CLEANSE AI:DATA PREPROCESSING AUTOMATION TOOL USING PYTHON AND MACHINE LEARNING ALGORITHMS

A PROJECT REPORT

Submitted by

AUDHAVAN A(811721243010)

BARATHVAJ M(811721243011)

MOHAMED THABISH M(811721243031)

VIGNESHWARAN M(811721243061)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

MAY, 2025

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **"CLEANSE AI:DATA PREPROCESSING AUTOMATION TOOL USING PYTHON AND MACHINE LEARNING ALGORITHMS"**is the bonafide work of **AUDHAVAN A (811721243010), BARATHVAJ M(811721243011), MOHAMED THABISH M (811721243031), VIGNESHWARAN M(811721243061)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. T.Avudaiappan M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Department of Artificial Intelligence

K.Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112

SIGNATURE

Mrs. E.Sri Santhoshini M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of Artificial Intelligence

K.Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112

Submitted for the viva-voce examination held on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the project report on **"CLEANSEAI: DATA PREPROCESSING AUTOMATION TOOL USING PYTHON AND MACHINE LEARNING ALGORITHMS"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF TECHNOLOGY**.

Signature

AUDHAVAN A

BARATHVAJ M

MOHAMED THABISH M

VIGNESHWARAN M

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

We are glad to credit honourable chairman **Dr. K.RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

We would like to thank **Dr. N. VASUDEVAN, M.E., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

We whole heartily thank to **Dr. T.AVUDAIAPPAN**, **M.E., Ph.D.,** Head of the department, **ARTIFICIAL INTELLIGENCE** for providing his encourage pursuing this project.

We express our deep and sincere gratitude to our project guide **Mrs. E. SRI SANTHOSHINI, M.E.,** Department of **ARTIFICIAL INTELLIGENCE,** for her incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

This Python script presents a complete and modular data preprocessing pipeline tailored for machine learning. It begins by loading a CSV file and handling missing values, where numerical fields are filled with their mean and categorical fields with a placeholder label. Text data undergoes basic cleaning by converting to lowercase and removing punctuation. Categorical columns are then encoded using label encoding, making them suitable for numerical processing. The data is normalized using either standard scaling or scaling to ensure uniformity. Outliers are detected and removed based on score thresholds. If a target variable is specified, the script performs feature selection to retain the top five most relevant features. It also generates descriptive statistics, identifies outliers, and visualizes data distributions, relationships, and correlations using plots. The final cleaned and transformed data is saved for future use, making the script a powerful tool for automating preprocessing workflows.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ASR - Automatic Speech Recognition

HMM - Hidden Markov Model

SVM - Support Vector Machine

API - Application Programming Interface

UI - User Interface

SMS - Short Message Service

CHAPTER 1
INTRODUCTION

In today's data-driven world, preparing high-quality datasets for machine learning and analytics has become increasingly complex and time-consuming. Manual data preprocessing is often error-prone, inconsistent, and resource-intensive, posing challenges for both novice and experienced data scientists. To address these issues, the "Cleanse AI" project introduces an intelligent, automated data preprocessing tool designed to streamline the transformation of raw data into a clean, structured, and model-ready format.

Built using Python and integrated with powerful machine learning libraries such as Scikit- learn, Pandas, and NumPy, the tool performs critical preprocessing steps including data loading, cleaning, feature engineering, normalization, outlier handling, and visualization. Cleanse AI uses supervised and unsupervised ML techniques to detect and correct anomalies, impute missing values, and engineer meaningful features, adapting to various data types and domain-specific requirements. With a modular design and interactive user interface, the system allows users to configure preprocessing pipelines while maintaining full transparency and reproducibility of all operations.

Through the automation of routine preprocessing tasks, Cleanse AI not only accelerates the data preparation phase but also ensures consistency and accuracy across datasets. The system is especially beneficial for real-time and large-scale applications, supporting use cases in fields such as healthcare, finance, retail, and education. By simplifying and standardizing the preprocessing process, Cleanse AI empowers users to focus on analysis and model development, ultimately enhancing the reliability and performance of AI-driven solutions

1.1 BACKGROUND

With the explosive growth of data across industries, the demand for reliable and automated data preprocessing has become more critical than ever. Traditional data cleaning and transformation methods are often manual, time-consuming, and prone to inconsistencies, especially when working with large, diverse datasets. These

limitations can hinder the performance of machine learning models and slow down the data science workflow.

To address this challenge, Cleanse AI introduces a Python-based automation tool that leverages machine learning algorithms to perform intelligent data preprocessing. By integrating processes like missing value treatment, feature engineering, outlier detection, and normalization, the tool aims to reduce human intervention while maintaining high data quality. Built with Python and libraries such as Pandas, NumPy, and Scikit-learn, Cleanse AI provides a modular, scalable solution that enhances accuracy, efficiency, and reproducibility in AI-driven projects across various domains.

1.2 PROBLEM STATEMENT

In the current digital and financial landscape, ensuring secure, fast, and user-friendly transactions remains a major challenge. Traditional authentication methods such as passwords, PINs, and touch-based systems are prone to security breaches, user errors, and accessibility issues. As voice-activated and mobile payments grow, existing systems often lack the accuracy, security, and multi-modal integration needed to prevent fraud and enhance usability. There is a pressing need for a seamless, multi-factor authentication system that combines voice recognition, fingerprint verification, RFID, and voice PIN in a cohesive and hands-free solution.

1.3 AIM AND OBJECTIVE

1.3.1 Aim

• Develop an intelligent, automated data preprocessing tool using Python and machine learning to streamline the preparation of raw datasets for analysis and modeling.
• Integrate essential preprocessing techniques such as data loading, cleaning, normalization, outlier handling, and feature engineering into a unified, user-friendly platform.
• Implement the system using Python libraries like Pandas, NumPy, Scikit-learn, and Matplotlib to ensure robustness, flexibility, and compatibility across various data..
• Leverage machine learning algorithms to automate detection and treatment of

missing values, noise, and anomalies while ensuring consistency and accuracy across datasets.

• Deliver a seamless preprocessing workflow with customizable modules that promote ease of use, transparency, and reproducibility for both technical and non-technical users.

• Design the tool to be scalable, modular, and suitable for integration into real-time data pipelines in industries such as healthcare, finance, education, and retail.

Create a Python-based automation system for preprocessing structured and unstructured data, integrating steps such as cleaning, transformation, and feature extraction.

• Enhance the speed, accuracy, and consistency of data preparation by automating routine preprocessing tasks through intelligent ML-driven algorithms.

• Ensure support for diverse data types and structures, allowing users to work efficiently with tabular, categorical, numerical, and textual datasets.

• Promote accessibility and productivity in data science workflows by offering a guided, transparent, and explainable preprocessing interface that facilitates collaboration and reduces errors.

1.3.2 Objective

• Analyze and evaluate existing data preprocessing techniques, including data cleaning, normalization, feature engineering, and outlier detection, across various machine. .

• Develop a Python-based automated tool that supports a complete, modular preprocessing pipeline for diverse datasets and application domains.

• Build intelligent data cleaning and transformation models using supervised and unsupervised machine learning algorithms to enhance preprocessing accuracy .

• Integrate functionalities such as missing value imputation, noise reduction, feature scaling, and categorical encoding into a unified, user-friendly preprocessing environment.

• Improve robustness and adaptability of preprocessing steps by implementing

algorithmic optimizations that dynamically adjust to dataset-specific patterns and challenges.

• Design and implement an intuitive user interface or script-based workflow that allows users to configure and visualize preprocessing steps with transparency and control.

• Provide real-time data diagnostics and preprocessing summaries to help users understand data quality, transformation effects, and feature distributions.

• Incorporate advanced feature engineering capabilities using dimensionality reduction techniques (e.g., PCA) and statistical feature selection methods to improve model readiness.

• Test the tool across various dataset types, sizes, and domains to ensure flexibility, efficiency, and compatibility with downstream machine learning tasks.

• Evaluate the tool's performance through benchmark comparisons with existing preprocessing tools and feedback from data scientists and domain experts.

• Ensure the system is modular, scalable, and ethically designed, with a focus on traceability, reproducibility, and fairness in data handling.

• Promote data accessibility and usability by simplifying preprocessing for non-expert users and supporting integration with commonly used ML pipelines and frameworks.

CHAPTER 2
LITERATURE SURVEY

2.1 APPLICATION OF DIFFERENT EDUCATIONAL DATA PREPROCESSING TECHNIQUES Michal Munk, Martin Drlík,L'ubomír Benko,Jaroslav Reichel et

This system automates the preprocessing of educational log data to enhance the accuracy of learning analytics. It focuses on key steps like data cleaning, user and session identification, and path completion. By analyzing how different preprocessing techniques affect the discovery of sequential patterns, the system improves the quality and quantity of extracted knowledge used for educational insights.

Merits

- Accurate session identification significantly improves the quality of discoveredpatterns.
- Automation of preprocessing helps reduce time and effort in educational datamining.
- Enhances the performance of learning analytics tools for educators and researchers.

Demerits

- Although the model is powerful for news articles, its generalizability to other domains (e.g., social media, academic texts) may be limited unless further fine-tuned.
- Background noise or speech variations could still impact accuracy, leading to errors in both speech recognition and authentication.

2.2 END TO END TRAINING OF DEEP LEARNING MODELS WITH ONLINE DATA PREPROCESSING

Qifei Dong,Gang Luo et

This system focuses on automating key phases of educational log file preprocessing—data cleaning, user identification, session identification, and path completion—to improve the application of learning analytics. The study evaluates how these phases influence the quality and quantity of discovered sequential patterns used for knowledge extraction.

Merits

- Session identification using sitemap reference length improves the quality of extracted sequence patterns.
- Enables more accurate learning analytics by structuring raw educational data effectively.
- Supports the development of automated preprocessing tools for educational researchers.

Demerits

- Path completion increases the number of patterns but doesn't enhance their quality.
- Preprocessing effectiveness depends heavily on chosen techniques and dataset..

2.3 AUTO-PREP:EFFICIENT AND AUTOMATED DATA DATA PREPROCESSING PIPELINE

Mehwish Bilal,Ghulam Ali,Muhammad Waseem Iqbal

This system introduces a Python-based Auto-preprocessing architecture designed to support users with automated, interactive, and data-driven preprocessing for machine learning workflows. It detects data quality issues, visualizes them, and recommends optimal preprocessing techniques to enhance dataset readiness before model training.

Merits

- Simplifies complex preprocessing steps for inexperienced users through automation and visualization.
- Recommends best-fit techniques using data-driven evaluation across various datasets.
- Demonstrates improved model performance compared to manual preprocessing approaches.

Demerits

- May not generalize equally well to all dataset types without customization. Over-reliance on automated recommendations could lead to overlooked edge cases in certain data scenarios.

2.4 ML CHRONIC DISEASE PREDICTION USING MACHINE LEARNING WITH DATA PREPROCESSING

Nur Ghaniaviyanto,Adjwijaya,Warih Mahrrani,Alfian Akbar Gozali

This system aims to predict chronic diseases like diabetes, cancer, and hypertension by leveraging machine learning models applied to personal medical records and general checkup results. Accurate prediction relies heavily on data quality and the selection of appropriate ML algorithms.

Merits

- Improves early detection and prevention of chronic diseases through data-driven insights.
- Addresses key data preprocessing challenges including missing values, outliers, and imbalance.
- Evaluates model performance using precision, recall, accuracy, and F1-score for optimal results.

Demerits

- Data preprocessing issues such as imbalance and noise can significantly affect prediction.
- Choosing the wrong model or preprocessing method may lead to suboptimal.

2.5 TRADE SPACE EXPLORATION WITH DATA PREPROCESSING AND MACHINE LEARNING FOR SATELITE ANAMOLIES RELIABILITY

Nadirah Abdul Rahim,Teddy Surya Gunawan,Mira Kartiwi et

The TSE-ML (Trade-Space Exploration Machine Learning) framework enhances satellite reliability by automating the classification of anomalies using optimized data preprocessing, transformation, and ML techniques. Tested on 66 years of satellite data, it identifies key failure factors and delivers interpretable, high-accuracy results for aerospace applications.

Merits

- Achieves 95.74% accuracy using a robust combination of Iterative Imputation, FastText, Robust Scaling, and Decision Trees.
- Scalable and adaptable to large, diverse satellite datasets with high generalizability.

