A Mini Project

Report On

# STRATEGIC RESOURCE INSIGHTS: UNDERSTANDING AMERICA'S ECONOMIC RELIANCE ON NATURAL RESOURCES

Submitted to Jawaharlal Nehru Technological University for the partial Fulfillment of the Requirement for the Award of the Degree of

**Bachelor of Technology**
**In**
**Computer Science and Engineering**


By

**MARRI VIGNESHKUMAR - (22RA1A05H6)**
**MANCHALA SHASHI PRADEEP- (22RA1A05E6)**
**DOLKA SNEHITH - (22RA1A05I9)**

Under the guidance of
**Mr.B.Ritesh kumar**
**(M.E)**
Assistant Professor, CSE




**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY**

**(UGC AUTONOMOUS)**

**(Affiliated to JNTUH, Ghanpur (V), Ghatkesar (M), Medchal (D)-500088)**


2022-2026

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY**

**(UGC AUTONOMOUS)**

**(Ghanpur (V), Ghatkesar (M), Medchal (D)-500088)**

**(AFFILIATED TO JNTU, Hyderabad)**


# CERTIFICATE

This is to certify that the Mini Project entitled **"STRATEGIC RESOURCE INSIGHTS: UNDERSTANDING AMERICA'S ECONOMIC RELIANCE ON NATURAL RESOURCES"** being submitted by **MARRI VIGNESH KUMAR (22RA1A05H6), MANCHALA SHASHI PRADEEP (22RA1A05E6), DOLKA SNEHITH (22RA1A05I9)** in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering to the Jawaharlal Nehru Technology University is a record of confined work carried out by them under my guidance and supervision.

Guide's signature
**MR.Ritesh kumar**
(M.E) .
Asst. Professor
Dept. of CSE

Signature of the Head of the Dept.
**K.VAMSHEE KRISHNA**
M.Tech, (Ph.D).
Asst. Professor
Dept. of CSE

Place: Ghanpur

Date:

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY**

**(UGC AUTONOMOUS)**

**(Ghanpur (V), Ghatkesar (M), Medchal (D)-500088)**

**(AFFILIATED TO JNTU, Hyderabad)**


# DECLARATION

We, **MARRI VIGNESH KUMAR (22RA1A05H6), MANCHALA SHASHI PRADEEP (22RA1A05E6), DOLKA SNEHITH (22RA1A05I9)** hereby declare that the mini project report titled **"STRATEGIC RESOURCE INSIGHTS: UNDERSTANDING AMERICA'S ECONOMIC RELIANCE ON NATURAL RESOURCES"** under the guidance of Mr. RITESH KUMAR , Assistant Professor, Department of Computer Science and Engineering, Kommuri Pratap Reddy Institute of Technology, Ghanpur, is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

This is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this thesis have not been submitted to any other University and Institute for the award of any Degree or Diploma.

**MARRI VIGNESH KUMAR (22RA1A05H6)**
**MANCHALA SHASHI PRADEEP (22RA1A05E6)**
**DOLKA SNEHITH (22RA1A05I9)**

# ACKNOWLEDGEMENT

We manifest our heartier thankfulness pertaining to our contentment over **Mr. RITESH KUMAR**, Assistant Professor as a project guide with whose adroit concomitance the excellence has been exemplified in bringing out this project work with artistry. We express our gratitude to **Mr. K. Vamshee Krishna**, Head of the Department for the encouragement and assistance to us, which contribute to the successful completion of this project. This Acknowledgement will be incomplete without mentioning our sincere gratefulness to our honorable Chairman **Sri. Kommuri Pratap Reddy**, our Director, **Dr. B. Sudheer Prem Kumar,** and Vice Principal, Dr.Srinath Kashyap, who have been observing posing valiance in abundance, forwarding our individuality to acknowledge our project work tendentiously. At the outset we thank teaching and non-teaching staff of Dept. of CSE for providing us with good faculty and for their moral support throughout the course.

A heartful and sincere gratitude to our beloved parents for their tremendous motivation and moral support. We also express the overall exhilaration and gratitude to all those who animated our project work and accentuated our stance.

**MARRI VIGNESH KUMAR (22RA1A05H6)**
**MANCHALA SHASHI PRADEEP (22RA1A05E6)**
**DOLKA SNEHITH (22RA1A05I9)**

## Vision of the Institute

To emerge as a premier institute for high quality professional graduates who can contribute to economic and social developments of the Nation.

## *Mission of the Institute*

| Mission | Statement |
|---|---|
| **IM1** | To have holistic approach in curriculum and pedagogy through industry interface to meet the needs of Global Competency. |
| **IM2** | To develop students with knowledge, attitude, employability skills, entrepreneurship, research potential and professionally Ethical citizens. |
| **IM3** | To contribute to advancement of Engineering & Technology that would help to satisfy the societal needs. |
| **IM4** | To preserve, promote cultural heritage, humanistic values and Spiritual values thus helping in peace and harmony in the society. |

KPRIT
Imagine. Innovate. Inspire.

KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

## *Vision of the Department*

To Provide Quality Education in Computer Science for the innovative professionals to work for the development of the nation.

