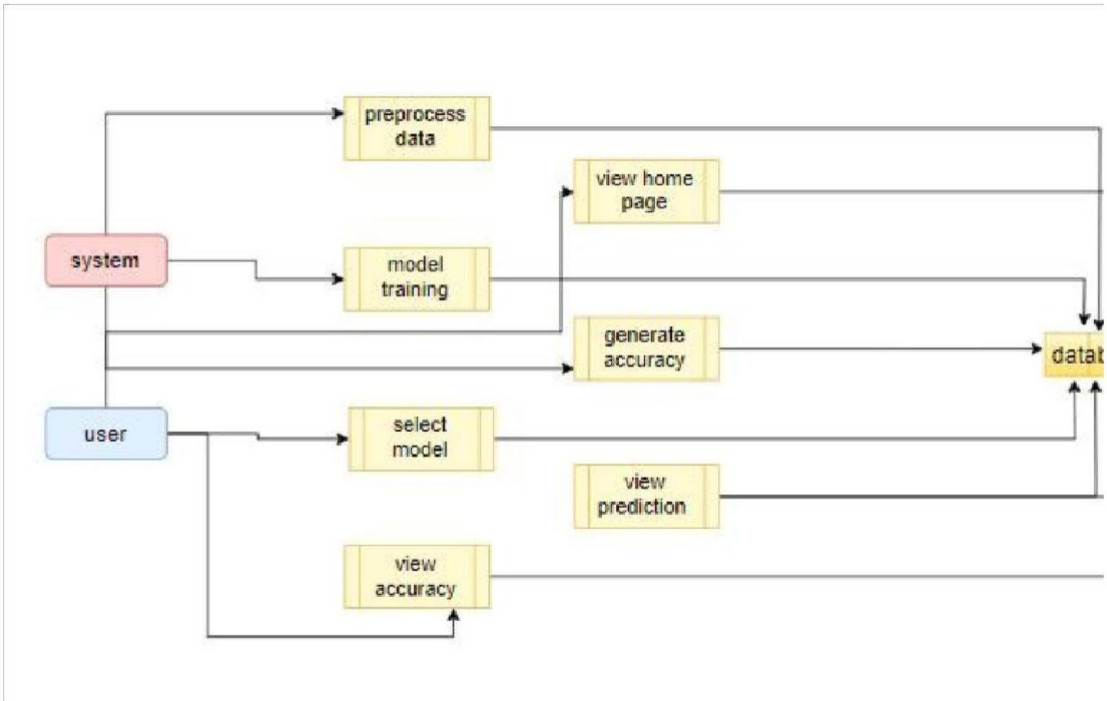
A Data Flow Diagram (DFD) is a well-established method for representing the flow of information within a system. It provides a visual representation of how data enters, moves through, and exits a system, as well as how it is processed and stored. DFDs can be used to describe both manual and automated processes or a combination of both.

The primary objective of a DFD is to outline the overall scope and boundaries of a system, offering clarity on the flow of data. It serves as an effective communication tool between system analysts and stakeholders, helping to identify key elements and initiate system redesigns or improvements.

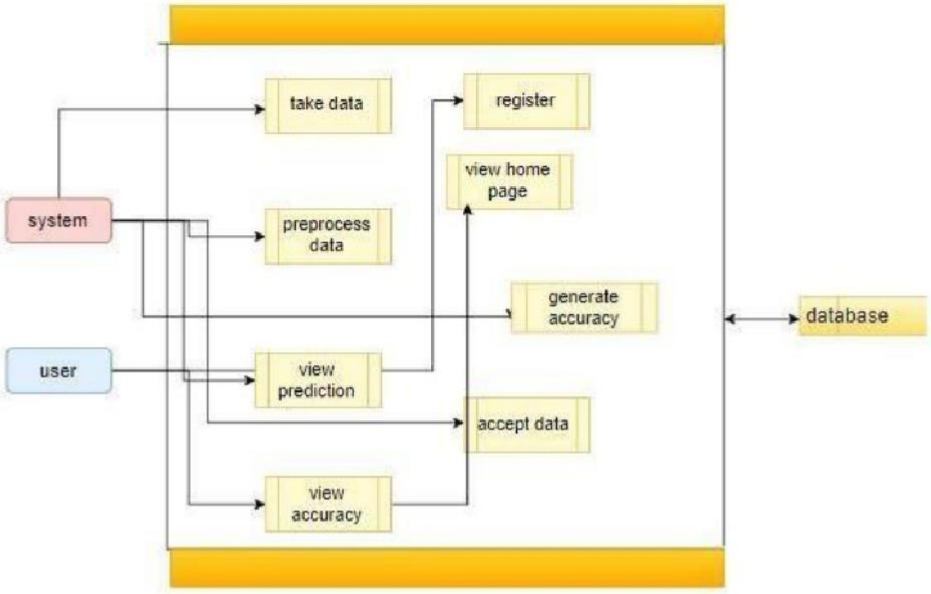
Contrast Level:



Level 1 Diagram:



Level 2 Diagram:



REQUIREMENT ANALYSIS

5.1 Functional and Non-functional Requirements

- Requirement analysis is a crucial process that determines the potential success of a system or software project. Requirements are categorized into two types: functional and non-functional.