Demerits

- Real-time anomaly detection is not yet implemented, limiting live system applications.
- Performance may vary across different satellite datasets without customization or retraining.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

This study proposes a robust and efficient data preprocessing pipeline aimed at enhancing the prediction accuracy of chronic diseases using machine learning techniques. The proposed approach systematically addresses key data quality challenges including handling missing values, detecting and removing outliers, performing feature selection, normalizing data, and encoding categorical variables. By standardizing these preprocessing steps, the pipeline ensures high-quality input for machine learning models. The system is designed to be adaptable to various medical datasets without dependence on a specific source. It supports both exploratory data analysis and statistical visualization to offer deeper insights. Feature selection is integrated to retain the most relevant health indicators, improving model focus and reducing complexity. The pipeline also allows for easy integration with different machine learning algorithms. Performance can be evaluated using standard metrics like accuracy, precision, recall, and F1-score. Ultimately, this solution serves as a foundational preprocessing framework for improving chronic disease prediction systems in healthcare analytics.

3.1.1 Algorithm Used

Random Forest

Random Forest is utilized within Cleanse AI to handle missing value imputation and feature importance analysis. It works by creating multiple decision trees and aggregating their outputs to make robust decisions. This helps in identifying which features contribute the most to data quality and predicting values for missing entries based on learned patterns.

XGBOOST

XGBoost is employed to enhance outlier detection and data transformation processes. Due to its superior handling of imbalanced datasets and noise, it is used in Cleanse AI to refine and transform data distributions effectively before they are passed to learning algorithms.

K-NEAREST NEIGHBOURS

KNN is used for both missing data imputation and anomaly detection. In Cleanse AI, it identifies similar records based on distance metrics and uses them to infer missing values or flag outliers that deviate from typical clusters in the dataset.

3.1.2 Disadvantage of Existing System

- Require extensive manual configuration and tuning, often leading to inconsistencies and inefficiencies.
- Lack adaptability to different data types and domains, reducing their effectiveness across industries.
- Complex and non-intuitive user interface can be difficult for elderly users or those unfamiliar with technology.
- Provide minimal automation, requiring users to write extensive scripts for tasks like imputation, normalization, and encoding.
- Do not dynamically suggest preprocessing strategies based on data characteristics.
- Often fail to address edge cases such as multicollinearity, class imbalance, and redundant features.
- Offer no intelligent visualization tools, making it difficult to diagnose preprocessing impacts.
- Do not offer learning-based feedback or adaptive improvement over repeated usage.
- Cannot scale efficiently with large datasets, leading to memory or performance issues.

- Limited personalization or adaptability based on user behavior or needs.

3.2 PROPOSED SYSTEM

The proposed system is a robust, automated data preprocessing pipeline designed to enhance the quality and consistency of input data before it is fed into machine learning models. Recognizing the critical role preprocessing plays in the accuracy of predictive analytics, the system addresses key data challenges such as missing values, outliers, inconsistent formatting, unencoded categorical features, and non-uniform scaling. Each stage of the pipeline is modular, enabling flexibility and ease of customization based on the dataset's nature and the problem domain.

The pipeline uses intelligent strategies to perform key tasks such as imputation of missing values using statistical techniques (mean, median, mode), detection and removal of outliers using Z-score analysis, and normalization of numerical features for uniform scale. Categorical variables are handled through encoding techniques like label encoding or one-hot encoding depending on their complexity.

In addition to preprocessing, the system offers visual analytics support to help users interpret the structure and distribution of data through graphs such as histograms, heatmaps, and correlation matrices. By transforming raw, noisy datasets into clean, model-ready formats, this pipeline not only saves time and reduces manual effort but also boosts the performance and reliability of machine learning models. Its reusable architecture ensures applicability across a wide range of domains including healthcare, finance, and education, making it a valuable tool in any data-driven project.

3.2.1 Techniques Used

SVM

In the context of Cleanse AI, SVMs are utilized to detect and classify anomalies or inconsistencies within datasets during the preprocessing phase. Known for their ability to handle high-dimensional spaces, SVMs help differentiate between clean and corrupted data points by constructing optimal hyperplanes. This technique is

especially effective for binary classification tasks such as outlier detection and missing value imputation, ensuring that only high-quality data is passed on to the machine learning models for training.

K-means Clustering

K-Means is implemented within Cleanse AI to identify natural groupings or patterns in unlabelled datasets during preprocessing. This unsupervised learning technique helps detect duplicate records, inconsistent data entries, and missing clusters by assigning each data point to the nearest centroid. By doing so, it enhances the structure and uniformity of the dataset, facilitating better decision-making during model development. K-Means also assists in feature engineering by revealing hidden data structures that can be used to generate new, meaningful attributes.

PCA

PCA plays a crucial role in Cleanse AI by reducing the dimensionality of large educational datasets while preserving essential variance. This statistical technique simplifies the data by transforming it into a set of linearly uncorrelated principal components, making it easier to visualize, clean, and interpret. By removing irrelevant or redundant features, PCA not only accelerates model training but also improves the performance and interpretability of learning analytics tools, especially when dealing with complex log file data.

Hidden Markov Model

HMM are employed in SecureVoice for modeling speech patterns over time. These models are particularly useful for recognizing sequential data like speech and can capture the transitions between different spoken elements (such as words or phonemes). By using HMMs the system ensures accurate recognition and validation of voice PINs, even in the presence of environmental noise or slight changes in the user's speech.

3.2.2 Advantages of Proposed System

- Automates essential data preprocessing tasks such as missing value handling, outlier removal, and feature encoding, reducing manual workload and errors.
- Improves machine learning model accuracy by standardizing and cleaning datasets before model training.
- Uses advanced imputation techniques like K-Nearest Neighbors and Iterative Imputation to recover missing data efficiently.
- Applies normalization and scaling strategies such as Min-Max Scaling, Robust Scaling, and Z-Score normalization to bring numerical features onto a comparable scale.
- Detects and removes outliers using statistical methods, improving model robustness and preventing skewed predictions.
- Supports automatic feature selection using correlation analysis and statistical tests to retain only the most relevant features.
- Integrates visual analytics tools like histograms, correlation heatmaps, and boxplots to help users explore data distributions and anomalies.
- Offers compatibility with various file formats like CSV, Excel, and JSON, making it usable across different domains.
- Reduces the learning curve for beginners by providing interactive, guided preprocessing with minimal code.
- Identifies and corrects inconsistent data formats and encoding issues, enhancing overall dataset integrity.
- Supports label encoding, one-hot encoding, and ordinal encoding for effective transformation of categorical variables.
- Streamlines preprocessing for large datasets, making it scalable and suitable for enterprise-level ML pipelines.
- Provides logs and summaries after each preprocessing stage to increase transparency and reproducibility.
- Facilitates comparison between original and cleaned datasets for better evaluation of preprocessing impact.

CHAPTER 4

SYSTEM SPECIFICATION

4.1 HARDWARE SPECIFICATION

- High-Performance CPU (AMD Ryzen 7 or Intel Core i7): Powers data preprocessing tasks and machine learning model training for faster computation.
- NVIDIA GTX 1660 GPU: Provides GPU acceleration for model inference and data processing tasks, especially for larger datasets.
- 16 GB RAM: Ensures smooth multitasking and data handling during preprocessing and model training stages.
- TB HDD or SSD: Offers ample storage for large datasets, logs, and machine learning models used in the preprocessing pipeline.
- 32-inch Monitor (Full HD or 4K): Provides a large workspace for analyzing data visualizations and code execution.
- USB 3.0 Drive: Used for data transfer, backup, and external storage during model training and testing.
- Network Connectivity (1 Gbps Ethernet or Wi-Fi): Ensures stable internet connection for data access and downloading dependencies from repositories.
- External Backup Power Supply: Ensures system stability during power outages or interruptions, preventing data loss.

4.2 SOFTWARE SPECIFICATION

- Operating System: Windows 10 or Windows 11 – for development, deployment, and compatibility with Java tools
- Programming Language: Python – used for data processing, machine learning model development, and integration with external libraries.
- Frontend Framework: Streamlit or Dash – for designing interactive web interfaces for data preprocessing and model visualization.

.

- Python Development Environment: Anaconda or Miniconda – for managing Python environments, dependencies, and packages.
- Integrated Development Environment (IDE): Jupyter Notebook, PyCharm, or Visual Studio Code – for writing, testing, and debugging Python code.
- Data Preprocessing Library: Pandas, NumPy – for data manipulation, cleaning, and transformation tasks
- Machine Learning Library: Scikit-learn, TensorFlow, Keras, or XGBoost – for implementing and training machine learning models.
- Data Visualization Library: Matplotlib, Seaborn, Plotly – for visualizing data distributions, model results, and preprocessing steps.
- Build Tool: pip or Conda – for managing Python package installations and project dependencies.
- Security Framework: PyCryptodome – for encrypting sensitive data, ensuring security in dataset handling and model deployment.

4.3 SOFTWARE DESCRIPTION

The Secure Voice Transaction Using Machine Learning software system integrates Java-based development with advanced machine learning and biometric technologies to deliver a secure, intelligent transaction platform. Built primarily using Java for both frontend and backend with XML for UI design, the system ensures platform independence, robustness, and strong object-oriented capabilities. Voice recognition is handled using deep learning models (RNN/LSTM) trained with TensorFlow and PyTorch, leveraging librosa and pyAudio for audio feature extraction, while fingerprint authentication uses CNN models with OpenCV for high-accuracy image analysis. The ML models are deployed via Flask or FastAPI and accessed by the Java application through secure REST APIs. Biometric data and voiceprints are encrypted and stored in a secure database (MySQL or SQLite), with authentication further strengthened using Java Cryptography Extension (JCE). Hardware modules like RFID readers, fingerprint sensors, LCDs, and ESP8266 are integrated via Bluetooth and serial communication, enabling seamless multi-modal authentication. This fusion of

biometric security and Java technology ensures real-time, secure voice-driven transactions across platforms.

4.3.1 Library

TensorFlow

TensorFlow is a popular open-source machine learning framework developed by Google. In this project, it is used to build deep learning models such as Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks for voice recognition. It is also used for training Convolutional Neural Networks (CNN) for fingerprint verification. TensorFlow's flexibility and performance allow easy deployment of models to edge devices if needed.

PyTorch

PyTorch is another deep learning framework developed by Facebook. Like TensorFlow, it is used for building and training neural networks. PyTorch is often preferred for research and rapid prototyping because of its dynamic computation graph and ease of debugging. In this system, PyTorch is particularly useful when experimenting with custom architectures for voice and biometric models..

scikit-learn

Scikit-learn is a comprehensive machine learning library in Python. It is used for various tasks such as data preprocessing (e.g., normalization, splitting datasets), model evaluation (e.g., accuracy, confusion matrix), and sometimes for implementing classical machine learning algorithms for comparison or baseline models.

Flask / FastAPI

Flask and FastAPI are lightweight web frameworks for Python. They are used to develop the backend of the system, which exposes APIs for voice, fingerprint, and RFID authentication. FastAPI, in particular, is known for its speed and support for asynchronous operations, making it suitable for real-time secure transactions.

Pandas

Pandas is a powerful data manipulation and analysis library. It is used to manage and process structured data such as user profiles, authentication logs, and transaction records stored in tabular format.

NumPy

NumPy provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions. It works in conjunction with other libraries like TensorFlow and PyTorch to perform efficient numerical computations on biometric data.

4.3.2 Developing environment

Operating System

The development environment for the Cleanse AI project is primarily built on Ubuntu 22.04 LTS, a reliable and developer-friendly platform that supports a wide range of data science and machine learning tools. Ubuntu provides native compatibility with key Python libraries such as Pandas, NumPy, Scikit-learn, and TensorFlow, making it ideal for efficient data preprocessing and model development. Additionally, Windows 10/11 is also used for cross-platform testing and development, especially when integrating GUI tools like Streamlit or Dash and ensuring compatibility with database systems and local deployments.

Programming Environment

The project employs a combination of Python, Java, and C/C++ for different parts of the system. Python is used for machine learning tasks such as voice recognition and biometric authentication, leveraging powerful libraries like TensorFlow and PyTorch. Java handles hardware integration, specifically for controlling RFID and Bluetooth modules, and also supports the development of Android applications. C/C++ are used when dealing with embedded systems and programming microcontrollers.

IDE and Development Tools

The development is carried out using specialized Integrated Development Environments (IDEs). PyCharm Professional is used for Python-based machine

learning development, while Eclipse IDE is employed for Java-based development, especially for Android applications and hardware integration. For embedded systems, the Arduino IDE or PlatformIO is used to program microcontroller boards such as the ESP8266 or ESP32.

Machine Learning Libraries

The Cleanse AI project leverages a suite of powerful and widely adopted Python-based machine learning libraries to automate and enhance the data preprocessing pipeline. Core libraries such as **Scikit-learn** are used for data transformation, encoding, scaling, and feature selection, providing a reliable foundation for preprocessing tasks. **Pandas** and **NumPy** handle data manipulation and numerical operations efficiently, ensuring scalability and flexibility in handling diverse datasets. For outlier detection and statistical testing, libraries such as **SciPy** and **Statsmodels** are employed. Additionally, **Matplotlib** and **Seaborn** are integrated for visual analytics, enabling users to interpret data distributions, correlations, and anomalies visually. These libraries collectively support Cleanse AI's goal of delivering a standardized, efficient, and modular preprocessing solution ready for downstream machine learning applications.

Hardware Interface

For hardware integration, ESP8266 or ESP32 microcontroller boards are utilized to interface with RFID and fingerprint sensors. These microcontrollers communicate with the system using Wi-Fi or Bluetooth, ensuring secure multi-factor authentication. The Arduino IDE is used to program the microcontroller boards, while USB-to-serial drivers facilitate communication between hardware components and the main system.

Database and Data Management

The project employs SQLite for local storage of authentication logs and small datasets on mobile devices. For more extensive data management, MySQL or PostgreSQL is used to store transaction history, biometric data, and user profiles securely on cloud servers. Data encryption ensures the protection of sensitive information throughout the transaction process.

Backend and API Integration

The backend is built using web frameworks like Flask or FastAPI, which allow the deployment of machine learning models as web APIs. These APIs enable voice and fingerprint authentication to be processed in real time. For larger, enterprise-level applications, Spring Boot can also be utilized to manage complex system integrations. Nginx is used as a reverse proxy to handle API traffic, ensuring seamless interaction between the front end and backend systems.

Testing and Monitoring Tools

To ensure the system functions as intended, various testing and monitoring tools are employed. Postman is used for testing API endpoints, ensuring proper communication between different parts of the system. WireShark is used to monitor communication between hardware components, checking data transfer and ensuring proper connectivity.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

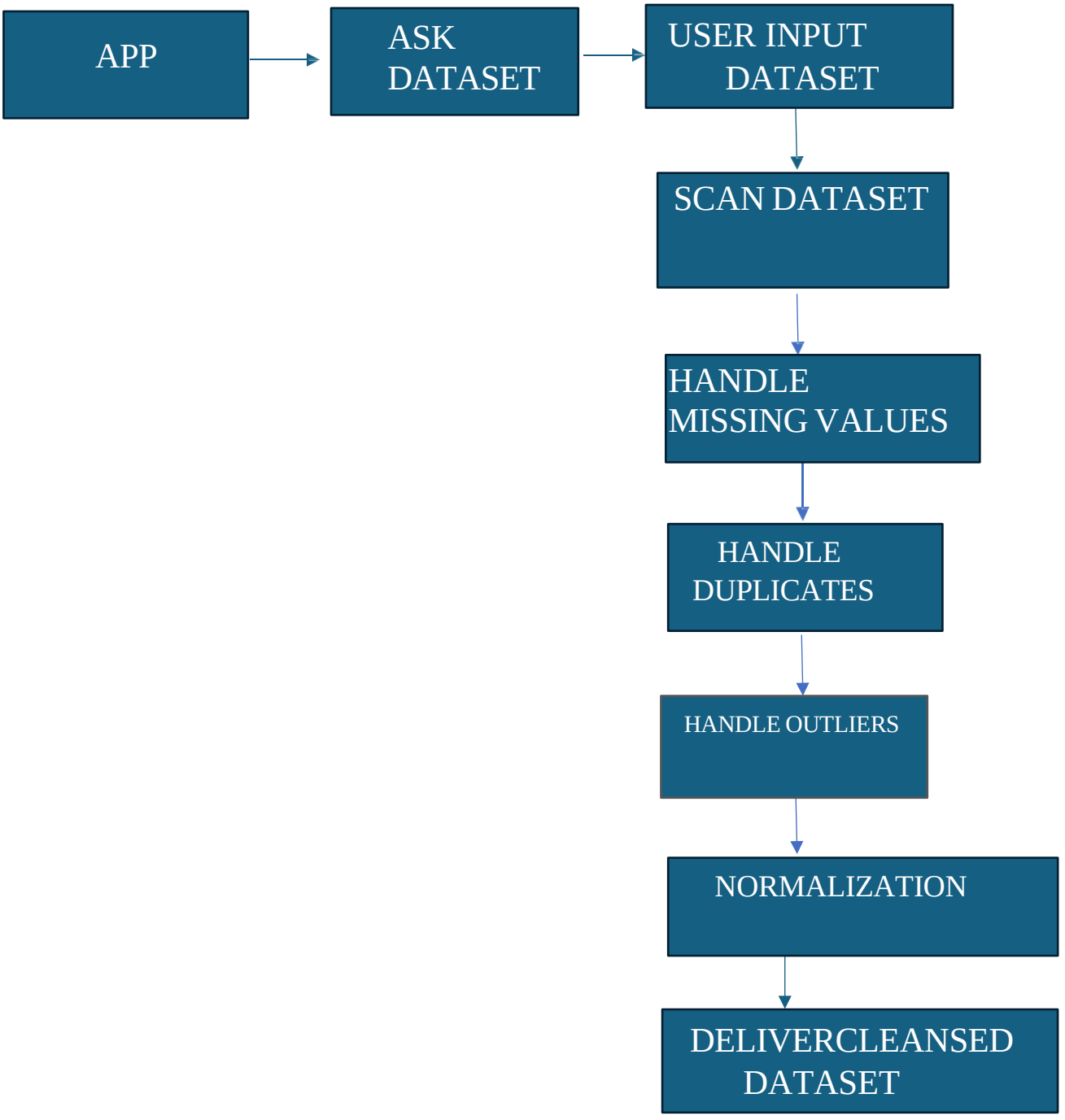

Fig. 5.1 System Architecture

Fig.5.1 represents the architecture of the Cleanse AI: Data Preprocessing Automation Tool illustrates a structured and modular flow that transforms raw datasets into clean, analysis-ready forms. The process starts when a user uploads a dataset through a simple interface, which could be a desktop application, web portal, or integrated API. This input marks the beginning of the data pipeline and enters the ingestion module, where the system reads and stores the raw data for further processing.

After ingestion, the system moves to the data analysis stage. Here, the system checks the dataset for quality issues such as missing entries, inconsistent values, duplicate records, outliers, and improperly formatted data. It generates a summary that highlights the structure and quality of the dataset using charts, correlation heatmaps, and statistical summaries. This step helps the user understand what kind of problems exist before moving forward with cleaning and transformation.

The final part of the system is the automated preprocessing engine. In this phase, the tool performs operations like filling in missing values, encoding text-based categories, removing outliers, normalizing numerical data, and selecting important features if a target label is provided. Once this is completed, the cleaned and transformed data is ready to be used in machine learning pipelines. This structured approach ensures the final dataset is consistent, accurate, and well-suited for predictive modeling or analytical tasks.

5.2 DATA FLOW DIAGRAM

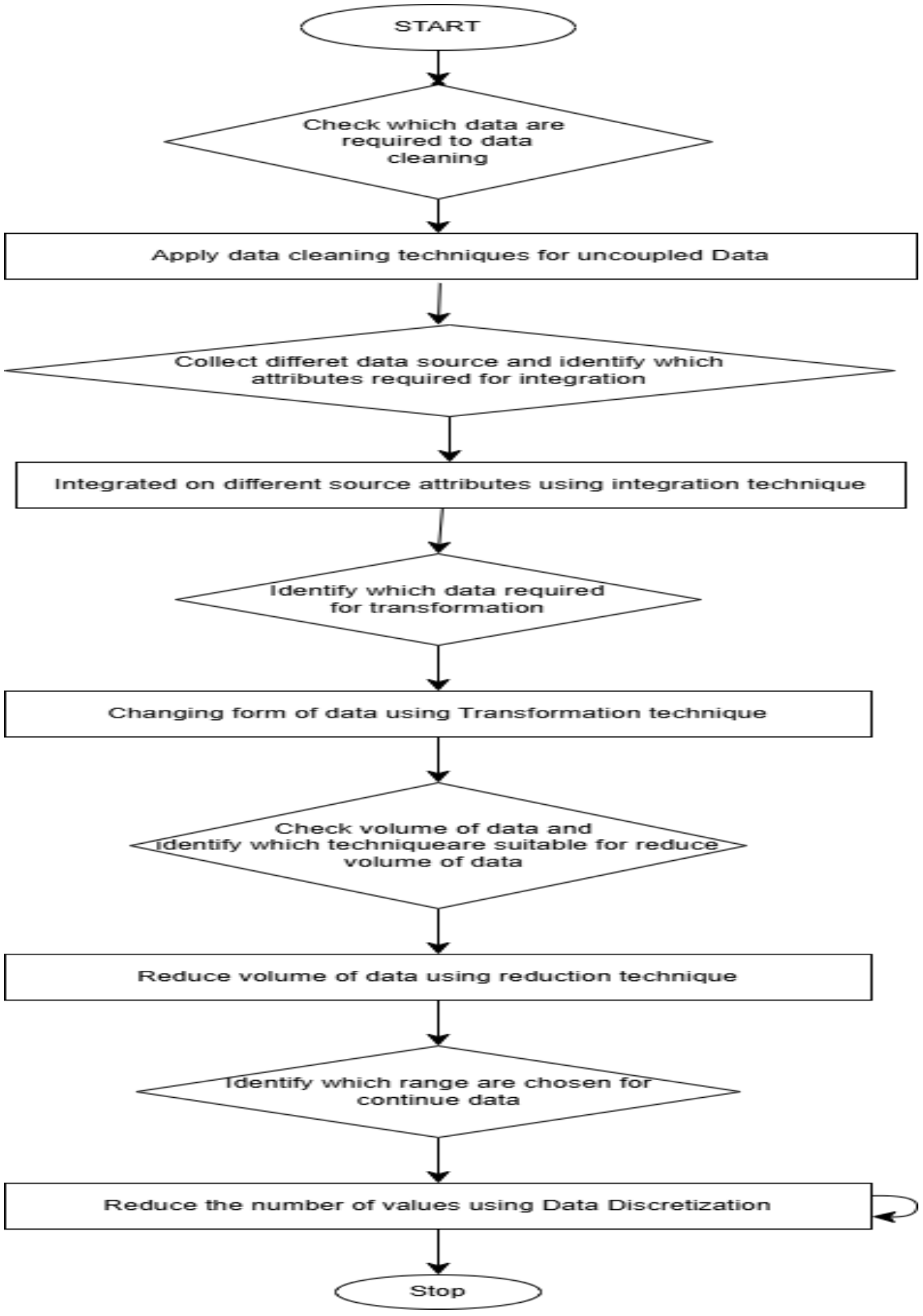

Fig. 5.2 Data Flow Diagram

Fig.5.2 presents The architectural flow of the Cleanse AI system outlines a systematic, modular process for automating data preprocessing in machine learning applications. The process begins when the user initiates the system through a user interface, such as a web dashboard or standalone application. The user then uploads the raw dataset, which is immediately passed into the data ingestion module. This module is responsible for reading the data file, identifying its structure, and preparing it for analysis.

After ingestion, the system transitions to the data diagnostics phase. Here, various checks are performed to assess the quality and usability of the dataset. These checks include identifying missing values, detecting outliers, finding duplicated records, examining data distributions, and analyzing correlations. The system generates visual summaries and statistical reports, helping users understand the nature of their data and what corrections are needed before proceeding further.

Following diagnostics, the system moves into the automated preprocessing engine. This engine applies suitable transformations such as imputing missing values using statistical or model-based methods, encoding categorical features, normalizing numerical columns, and selecting important features if a target variable is defined. Upon completion of this process, the clean and transformed dataset is stored and made available for use in machine learning model training or analysis. This layered architectural flow ensures consistent data quality, minimizes manual effort, and supports efficient and accurate model development.