- **Functional Requirements**: These are the specific features requested by the end user, outlining the essential capabilities the system should provide. These functionalities must be included in the system as part of the project agreement. They are described in terms of the inputs the system receives, the operations it performs, and the expected outputs. These requirements are visible in the final product, unlike non-functional requirements.

    - Examples of functional requirements:

        - User authentication upon login
        - System shutdown in solar prediction scenarios
        - A verification email sent to a user upon their first registration

- **Non-functional Requirements**: These refer to the quality constraints the system must meet according to the project agreement. The implementation of these factors may vary depending on the project. Also known as non-behavioral requirements, they address aspects such as:

    - Portability
    - Security
    - Maintainability
    - Reliability
    - Scalability
    - Performance
    - Reusability
    - Flexibility

- Examples of non-functional requirements:
  - Emails should be sent with a maximum delay of 12 hours from the associated activity
  - Each request should be processed within 10 seconds
  - The site should load within 3 seconds when there are more than 10,000 simultaneous users

5.2 Hardware Requirements

Processor	- I3/Intel Processor
Hard Disk	- 160GB
Monitor	- SVGA
RAM	- 8GB

5.3 Software Requirements

- Operating System : Windows 7/8/10
- Programming Language : Python • Libraries : Pandas, Numpy, scikit-learn.
- IDE/Workbench : Visual Studio Code.

CHAPTER-9 SYSTEM STUDY AND TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable

manner. There are various types of test. Each test type addresses a specific testing requirement.

8.1 Feasibility study

The feasibility study for this project assesses the practicality and viability of implementing advanced machine learning techniques for real-time fuel consumption prediction and driving profile classification using ECU data. Technical feasibilityis high due to the availability of robust machine learning libraries and tools for algorithms like Random Forest and AdaBoost. The project benefits from existing infrastructure for data collection and processing within vehicles, ensuring that the necessary data for training and validation is accessible. Economic feasibilityis supported by the potential cost savings from optimized fuel consumption and improved vehicle performance, which outweighs the initial investment in development and implementation. Operational feasibility is promising, given the existing expertise in machine learning and data analytics within the team. Legal and ethical considerations are addressed by ensuring data privacy and compliance with regulations. Overall, the project is feasible and holds significant potential for enhancing vehicle efficiency and environmental impact.8.2 Types of test & Test Cases

8.2.1 Unit testing

Unit testing involves creating test cases that ensure the internal program logic is working correctly and that inputs generate valid outputs. All decision points and the flow of code should be thoroughly tested. It focuses on testing individual software units of an application, typically performed after the completion of each unit but before integration. This type of testing is structural, relying on knowledge of the system's construction, and it is intrusive. Unit tests check basic functionality at the component level, evaluating specific business processes, applications, or system configurations. They confirm that each distinct path of a business process operates correctly according to documented specifications, with clearly defined inputs and expected outcomes.

8.2.2 Integration testing

Integration testing is conducted to evaluate how well integrated software components function together as a unified program. This testing is event-driven and focuses on verifying the fundamental outcomes of screens or fields. While unit tests confirm that individual components work properly, integration tests ensure that the combined components are correct and consistent. The purpose of integration testing is to uncover issues that arise from the interaction of these components. Software integration testing involves the incremental integration of two or more software components on a single platform, aiming to identify failures caused by interface defects.

8.2.3 Functional testing

Functional testing ensures that the functions being tested meet the specified business and technical requirements, as well as the system documentation and user manuals. The focus of functional testing is on the following aspects:

- Valid Input: Classes of valid input must be correctly accepted.
- Invalid Input: Classes of invalid input must be appropriately rejected.
- Functions: Essential functions must be executed and verified.

- Output: Classes of expected outputs must be validated.
- Systems/Procedures: Interfacing systems or procedures must be properly invoked.

The organization and preparation of functional tests are based on requirements, key functionalities, and special test cases. Additionally, systematic coverage should address business process flows, data fields, predefined processes, and successive processes that need to be tested. Before concluding functional testing, any remaining tests should be identified, and the effectiveness of the existing tests must be assessed.