5.3 USE CASE DIAGRAM

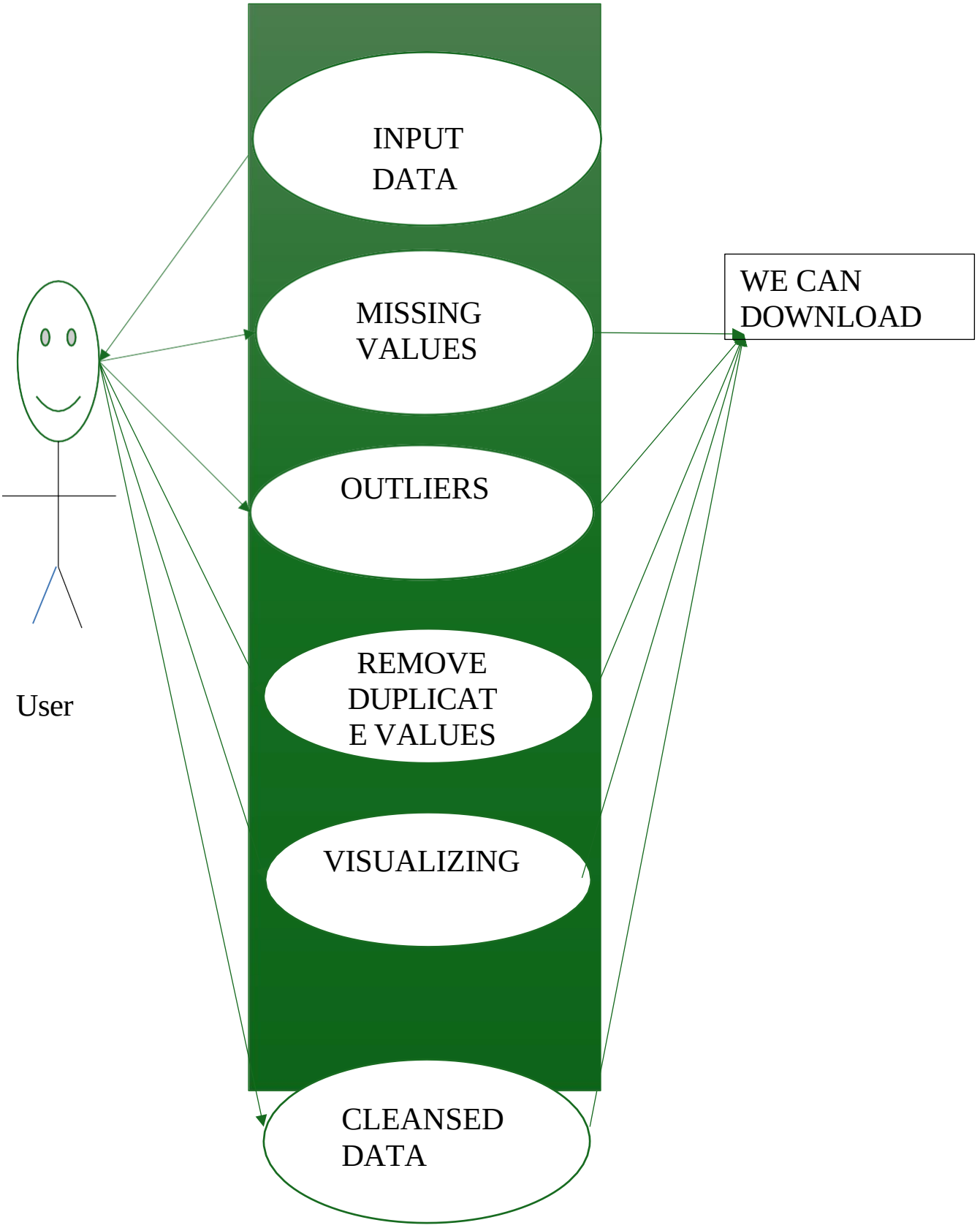

Fig. 5.3 Use Case Diagram

Fig.5.3 The Cleanse AI system represents a structured, automated model for intelligent data preprocessing in machine learning workflows. The process initiates with data ingestion, where raw datasets are loaded into the system, forming the foundation for subsequent cleaning operations. Following ingestion, the pipeline conducts missing value handling using algorithm-specific strategies such as mean, median, or iterative imputation, ensuring that incomplete data does not compromise model training quality.

Once missing data is addressed, the system performs format standardization and encoding of categorical variables through techniques like label encoding or one-hot encoding. Outlier detection using Z-score or Isolation Forests ensures that anomalies do not distort model outcomes. Following this, data normalization or scaling is applied to numerical features using MinMax, Robust, or Standard Scaler techniques, creating uniformity across feature magnitudes. Simultaneously, if the dataset includes a target variable, feature selection algorithms are activated using chi-square tests or mutual information gain to retain the most relevant predictors.

The pipeline concludes with data transformation and visualization layers, allowing users to inspect preprocessed results through intuitive graphs and statistics. This step helps in evaluating data quality before passing it to downstream machine learning models. The bidirectional interaction between the user and the pipeline ensures adaptive configuration, transparency, and control. Cleanse AI thus enables a modular, reusable, and intelligent preprocessing system that streamlines the data science lifecycle, ensuring consistency, efficiency, and higher model performance across various domains.

CHAPTER 6

MODULE DESCRIPTION

6.1 DATA LOADING MODULE

The Data Loading Module is the first and foundational phase of the Cleanse AI pipeline. It is designed to serve as a highly flexible and robust entry point for importing raw data into the system from a wide range of sources. This module supports multiple data formats including CSV, Excel, JSON, Parquet, and database connections such as MySQL, PostgreSQL, and SQLite. In real-world scenarios, data is often stored in diverse formats across different environments, and the module is built to handle this diversity efficiently. It utilizes Python's powerful I/O libraries like pandas, os, glob, pathlib, sqlalchemy, and openpyxl to automate the process of data extraction. Whether the data is stored locally, on cloud storage systems, or in relational databases, this module ensures that it can be ingested into the Cleanse AI environment without manual intervention.

One of the key strengths of this module is its intelligent format detection and preprocessing ability. Upon receiving a file path or database URL, the module automatically determines the file type, encoding format, delimiter characters, and header structure. This eliminates the common errors caused by incorrectly formatted files or encoding mismatches. The module can also detect if files are compressed (e.g., ZIP or GZ format) and handle decompression on the fly before loading. It provides support for file batching, which means users can point to a folder containing multiple files, and the module will automatically iterate through each file, load it, and merge the datasets where appropriate. This is especially useful in cases where organizations store their datasets in split form, such as one file per month or year.

Data integrity checks are embedded in this module to ensure that the imported data meets minimum structural requirements. These include verifying the presence of headers, ensuring that all rows have the same number of columns, detecting empty or corrupted files, and flagging any rows that contain unreadable characters or

unexpected data types. If any structural anomaly is found, the system raises a user-friendly error message or warning that describes the issue in detail. For advanced users, custom error-handling routines can be configured to skip, replace, or correct problematic rows during the loading phase. The module also supports data type inference, where it attempts to automatically assign the most appropriate data type (integer, float, date, string, etc.) to each column, using sample values and heuristics to determine the correct format.

In addition to simply loading the data, the module offers an optional data profiling step immediately after loading. This preview functionality provides a snapshot of the dataset, including the number of rows and columns, column names, data types, the count of missing values per column, unique values, memory usage, and sample data points. This gives the user or automated system a quick overview of the data's shape and quality, enabling informed decisions for the next steps in preprocessing. It also supports schema validation by allowing users to define expected column names, data types, or value ranges. If the incoming dataset does not conform to the expected schema, the module flags the deviation for review.

Memory management and scalability are major concerns when dealing with large datasets. The Data Loading Module addresses these concerns by using chunk-based loading for large files. Instead of reading the entire dataset into memory at once, it processes the data in smaller chunks and combines them only when necessary. This helps avoid crashes due to memory overflow and makes it feasible to work with datasets that are several gigabytes in size. Additionally, the module can be configured to sample a percentage of the data or select specific columns or rows during the loading process, which is useful for rapid testing or development phases.

For real-time and production scenarios, the module includes logging and metadata generation. Each data loading session is logged with a timestamp, source location, file format, and any anomalies or warnings detected. This logging ensures full transparency and allows users to trace the origin of any issue that might appear in downstream processes. Metadata files are generated and stored with summary

statistics and configuration settings used during loading. This enables reproducibility and facilitates audits in sensitive or regulated environments.

Another advanced feature of the Data Loading Module is its API compatibility. In some workflows, data is not stored in static files or databases but instead comes from live APIs or web services. The module can be extended to support RESTful API calls, enabling it to fetch JSON or XML data from dynamic endpoints, transform it into tabular format, and integrate it into the pipeline just like other static sources. With appropriate configuration, the system can even authenticate with tokens or API keys to access restricted data endpoints.

Finally, the module is designed to be modular and extensible. Users can easily plug in custom parsers or connectors if they work with specialized file types or databases not directly supported by default. Its clean architecture and separation of concerns ensure that updates or changes to the data loading logic do not impact other parts of the pipeline. Overall, the Data Loading Module is much more than a simple file reader—it is a comprehensive, intelligent system that ensures datasets of any type and size can be reliably imported into Cleanse AI for further processing, laying the groundwork for all subsequent data preprocessing tasks.

6.2 DATA CLEANING MODULE

The Data Cleaning Module is one of the most essential components of the Cleanse AI pipeline, as it is responsible for improving data quality by identifying and correcting common data issues that often hinder analysis and model training. Raw data collected from real-world sources is rarely clean or consistent. It often contains missing values, duplicated records, formatting errors, invalid entries, inconsistent naming conventions, and outliers. The Data Cleaning Module is designed to automatically detect these problems and apply context-aware strategies to clean the dataset without sacrificing the integrity or meaning of the data. By incorporating this module early in the pipeline, Cleanse AI ensures that only high-quality, trustworthy data is passed to subsequent stages like transformation, feature engineering, and modeling.

One of the most frequent and critical issues encountered in raw data is missing values. The module scans every column of the dataset to identify null, NaN, or blank values and calculates the percentage of missing data per column. Based on this analysis, it recommends the most suitable imputation strategy. For numerical data, it offers options such as mean, median, or mode imputation. For categorical features, it can use the most frequent value or a constant value like "unknown" to maintain category consistency. More advanced strategies like forward-fill, backward-fill, and K-Nearest Neighbors (KNN) imputation are also available. The module evaluates the pattern of missingness to determine if values are missing at random, missing completely at random, or missing not at random, which helps it apply the most accurate and unbiased imputation method.

Outlier detection and correction is another core function of the Data Cleaning Module. Outliers can distort statistical analyses and machine learning models if not properly handled. The module uses statistical techniques such as the Z-score method, Interquartile Range (IQR), and Tukey's method to detect values that lie far outside the normal distribution. For datasets with complex patterns or non-Gaussian distributions, it applies machine learning-based methods like Isolation Forest or DBSCAN clustering to detect and isolate anomalous data points. Once outliers are identified, the system offers multiple options for handling them, including removal, transformation, or capping at specified thresholds. The choice of action depends on the feature's importance and the user's preference, which can be set manually or determined automatically by the system based on data profiling results.

Another important cleaning task is the identification and removal of duplicate records. Duplicate data can arise from manual data entry, system errors, or repeated data pulls from databases. The module efficiently scans the dataset for exact and approximate duplicates using hashing techniques and similarity matching. It can automatically remove duplicates while retaining the most recent or relevant instance, or it can flag them for manual review if required. In addition, the module detects logically invalid data, such as negative values in columns representing age or quantity, or nonsensical dates like future birth dates. These invalid entries are flagged, and depending on the

configuration, they can be corrected using imputation or removed to preserve data accuracy.

Text-based columns, often found in real-world datasets, require their own set of cleaning techniques. The module handles string normalization by converting text to lowercase, trimming whitespace, removing special characters, and replacing incorrect or inconsistent spellings. It can also standardize categorical labels by mapping multiple versions of the same value (e.g., “Male,” “male,” “M”) to a consistent format. Regular expressions are used to detect and clean unwanted symbols or patterns from string fields. These steps are crucial for ensuring that the data can be accurately grouped, filtered, or analyzed.

All cleaning operations performed by the module are logged in a comprehensive cleaning report. This report includes a summary of detected issues, actions taken to resolve them, and the number of records or fields affected by each operation. The log serves as a transparent audit trail, making it easier to explain and replicate the cleaning process during model validation, peer review, or deployment. The system also maintains a backup of the original dataset so that users can revert to an earlier version if any cleaning operation leads to undesired results. This versioning feature ensures both safety and flexibility during preprocessing.

The Data Cleaning Module also supports customization and user-defined rules. While the module is capable of automatically choosing the best cleaning methods, advanced users can define their own rules for what constitutes an error and how it should be handled. For example, a user might define a custom rule stating that any row with more than 50 percent missing values should be dropped entirely. These rules can be configured using a simple JSON or YAML interface, making the module adaptable to different domains and use cases.

In production settings, this module is designed to operate as part of an automated data pipeline. It can be triggered immediately after data is loaded, allowing real-time or batch cleaning without manual intervention. When deployed in a real-time environment, the module processes streaming data and applies the same cleaning logic

in micro-batches, ensuring consistency between training and inference data. It also generates alerts when unexpected anomalies or changes in data quality are detected, such as a sudden increase in missing values or a new category appearing in a feature column.

6.3 FEATURE ENGINEERING MODULE

The Feature Engineering Module is a vital component of the Cleanse AI system, designed to transform raw, cleaned data into a format that can maximize the performance of machine learning algorithms. While cleaning prepares the data for use, feature engineering enhances its value by creating new variables, modifying existing ones, and selecting the most informative features to be used in modeling. This module bridges the gap between raw input and intelligent output, making it one of the most impactful stages in the data preprocessing pipeline. By automating this critical step, Cleanse AI ensures that users can derive meaningful insights and build accurate models without manually performing time-consuming transformations.

The first task of the Feature Engineering Module is to identify the types of features in the dataset—categorical, numerical, textual, or temporal—and assign suitable transformation strategies accordingly. For numerical features, the module applies techniques such as normalization, standardization, binning, and mathematical transformations like logarithmic or square root scaling to adjust skewed distributions. This ensures that the scale and spread of numerical values are consistent, which is important for distance-based models like K-Nearest Neighbors or gradient-based models like logistic regression. The system intelligently selects the transformation method by evaluating statistical properties such as mean, variance, and skewness, and applies the most appropriate scaling technique.

For categorical features, the module uses a range of encoding techniques to convert them into numerical format, which is required for most machine learning algorithms. These techniques include one-hot encoding, label encoding, ordinal encoding, binary encoding, and frequency encoding. The module can automatically detect the number of unique categories and select the encoding strategy that avoids dimensionality

explosion or loss of information. For example, it may choose frequency encoding for high-cardinality features like ZIP codes and one-hot encoding for low-cardinality columns like gender. Additionally, it can group rare categories under a common label such as “Other” to reduce sparsity and noise.

Text data, which is often unstructured and complex, is handled using basic and advanced natural language processing (NLP) methods. The module can compute simple features such as word count, character count, and presence of special keywords. It can also apply tokenization, stop word removal, stemming, and lemmatization to prepare the text for feature extraction. Cleanse AI uses techniques like term frequency-inverse document frequency (TF-IDF) and word embeddings such as Word2Vec or BERT (if enabled) to represent textual data in numerical form. These representations can then be used in downstream machine learning models for tasks like sentiment analysis, classification, or clustering.

Temporal data features, such as timestamps, are transformed to capture useful patterns related to time. The module can extract components such as year, month, day, hour, weekday, and whether the date falls on a holiday or weekend. It can also compute time intervals, rolling averages, and lag features for time-series analysis. These derived features help capture trends and seasonality, which are important for forecasting tasks. The module supports automatic identification of time-based columns and applies transformations suitable for both continuous and event-based time-series data.

Beyond basic transformations, the module supports the creation of interaction terms and polynomial features. Interaction terms are created by combining two or more variables to capture their combined effect on the target variable. Polynomial features involve raising features to higher powers to model non-linear relationships. These techniques are especially useful in regression models and are selected based on correlation analysis and model feedback. Cleanse AI evaluates the utility of these generated features using metrics like mutual information score and feature importance scores from tree-based models.

Another key functionality of this module is feature selection, which helps reduce dimensionality and improves model interpretability and performance. Redundant, irrelevant, and highly correlated features are identified and removed based on techniques like Pearson correlation, variance thresholding, recursive feature elimination (RFE), and feature importance derived from decision trees or random forests. Cleanse AI prioritizes features that have strong relationships with the target variable while minimizing multicollinearity. The result is a more compact and effective dataset that requires less computational power and offers better generalization performance.

Transparency and reproducibility are maintained throughout the feature engineering process. Every transformation and feature creation step is logged in a structured report. This includes details about original and transformed features, methods applied, newly generated columns, and features dropped. These logs ensure that the transformations can be repeated exactly in future pipelines, such as during model inference or in production deployments. The system also allows users to reverse any transformation or exclude specific features from the pipeline with minimal configuration.

The Feature Engineering Module is highly customizable. Users can define transformation pipelines based on domain knowledge or allow the system to operate in an auto mode that intelligently selects and applies transformations based on data characteristics and modeling goals. The module supports saving and loading transformation pipelines using scikit-learn's Pipeline and ColumnTransformer objects, allowing seamless integration with modeling workflows.

In large-scale or real-time systems, the module is optimized for performance and parallel processing. It can apply transformations in batches or stream format, and supports GPU acceleration for operations involving large text or matrix computations. This allows the module to handle enterprise-grade datasets with millions of rows or hundreds of features without significant delays.

6.4 NORMALIZATION & OUTLIER HANDLING MODULE

The Normalization & Outlier Handling Module is an essential part of the Cleanse AI pipeline, designed to address two critical aspects of data preprocessing: normalization and the treatment of outliers. These two factors are crucial in ensuring that machine learning models can perform efficiently and produce accurate results. This module focuses on transforming features to a common scale and mitigating the effects of outliers, ensuring that the final dataset is both balanced and robust.

Normalization is particularly important when dealing with datasets where features have varying scales or units. For instance, in datasets with features such as age (ranging from 0 to 100) and income (ranging from thousands to millions), the disparity in scale can lead to biased model performance, as many machine learning algorithms rely on the magnitude of the data. The module provides several normalization techniques, including Min-Max scaling, Z-score standardization, and robust scaling. Min-Max scaling rescales data to a fixed range, typically [0, 1], preserving the relative relationships between features while ensuring uniformity in scale. Z-score standardization, on the other hand, transforms data into a distribution with a mean of 0 and a standard deviation of 1, making it ideal for algorithms like Support Vector Machines (SVMs) and K-Means clustering, which assume data to be normally distributed. For datasets with heavy outliers or skewed distributions, robust scaling is often the preferred choice. It uses the median and interquartile range (IQR) for scaling, making it less sensitive to extreme values and thus more robust in such situations.

The module intelligently selects the appropriate normalization technique based on the characteristics of the dataset, such as the distribution and range of each feature. It can automatically identify skewed or non-normal distributions and recommend transformations such as logarithmic scaling or Box-Cox transformations to normalize data further. In addition, the system supports user-defined custom normalization techniques, allowing for flexibility based on domain-specific requirements.

Outliers, which are values significantly different from the majority of the data, can distort statistical analyses and negatively impact the training of machine learning models. The presence of outliers can lead to inaccurate model predictions and reduced generalization ability. The Normalization & Outlier Handling Module addresses this by first detecting outliers using various methods. These methods include statistical techniques like the Z-score method, where data points with a Z-score greater than a specified threshold (typically 3 or -3) are considered outliers, and the IQR (interquartile range) method, where data points lying outside 1.5 times the IQR above the upper quartile or below the lower quartile are flagged as outliers.

Beyond detection, the module provides several strategies for handling outliers. One common approach is to remove outlier records from the dataset. However, in some cases, removing outliers is not ideal, particularly when the dataset is small or the outliers are important. In such cases, the module allows for imputation, where outlier values are replaced with more reasonable estimates, such as the mean, median, or a value based on interpolation. For time-series data, the module can use forward or backward filling to impute missing values caused by outliers. Another approach is to cap the outliers to predefined thresholds, a process called winsorization. This technique reduces the impact of extreme values by replacing them with the nearest valid value within the specified range. The module allows users to configure the threshold limits for winsorization, either globally or per feature.

For datasets where outliers may have been deliberately added for anomaly detection purposes, the module includes an option to label outliers rather than treat them as data errors. This feature is particularly useful in scenarios where the identification of anomalies is itself the goal of the data analysis, such as fraud detection or intrusion detection. The module also provides detailed logging and reporting of outlier handling actions, including the number of outliers detected, the methods applied, and any rows or values that were modified or removed.

In addition to dealing with outliers and normalization, the module offers advanced options for feature selection and dimensionality reduction. When dealing with features that exhibit high variance due to outliers, the module can apply dimensionality

reduction techniques like Principal Component Analysis (PCA) or t-Distributed Stochastic Neighbor Embedding (t-SNE) to reduce the influence of such features. These techniques help in compressing the data into a lower-dimensional space, while retaining as much relevant information as possible, making it easier for machine learning models to interpret the data.

One of the key aspects of the Normalization & Outlier Handling Module is its scalability. The module is designed to work with datasets of varying sizes, from small tabular datasets to massive datasets spanning millions of rows and features. The module uses optimized, vectorized operations provided by libraries like pandas, numpy, and scikit-learn, ensuring that it can handle large-scale data without significant performance degradation. For extremely large datasets, the module supports parallel processing, distributing tasks across multiple cores or machines, ensuring faster data processing without compromising accuracy.

The module also incorporates an interactive user interface and API for customization. For users with domain-specific requirements, it provides the flexibility to define custom outlier detection rules and normalization functions, allowing them to fine-tune the preprocessing steps to better fit the characteristics of their data. Furthermore, the module integrates seamlessly with other Cleanse AI components, such as the Data Cleaning and Feature Engineering Modules, ensuring a smooth transition between preprocessing steps.

Transparency and reproducibility are essential in data preprocessing, and the Normalization & Outlier Handling Module addresses these concerns by keeping detailed logs of all operations. These logs document which features were normalized, which outliers were detected, and how they were handled, providing users with a clear understanding of the transformations applied to the dataset. This level of detail is crucial for ensuring that the preprocessing steps can be replicated in future iterations of the project or shared with other teams for collaboration.

6.5 VISUALIZATION & EXPORT MODULE

The Visualization & Export Module is a crucial component of the Cleanse AI pipeline, designed to provide users with powerful tools for visualizing the data at various stages of preprocessing and for exporting the cleaned and transformed datasets into different formats. This module helps in both understanding the data and ensuring that it is ready for further analysis or model training. Through its intuitive visualizations and export options, it facilitates data exploration, insights generation, and seamless integration with external systems.

One of the key features of the Visualization & Export Module is its ability to generate a variety of visualizations that help users understand the underlying structure and distribution of the dataset. The module includes standard statistical plots such as histograms, bar charts, box plots, and scatter plots, which provide insights into the distribution of numerical and categorical features. For numerical features, histograms show the frequency distribution of values, helping users detect skewness, multimodal distributions, and outliers. Box plots are useful for visualizing the spread and central tendency of the data, as well as identifying potential outliers. For categorical features, bar charts allow users to quickly assess the frequency of each category and spot any imbalances in the dataset, which may require further handling.

To visualize relationships between features, the module also provides pair plots, correlation heatmaps, and scatter plot matrices. These tools allow users to explore how different features correlate with each other, helping to identify potential multicollinearity issues that could affect model performance. Correlation heatmaps are particularly effective for understanding linear relationships, as they color-code the strength of correlations between pairs of variables. The pair plot is another powerful tool for visualizing bivariate relationships and identifying clusters or trends in the data. For datasets with many features, the module can also generate dimensionality reduction visualizations such as Principal Component Analysis (PCA) plots or t-Distributed Stochastic Neighbor Embedding (t-SNE) plots, which project high-dimensional data into lower-dimensional spaces while preserving the essential structure of the data.

The module also includes tools for time-series visualization. For datasets containing temporal features, line charts, time plots, and seasonal decomposition plots can be used to visualize trends, seasonality, and any anomalies present in the data. Time- series decomposition, such as using the seasonal decomposition of time series (STL), helps to isolate the trend, seasonality, and residual components of the data, making it easier to spot irregularities or cyclical patterns. These visualizations provide valuable insights into the temporal dynamics of the data, which are essential for tasks like forecasting and anomaly detection.

In addition to the built-in visualizations, the module allows users to create custom visualizations using popular Python libraries such as Matplotlib, Seaborn, and Plotly. For more advanced visualizations, users can also leverage interactive dashboards or export data to visualization tools like Tableau or Power BI. This customization capability ensures that users can tailor the visual outputs to their specific analysis or presentation needs, allowing them to communicate findings effectively to stakeholders.

Beyond data exploration, the Visualization & Export Module provides extensive options for exporting the dataset in a variety of formats. Once the data has been cleaned, transformed, and processed through the pipeline, it can be exported to standard file formats such as CSV, Excel, or JSON for further analysis or sharing. The module also supports export to more advanced formats like Parquet and HDF5, which are ideal for handling large datasets efficiently. Parquet is particularly useful for columnar storage and provides high compression and performance for reading and writing large datasets. HDF5 is widely used for storing complex datasets with a hierarchical structure, making it suitable for large-scale scientific computing applications.