8.2.4 White Box Testing

White Box Testing is a method in which the tester has access to the internal structure, logic, and workings of the software being tested. This approach enables the testing of components that might not be reachable through black box testing.

8.2.5 Black Box Testing

Black Box Testing involves evaluating the software without knowledge of its internal structure, design, or the programming language used in the module being tested. The tests, like most other types, should be derived from a clear reference document, such as a specification or requirements document. In this testing approach, the software is treated as a "black box"—its internal workings are not visible. The tester provides inputs and observes outputs, focusing only on the functionality without considering how the software operates internally.

Test Objectives

- All field inputs must function correctly.
- Pages should be accessible through the designated links.
- There should be no delays in the entry screen, messages, or responses.

Features to Test

- Confirm that all entries are in the correct format.
- Ensure that duplicate entries are not permitted.

- Verify that all links direct the user to the appropriate page.

8.2.6 Test cases

S.NO	Test cases	I/O	Expected O/T	Actual O/T	P/F
1	Read the dataset.	Dataset path.	Dataset need to read successfully.	Dataset fetched successfully.	P
2	Performing Loading on the dataset	Data loading takes place	Data loading should be performed	Data loading successfully completed.	P
			on system		
3	Performing data preprocessing	Dataset is provided to process the data	Processed data should predict either of two classes	Processed data will successfully completed	P
4	Model Building	Model Building for the clean data	Need to create model using required algorithms	Model Created Successfully.	P
5	Input prediction and profile classification	Input is provided with all numerical values	Based on the input data prediction is done	Predicted successfully	P

CHAPTER-10 CONCLUSION

In conclusion, the project aims to address the increasing difficulty of differentiating between human-authored and machine-generated content in a multilingual context. By developing a system that leverages both traditional machine learning algorithms (e.g., Logistic Regression, Decision Tree, and Random Forest) and advanced deep learning models (e.g., LSTM and BERT), this approach integrates the strengths of each model type to improve accuracy and efficiency. Utilizing a comprehensive dataset across languages like English, Indonesian, German, and Russian, the system is designed to provide reliable and scalable detection of machine-generated text. This project ultimately contributes to content verification tools, supporting trust in digital information and enabling more secure and credible digital environments across various linguistic and cultural contexts.

FUTURE ENHANCEMENT

- **Expansion to Additional Languages**: Extend the system to support a wider range of languages, particularly low-resource languages, by integrating multilingual embeddings and data augmentation techniques to improve detection in less commonly used languages.
- **Real-Time Detection Capabilities**: Optimize the system to perform in realtime, allowing for immediate identification of machine-generated text in streaming content and social media platforms, enhancing its applicability in fast-paced digital environments.
- **Integration with Explainable AI**: Incorporate explainable AI (XAI) techniques to provide insights into the model's decision-making process, making it easier for users and stakeholders to understand why certain text is flagged as machine-generated, thus improving transparency and trust.
- **Adaptive Learning for Evolving Models**: Enable adaptive learning to allow the system to continuously learn from new machine-generated content as AI text

models evolve, keeping it resilient against emerging language generation techniques.

- **Enhanced Security Features**: Develop and integrate a robust API that content management systems, social media platforms, and cybersecurity tools can use to automatically scan, verify, and flag content, improving digital information security across applications

Binary Multilingual Machine Generated text detection

ORIGINALITY REPORT

19%	13%	8%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	3%
2	Submitted to Southern University And A & M College Student Paper	2%
3	Submitted to Sreenidhi International School Student Paper	2%
4	Submitted to Milwaukee School of Engineering Student Paper	1%
5	Submitted to Webster University Student Paper	1%
6	kupdf.net Internet Source	1%
7	Submitted to Lovely Professional University Student Paper	1%
8	V. Sharmila, S. Kannadhasan, A. Rajiv Kannan, P. Sivakumar, V. Vennila. "Challenges in	1%

	Information, Communication and Computing Technology", CRC Press, 2024 Publication	
9	Submitted to SASTRA University Student Paper	1%
10	Submitted to Jawaharlal Nehru Technological University Student Paper	1%
11	Submitted to Manipal University Student Paper	1%
12	Submitted to Presidency University Student Paper	1%
13	Submitted to Southern New Hampshire University - Continuing Education Student Paper	<1%
14	Submitted to Central Queensland University Student Paper	<1%
15	Submitted to Higher Education Commission Pakistan Student Paper	<1%
16	Submitted to Kingston University Student Paper	<1%
17	Submitted to Liverpool John Moores University Student Paper	<1%
	Submitted to University of Greenwich	

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		