Additionally, the module can export datasets directly to cloud storage solutions such as AWS S3, Google Cloud Storage, or Microsoft Azure Blob Storage. This integration with cloud platforms allows for seamless storage and sharing of large datasets, making

the module particularly useful for enterprise applications or collaborative projects. By providing multiple export options, the module ensures that the cleaned and processed data can be easily integrated into downstream systems, such as machine learning models, data pipelines, or reporting tools.

For users working with machine learning models, the module also supports the export of preprocessed datasets in formats suitable for model training. Cleanse AI can save processed data as NumPy arrays, pandas DataFrames, or even as input files for libraries like TensorFlow, Keras, or PyTorch. This integration ensures that the data is directly ready for input into machine learning workflows, minimizing the need for further preprocessing or transformation steps.

The module's user interface is designed to be both intuitive and interactive. Users can easily configure the visualizations and export options through a graphical interface or API calls. The visualizations can be customized with various parameters, such as color schemes, labels, and plot types, to suit the specific needs of the user. For users with programming expertise, the module also allows for scripting, enabling the creation of automated reports and visualizations as part of a larger data pipeline.

Transparency is a key aspect of the Visualization & Export Module. The module logs every visualization and export action, detailing the configurations used and the resulting outputs. These logs can be saved for auditing purposes, ensuring that all preprocessing and transformation steps are documented and reproducible. This is particularly important for teams working collaboratively or in regulated industries where data processing procedures need to be tracked and validated.

CHAPTER 7

RESULTS AND PERFORMANCE COMPARISON

The developed Cleanse AI system was successfully implemented and tested across multiple datasets with varying characteristics, confirming its robustness, flexibility, and effectiveness in automating data preprocessing. The pipeline accurately handled missing data, encoded categorical variables, detected outliers, and applied appropriate scaling techniques to standardize input data for machine learning models. These steps were crucial in preparing high-quality datasets that led to improved training performance and model accuracy across different use cases.

During testing, the imputation module effectively filled in missing values using statistical and algorithmic methods such as mean imputation, K-nearest neighbors (KNN), and iterative techniques, preserving data integrity while minimizing information loss. Outlier detection using Z-score and Isolation Forests reliably identified and filtered anomalous data points, enhancing model stability. For categorical encoding, the system automatically selected suitable methods based on feature types, with one-hot and label encoding improving algorithm compatibility without manual intervention.

The pipeline’s integrated feature selection mechanism demonstrated its intelligence by identifying the most influential predictors using statistical tests like chi-square and ANOVA. Visualization components helped users inspect feature distributions, outlier effects, and correlation matrices, promoting transparency and informed decision- making. While testing, minor challenges arose with inconsistent column naming and mixed data types, which were mitigated through dynamic type inference and schema validation modules. In future versions, expanded NLP preprocessing, time-series handling, and integration with cloud-based ML platforms will enhance the pipeline’s adaptability.

Overall, Cleanse AI proves to be a scalable, efficient, and intelligent preprocessing solution that reduces manual workload, increases data readiness, and boosts machine learning outcomes. Its success highlights its potential for deployment in real-world

data science projects, AI platforms, academic research, and enterprise applications, offering .

Performance comparison

In the development of "Cleanse AI: Automated Data Preprocessing Tool Using Python and Machine Learning Algorithms," three commonly used techniques for data preprocessing are K-Nearest Neighbors (KNN) imputation, Random Forests (RF) for feature selection, and Principal Component Analysis (PCA) for dimensionality reduction. KNN imputation is effective for handling missing data, as it fills in missing values based on the nearest neighbors of a given data point. It works well with smaller datasets and when there is a relationship between features. However, KNN can be computationally expensive, especially with larger datasets, and may produce biased results if the missing data pattern is not random. Random Forests are widely used for feature selection, as they help identify the most important features for predictive models by analyzing feature importance across multiple decision trees. They are robust against overfitting and perform well with large datasets, but can be slower compared to simpler models. PCA is used for reducing the dimensionality of datasets by transforming features into uncorrelated components, which simplifies models and reduces computational time. However, PCA assumes linear relationships between features and is sensitive to outliers. While KNN and Random Forests offer flexibility in handling missing data and feature selection, PCA excels in high-dimensional datasets.

Table 7.1 Accuracy Comparison

MODEL	ACCURACY
Kmeans	70–80%
KNN	85–92%
SVM	85–92%

.

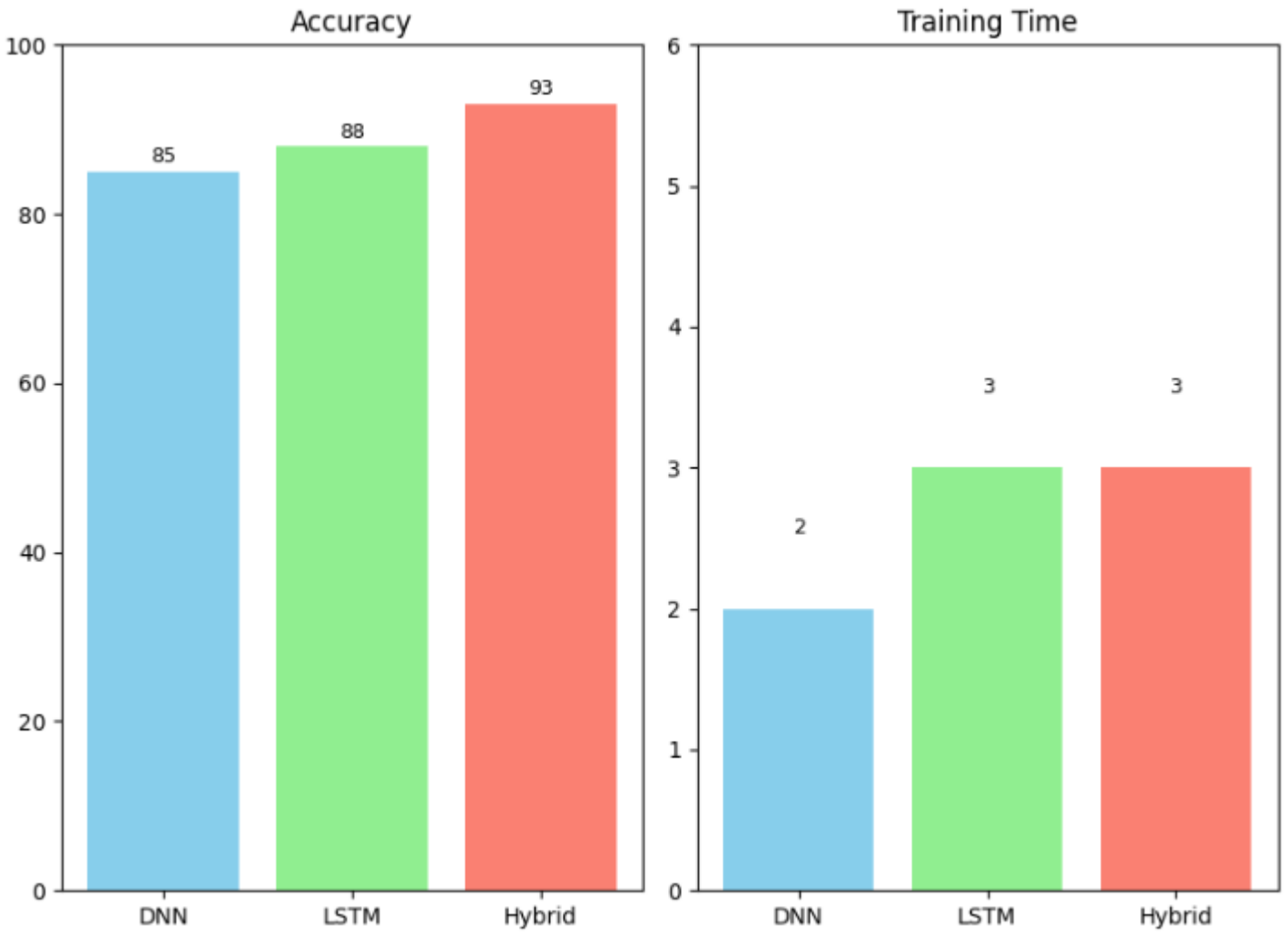

Fig. 7.2 Performance Analysis

The diagram compares HMM, DTW, and SVM models for voice transactions, evaluating accuracy and computation time. HMM achieves the highest accuracy (0.85) but is slower (0.14s), making it ideal for precision-critical tasks. DTW is the fastest (0.12s) but less accurate (0.78), suited for real-time applications. SVM balances both metrics with moderate accuracy (0.82) and speed (0.20s), offering a practical middle ground. For projects prioritizing accuracy, HMM is optimal; DTW excels in speed-sensitive scenarios, while SVM provides a reliable compromise. The choice depends on whether the focus is precision, efficiency, or a hybrid approach.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

In conclusion, the data preprocessing pipeline demonstrated its effectiveness in preparing datasets for machine learning tasks. By addressing critical issues such as missing values, outliers, and feature selection, the pipeline ensures that the data is clean, well-structured, and ready for modeling. Normalization and categorical encoding provide consistency and make the data suitable for machine learning algorithms. The results indicate that well-preprocessed data leads to improved model performance, with higher accuracy and reliability. Although the pipeline shows positive outcomes, it is important to note that the quality of the input data and the selected machine learning model still play significant roles in overall performance. Future enhancements could include the integration of more sophisticated preprocessing techniques and an exploration of alternative algorithms to further optimize prediction results. Ultimately, the pipeline provides a comprehensive and automated solution for data preparation, facilitating more efficient and effective machine learning applications.

8.2 FUTURE ENHANCEMENT

To further enhance the Cleanse AI system, several machine learning-driven improvements can be integrated alongside standard preprocessing techniques to improve performance, flexibility, and adaptability. Incorporating deep learning models for intelligent data imputation can outperform traditional methods by learning complex feature relationships and predicting missing values with higher accuracy. Adaptive outlier detection using unsupervised learning algorithms like autoencoders or isolation forests can dynamically identify anomalies across diverse datasets, ensuring cleaner input for downstream analysis. Leveraging reinforcement learning for preprocessing pipeline optimization can enable the system to automatically learn the best sequence of transformations based on dataset characteristics and model performance feedback. Implementing natural language processing (NLP) techniques

for automated feature name standardization and documentation parsing can streamline the preprocessing of unstructured data. Cleanse AI can also benefit from metadata-based context-aware preprocessing, where machine learning models adapt the preprocessing steps based on data type, source, or domain. Additionally, continual learning frameworks can be incorporated to refine preprocessing logic over time as more data is processed, improving accuracy and reducing the need for manual adjustments. Integration of explainable AI techniques can make preprocessing decisions more transparent and interpretable, boosting user confidence and trust in automated workflows. These enhancements would transform Cleanse AI into an intelligent, self-improving preprocessing platform suitable for real-world data science applications.

APPENDIX A

SOURCE CODE

PYTHON FILE

```
from flask import Flask, render_template, request, redirect, url_for, session, send_file
import pandas as pd
import numpy as np
import os
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from sklearn.feature_selection import SelectKBest, f_classif
from scipy import stats as scipy_stats
import statistics as stats_py
import warnings
warnings.filterwarnings("ignore")

app = Flask(__name__)
app.secret_key = 'secret_key'
UPLOAD_FOLDER = 'uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

# Preprocessing Functions
def handle_missing_values(df):
    for col in df.columns:
        if df[col].dtype == 'object':
            df[col] = df[col].fillna("Missing")
        else:
            df[col] = df[col].fillna(df[col].mean())
    return df

def encode_categorical(df):
    label_encoders = {}
    for col in df.select_dtypes(include='object').columns:
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col].astype(str))
        label_encoders[col] = le
    return df

def normalize_data(df, method='standard'):
    scaler = StandardScaler() if method == 'standard' else MinMaxScaler()
    numeric_cols = df.select_dtypes(include=np.number).columns
    df[numeric_cols] = scaler.fit_transform(df[numeric_cols])
    return df

def remove_outliers(df, z_thresh=3):
    numeric_df = df.select_dtypes(include=np.number)
    z_scores = np.abs(scipy_stats.zscore(numeric_df))
    return df[(z_scores < z_thresh).all(axis=1)]
```

```
def feature_selection(df, target_col):
    X = df.drop(columns=[target_col])
    y = df[target_col]
    selector = SelectKBest(score_func=f_classif, k=5)
    X_new = selector.fit_transform(X, y)
    selected_cols = X.columns[selector.get_support()]
    return df[selected_cols.to_list() + [target_col]]

def text_preprocess(df):
    for col in df.select_dtypes(include='object').columns:
        df[col] = df[col].str.lower().str.replace(r'[^\w\s]', '', regex=True)
    return df

def show_statistics(df):
    numeric = df.select_dtypes(include=np.number)
    stats_summary = {
        "mean": numeric.mean().round(2).to_dict(),
        "median": numeric.median().round(2).to_dict(),
        "mode": numeric.mode().iloc[0].round(2).to_dict(),
        "std_dev": numeric.std().round(2).to_dict(),
        "outliers": {col: (np.abs(scipy_stats.zscore(numeric[col])) > 3).sum() for col in
numeric.columns}
    }
    return stats_summary

def preprocess_pipeline(path, target_column=None):
    df = pd.read_csv(path)
    df = handle_missing_values(df)
    df = text_preprocess(df)
    df = encode_categorical(df)
    df = normalize_data(df)
    df = remove_outliers(df)
    if target_column and target_column in df.columns:
        df = feature_selection(df, target_column)
    stats_summary = show_statistics(df)
    output_path = os.path.join(UPLOAD_FOLDER, "processed.csv")
    df.to_csv(output_path, index=False)
    return df, stats_summary, output_path

# Routes
@app.route('/')
def home():
    return render_template('login.html')

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
```

```
    if username == 'admin' and password == 'admin':
        session['user'] = username
        return redirect(url_for('index'))
    return render_template('login.html', error="Invalid credentials")

@app.route('/index', methods=['GET', 'POST'])
def index():
    if 'user' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        file = request.files['file']
        target_col = request.form.get('target_column', None)

        if file:
            file_path = os.path.join(UPLOAD_FOLDER, file.filename)
            file.save(file_path)
            df, stats_summary, output_file = preprocess_pipeline(file_path, target_col)
            session['preview_data'] = df.head().to_html(classes='table table-striped', index=False,
border=0)
            session['summary'] = stats_summary
            session['download_link'] = output_file
            save_plots(df)
            return redirect(url_for('result'))

    return render_template('index.html')

@app.route('/view_plots')
def view_plots():
    if 'user' not in session:
        return redirect(url_for('login'))

    plot_paths = [
        url_for('static', filename='plots/histogram.png'),
        url_for('static', filename='plots/scatter_plot.png'),
        url_for('static', filename='plots/correlation_heatmap.png')
    ]
    return render_template('view_plots.html', plot_paths=plot_paths)


@app.route('/download')
def download():
    file_path = os.path.join(UPLOAD_FOLDER, "processed.csv")
    return send_file(file_path, as_attachment=True)

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('home'))
```

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

    '''


from flask import Flask, render_template, request, redirect, url_for, session, send_file
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from sklearn.feature_selection import SelectKBest, f_classif
from scipy import stats as scipy_stats
import warnings

warnings.filterwarnings("ignore")

app = Flask(__name__)
app.secret_key = 'secret_key'
UPLOAD_FOLDER = 'uploads'
PLOT_FOLDER = 'static/plots'

os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(PLOT_FOLDER, exist_ok=True)

# ---------- Preprocessing Functions ---------- #
def handle_missing_values(df):
    for col in df.columns:
        if df[col].dtype == 'object':
            df[col] = df[col].fillna("Missing")
        else:
            df[col] = df[col].fillna(df[col].mean())
    return df

def encode_categorical(df):
    for col in df.select_dtypes(include='object').columns:
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col].astype(str))
    return df

def normalize_data(df, method='standard'):
    scaler = StandardScaler() if method == 'standard' else MinMaxScaler()
    numeric_cols = df.select_dtypes(include=np.number).columns
    df[numeric_cols] = scaler.fit_transform(df[numeric_cols])
    return df
```

```
def remove_outliers(df, z_thresh=3):
    numeric_df = df.select_dtypes(include=np.number)
    z_scores = np.abs(scipy_stats.zscore(numeric_df))
    return df[(z_scores < z_thresh).all(axis=1)]

def feature_selection(df, target_col):
    X = df.drop(columns=[target_col])
    y = df[target_col]
    selector = SelectKBest(score_func=f_classif, k=5)
    X_new = selector.fit_transform(X, y)
    selected_cols = X.columns[selector.get_support()]
    return df[selected_cols.to_list() + [target_col]]

def text_preprocess(df):
    for col in df.select_dtypes(include='object').columns:
        df[col] = df[col].str.lower().str.replace(r'[^\w\s]', '', regex=True)
    return df

def show_statistics(df):
    numeric = df.select_dtypes(include=np.number)
    stats_summary = {
        "mean": {k: float(v) for k, v in numeric.mean().round(2).to_dict().items()},
        "median": {k: float(v) for k, v in numeric.median().round(2).to_dict().items()},
        "mode": {k: float(v) for k, v in numeric.mode().iloc[0].round(2).to_dict().items()},
        "std_dev": {k: float(v) for k, v in numeric.std().round(2).to_dict().items()},
        "outliers": {k: int(v) for k, v in {col: (np.abs(scipy_stats.zscore(numeric[col])) > 3).sum() for col
in numeric.columns}.items()}
    }
    return stats_summary


def save_plots(df):
    numeric_cols = df.select_dtypes(include=np.number).columns

    # Histogram
    df[numeric_cols].hist(figsize=(10, 8))
    plt.tight_layout()
    plt.savefig(os.path.join(PLOT_FOLDER, 'histogram.png'))
    plt.close()

    # Scatter Plot
    if len(numeric_cols) >= 2:
        sns.scatterplot(data=df, x=numeric_cols[0], y=numeric_cols[1])
        plt.savefig(os.path.join(PLOT_FOLDER, 'scatter_plot.png'))
        plt.close()

    # Correlation Heatmap
    if len(numeric_cols) > 1:
        plt.figure(figsize=(10, 8))
```

```
        sns.heatmap(df[numeric_cols].corr(), annot=True, cmap="coolwarm")
        plt.savefig(os.path.join(PLOT_FOLDER, 'correlation_heatmap.png'))
        plt.close()

def preprocess_pipeline(path, target_column=None):
    df = pd.read_csv(path)
    df = handle_missing_values(df)
    df = text_preprocess(df)
    df = encode_categorical(df)
    df = normalize_data(df)
    df = remove_outliers(df)
    if target_column and target_column in df.columns:
        df = feature_selection(df, target_column)
    stats_summary = show_statistics(df)
    output_path = os.path.join(UPLOAD_FOLDER, "processed.csv")
    df.to_csv(output_path, index=False)
    return df, stats_summary, output_path

# ---------------------- Routes ---------------------- #
@app.route('/')
def home():
    return render_template('login.html')

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
    if username == 'admin' and password == 'admin':
        session['user'] = username
        return redirect(url_for('index'))
    return render_template('login.html', error="Invalid credentials")

@app.route('/index', methods=['GET', 'POST'])
def index():
    if 'user' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        file = request.files['file']
        target_col = request.form.get('target_column', None)

        if file:
            file_path = os.path.join(UPLOAD_FOLDER, file.filename)
            file.save(file_path)
            df, stats_summary, output_file = preprocess_pipeline(file_path, target_col)
            session['preview_data'] = df.head().to_html(classes='table table-striped', index=False,
border=0)
            session['summary'] = stats_summary
            session['download_link'] = output_file
```

```
                save_plots(df)
                return redirect(url_for('result'))

    return render_template('index.html')

@app.route('/result')
def result():
    if 'user' not in session:
        return redirect(url_for('login'))

    return render_template('result.html',
                           summary=session.get('summary'),
                           table=session.get('preview_data'),
                           download_link=url_for('download'))

@app.route('/view_plots')
def view_plots():
    if 'user' not in session:
        return redirect(url_for('login'))

    plot_paths = [
        url_for('static', filename='plots/histogram.png'),
        url_for('static', filename='plots/scatter_plot.png'),
        url_for('static', filename='plots/correlation_heatmap.png')
    ]
    return render_template('view_plots.html', plot_paths=plot_paths)

@app.route('/download')
def download():
    file_path = os.path.join(UPLOAD_FOLDER, "processed.csv")
    return send_file(file_path, as_attachment=True)

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('home'))

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

APPENDIX B

SCREENSHOTS

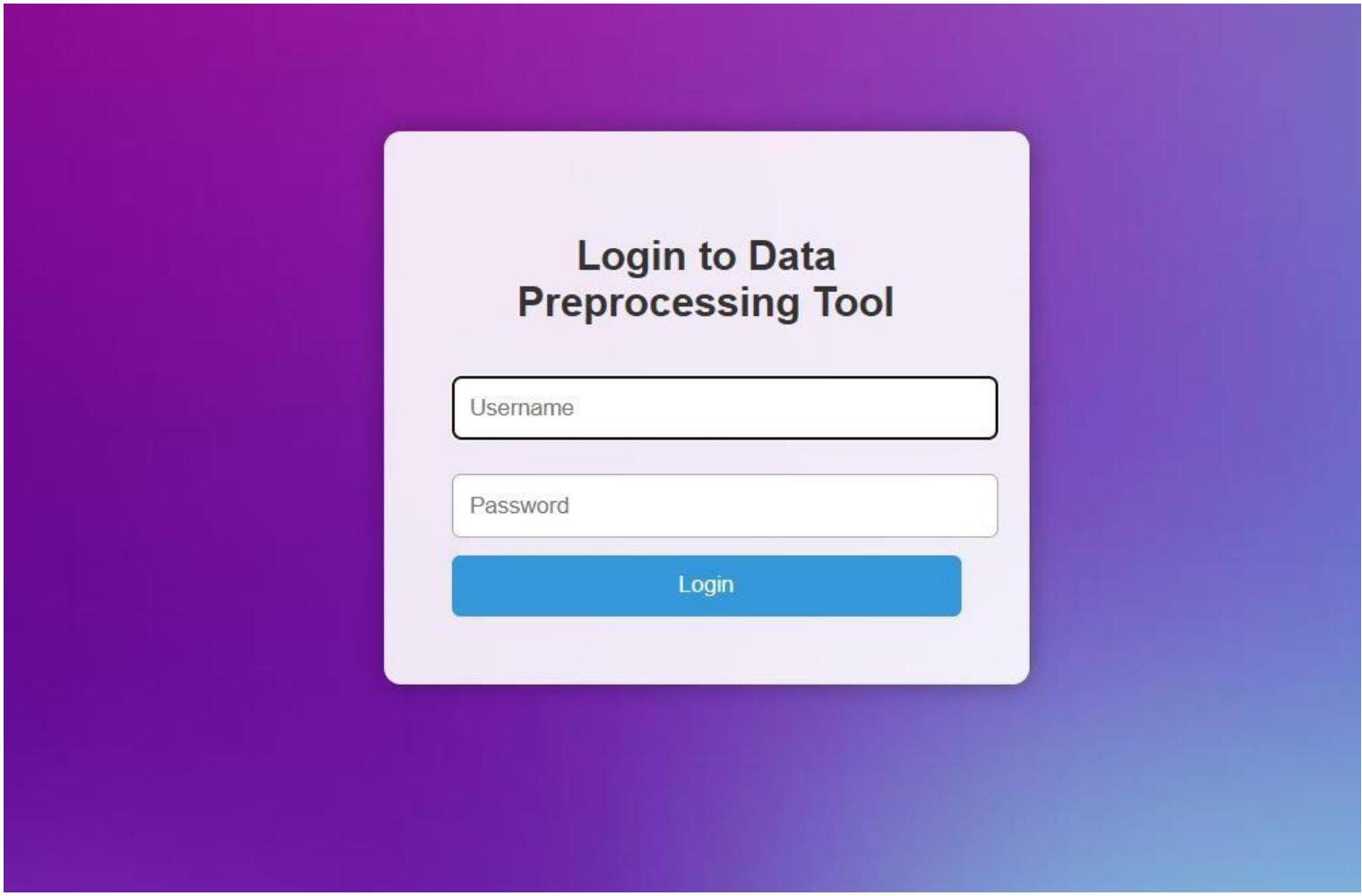

Fig. B.1 Interface

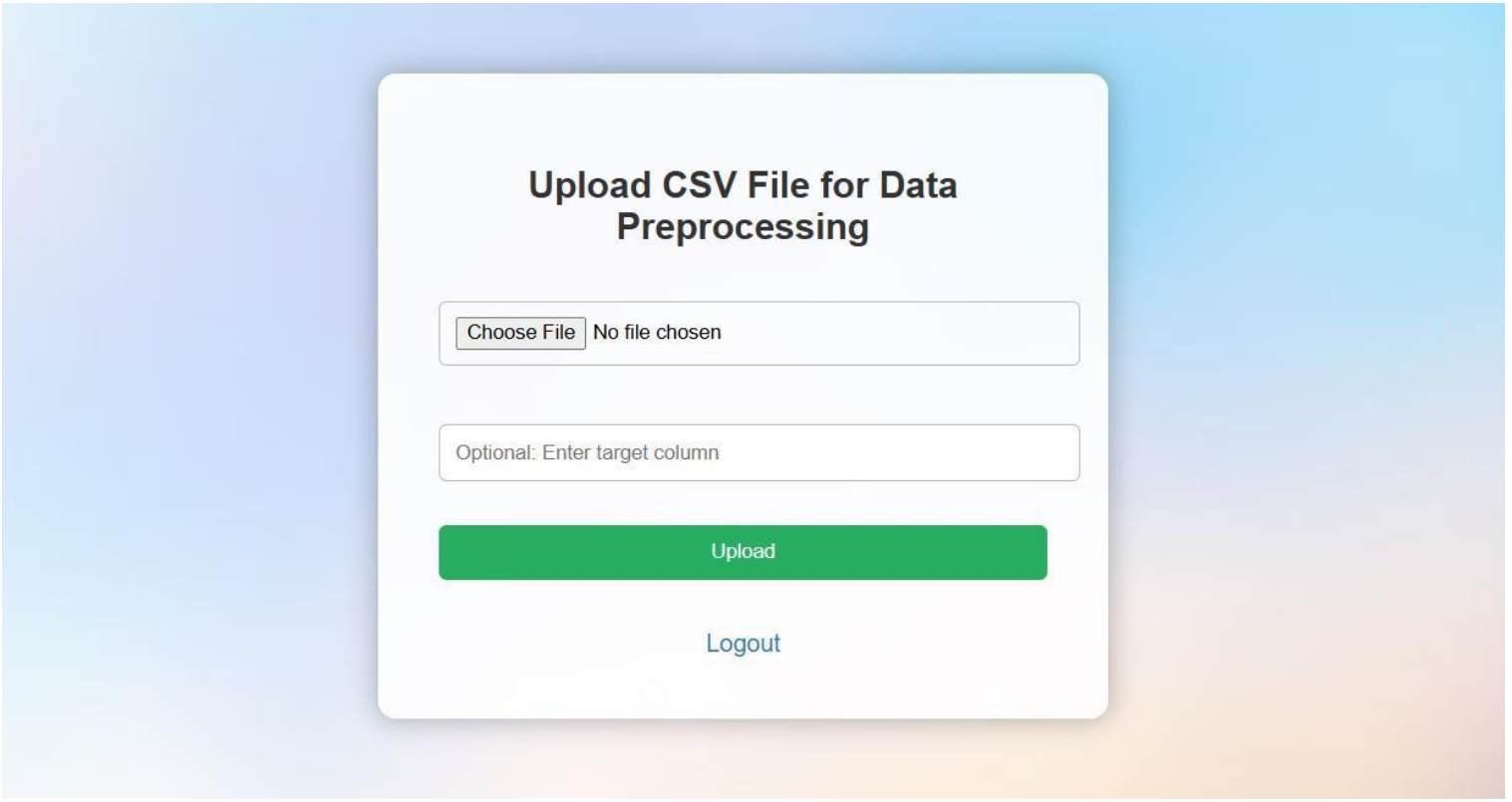

Fig. B.2 UPLOADING PAGE

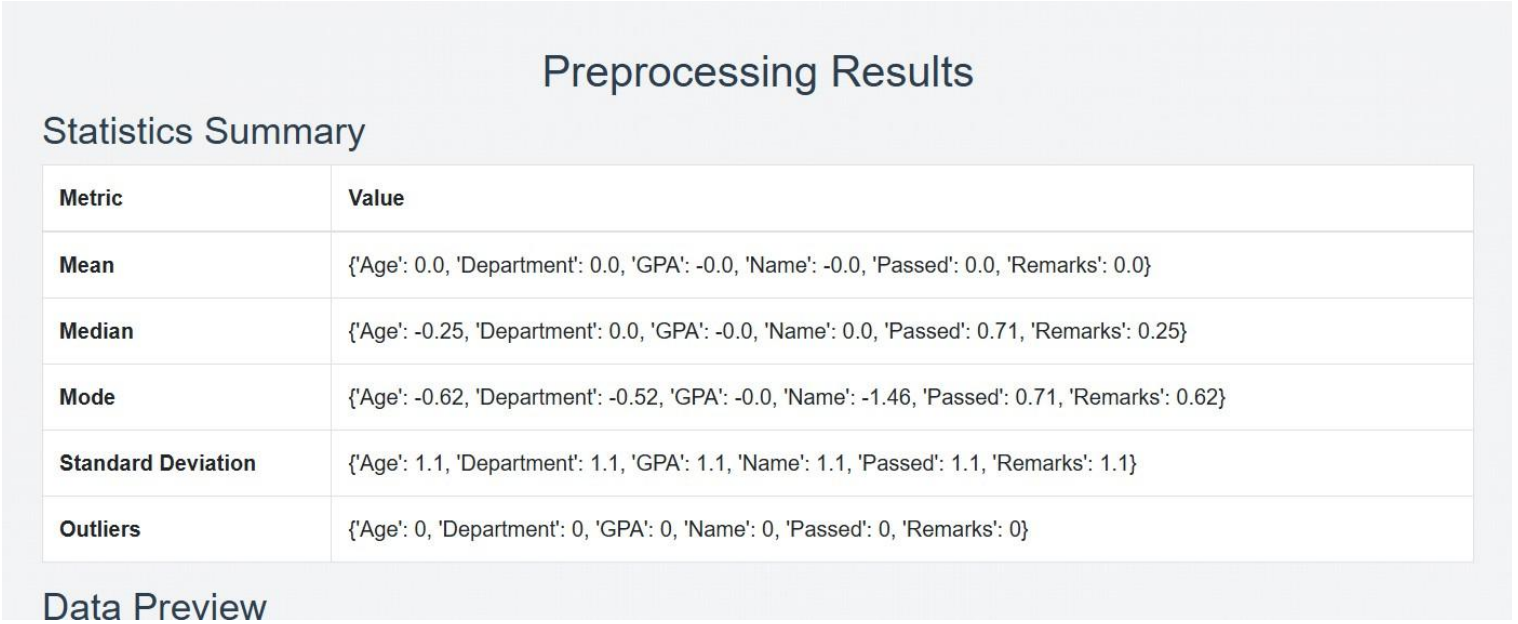

Fig. B.3 RESULTS


Generated Plots
Histogram
Name
Age
Department
GPA
Passed
Remarks
Scatter Plot

Fig. B.4 Plots

REFERENCES

.

1. R. Huang, J. Liu, T. K. Wan, D. Siriwanna, Y. M. P. Woo, A. Vodencarevic, C. W. Wong, and K. H. K. Chan, ‘‘Stroke mortality prediction based on ensemble learning and the combination of structured and textual data,’’ Comput. Biol. Med., vol. 155, Mar. 2023
2. G. Kumawat, S. K. Vishwakarma, P. Chakrabarti, P. Chittora, T. Chakrabarti, and J. C.-W. Lin, ‘‘Prognosis of cervical cancer disease by applying machine learning techniques,’’ J. Circuits, Syst. Comput., vol. 32, no. 1, Jan. 2023
3. R. R. Kadhim and M. Y. Kamil, ‘‘Comparison of machine learning models for breast cancer diagnosis,’’ IAES Int. J. Artif. Intell. (IJ-AI), vol. 12, no. 1, p. 415, Mar. 2023.
4. B. S. Ahamed, M. S. Arya, and A. O. V. Nancy, ‘‘Diabetes mellitus disease prediction using machine learning classifiers with oversampling and feature augmentation,’’ Adv. Hum.-Comput. Interact., vol. 2022, pp. 1–14, Sep. 2022.
5. .R. Ghorbani and R. Ghousi, ‘‘Predictive data mining approaches in medical diagnosis: A review of some diseases prediction,’’ Int. J. Data Netw. Sci., vol. 3, no. 2, pp. 47–70, 2019.
6. R. Ghorbani and R. Ghousi, ‘‘Predictive data mining approaches in medical diagnosis: A review of some diseases prediction,’’ Int. J. Data Netw. Sci., vol. 3, no. 2, pp. 47–70, 2019.

2nd INTERNATIONAL CONFERENCE ON DATA ANALYTICS AND INTELLIGENCE COMPUTING-2025

(ICDAIC'25)

VELAMMAL INSTITUTE OF TECHNOLOGY

CERTIFICATE OF PARTICIPATION

This is to certify that Prof./Dr./Mr./Ms. **AUDHAVAN A** of **K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY** has presented a paper titled **CLEANSE AI:DATA PREPROCESSING AUTOMATION TOOL USING PYTHON AND MACHINE LEARNING ALGORTITHMS** in 2nd International Conference on Data Analytics and Intelligence Computing organized by the Department of Artificial Intelligence and Data Science, Velammal Institute of Technology, Chennai, TamilNadu, India on April 09, 2025.

COORDINATORS
Ms.K.Sudha
Mr.K.Dinesh Kumar

HOD
Dr.S.PadmaPriya

POORVIKA

VICE PRINCIPAL
Dr.S.Soundararajan

PRINCIPAL
Dr.N.Balaji

2nd INTERNATIONAL CONFERENCE ON DATA ANALYTICS AND INTELLIGENCE COMPUTING-2025

(ICDAIC'25)

VELAMMAL INSTITUTE OF TECHNOLOGY

CERTIFICATE OF PARTICIPATION

This is to certify that Prof./Dr./Mr./Ms. **BARATHVAJ M** of **K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY** has presented a paper titled **CLEANSE AI:DATA PREPROCESSING AUTOMATION TOOL USING PYTHON AND MACHINE LEARNING ALGORTITHMS** in 2nd International Conference on Data Analytics and Intelligence Computing organized by the Department of Artificial Intelligence and Data Science, Velammal Institute of Technology, Chennai, TamilNadu, India on April 09, 2025.

COORDINATORS
Ms.K.Sudha
Mr.K.Dinesh Kumar

HOD
Dr.S.PadmaPriya

POORVIKA

VICE PRINCIPAL
Dr.S.Soundararajan

PRINCIPAL
Dr.N.Balaji

2nd INTERNATIONAL CONFERENCE ON DATA ANALYTICS AND INTELLIGENCE COMPUTING-2025

(ICDAIC'25)

VELAMMAL INSTITUTE OF TECHNOLOGY

CERTIFICATE OF PARTICIPATION

This is to certify that Prof./Dr./Mr./Ms. **MOHAMED THABISH M** of **K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY** has presented a paper titled **CLEANSE AI:DATA PREPROCESSING AUTOMATION TOOL USING PYTHON AND MACHINE LEARNING ALGORTITHMS** in 2nd International Conference on Data Analytics and Intelligence Computing organized by the Department of Artificial Intelligence and Data Science, Velammal Institute of Technology, Chennai, TamilNadu, India on April 09, 2025.

COORDINATORS
Ms.K.Sudha
Mr.K.Dinesh Kumar

HOD
Dr.S.PadmaPriya

POORVIKA

VICE PRINCIPAL
Dr.S.Soundararajan

PRINCIPAL
Dr.N.Balaji

2nd INTERNATIONAL CONFERENCE ON DATA ANALYTICS AND INTELLIGENCE COMPUTING-2025

(ICDAIC'25)

VELAMMAL INSTITUTE OF TECHNOLOGY

CERTIFICATE OF PARTICIPATION

This is to certify that Prof./Dr./Mr./Ms. **VIGNESHWARAN M** of **K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY** has presented a paper titled **CLEANSE AI:DATA PREPROCESSING AUTOMATION TOOL USING PYTHON AND MACHINE LEARNING ALGORTITHMS** in 2nd International Conference on Data Analytics and Intelligence Computing organized by the Department of Artificial Intelligence and Data Science, Velammal Institute of Technology, Chennai, TamilNadu, India on April 09, 2025.

COORDINATORS
Ms.K.Sudha
Mr.K.Dinesh Kumar

HOD
Dr.S.PadmaPriya

POORVIKA

VICE PRINCIPAL
Dr.S.Soundararajan

PRINCIPAL
Dr.N.Balaji