## *Mission of the Department*

| Mission | Statement |
|---|---|
| **DM1** | Laying the path for rich skills in Computer Science through the basic knowledge of mathematics and fundamentals of engineering |
| **DM2** | Provide latest tools and technology to the students as a part of learning infrastructure |
| **DM3** | Training the students towards employability and entrepreneurship to meet the societal needs. |
| **DM4** | Grooming the students with professional and social ethics. |

## Program Educational Objectives (PEOs)

| PEO's | Statement |
|---|---|
| **PEO1** | The graduates of Computer Science and Engineering will have successful career in technology. |
| **PEO2** | The graduates of the program will have solid technical and professional foundation to continue higher studies. |
| **PEO3** | The graduate of the program will have skills to develop products, offer services and innovation. |
| **PEO4** | The graduates of the program will have fundamental awareness of industry process, tools and technologies. |

## Program Outcomes

| | |
|---|---|
| **PO1** | **Engineering Knowledge:** Apply the knowledge of mathematics, science, Engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| **PO2** | **Problem Analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. |
| **PO3** | **Design/development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. |
| **PO4** | **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. |
| **PO5** | **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations. |
| **PO6** | **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. |
| **PO7** | **Environment and sustainability:** Understand the Impact of the professional engineering solutions in societal and environmental context, and demonstrate the knowledge of, and need for sustainable development. |
| **PO8** | **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |

| | |
|---|---|
| **PO9** | Individual and team network: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. |
| **PO10** | Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, being able to comprehend and write effective reports and design documentation, make Effective presentations, and give and receive clear instructions. |
| **PO11** | Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environment. |
| **PO12** | Life-Long learning: Recognize the need for, and have the preparation and able to engage in independent and life-long learning in the broadest context of technological change. |

## PROGRAM SPECIFIC OUTCOMES

| | |
|---|---|
| **PSO1** | Foundation of mathematical concepts: To use mathematical methodologies to crack problem using suitable mathematical analysis, data structure and suitable algorithm. |
| **PSO2** | Foundation of Computer Science: The ability to interpret the fundamental concepts and methodology of computer systems. Students can understand the functionality of hardware and software aspects of computer systems. |
| **PSO3** | Foundation of Software development: The ability to grasp the software development lifecycle and methodologies of software systems. Possess competent skills and knowledge of software design process. |

# ABSTRACT

The rapid evolution of Internet of Things (IoT) technology has catalyzed advancements in resource economics, particularly in understanding America's dependency on minerals to foster sustainable solutions. This IoT-enabled resource economics system aims to streamline the management of mineral resources, analyzing their usage, extraction, and efficiency in various industries. Historically, the resource sector relied on manual tracking and data collection, with a focus on traditional mining and resource management systems, which often lacked real-time insights and predictive capabilities. Before the integration of Artificial Intelligence (AI) and IoT, challenges such as inefficient resource allocation, long reaction times to changes in mineral demand, and limited visibility into resource management were prevalent. These methods used systems like spreadsheets, manual audits, and basic automation tools, but they were often disconnected, leading to inefficiencies and data silos. The motivation behind this research stems from the growing need for real-time, data-driven decisions in a world that demands more sustainable and efficient resource usage. The development of this system was inspired by the need to monitor, track, and optimize mineral resource management, ensuring that solutions are both environmentally and economically sustainable. One major problem in the traditional system was the lack of integrated systems to monitor real-time data from the mining process, leading to delays and inaccuracies in reporting. Moreover, reliance on outdated methods often led to over-extraction, depletion of mineral reserves, and unforeseen environmental impacts. This proposed IoT-enabled system, augmented by machine learning algorithms, will improve resource management by enabling real-time monitoring, predictive analytics, and automation of decision-making processes. The integration of AI will facilitate the optimization of mineral usage and supply chains, improving sustainability while minimizing resource wastage, ultimately providing a smarter, data-centric approach to tackling America's reliance on critical minerals.

# Table of Contents

# DIAGRAMS

# TABLES

# 1.INTRODUTION

# CHAPTER 1

# INTRODUCTION

## 1.1 Introduction

In recent years, the intersection of IoT (Internet of Things) and resource economics has emerged as a critical domain for enhancing sustainable solutions, particularly in the context of America's dependence on mineral resources. The increasing demand for minerals due to technological advancements, renewable energy production, and electric vehicle development has underscored the importance of ensuring efficient and sustainable extraction and usage of these resources. IoT-enabled resource economics allows for real-time monitoring, analysis, and optimization of mineral use across different sectors, providing a comprehensive and data-driven approach to resource management. By utilizing connected devices and sensors, data on mineral reserves, extraction rates, and market demand can be gathered and analyzed to offer actionable insights. This framework aims to bridge the gap between resource availability, environmental concerns, and economic growth. Additionally, it facilitates more accurate predictions, reducing waste and inefficiency, while promoting the responsible use of natural resources. Given that mineral resources are finite, the need to optimize their usage for both economic and environmental sustainability has never been more pressing. IoT technology, combined with machine learning, is paving the way for a smarter, more efficient resource management ecosystem that directly addresses these challenges. This research aims to showcase how IoT and AI-based solutions can enhance the sustainability of mineral resource management, ensuring their availability for future generations.

## 1.2 Research Motivation

The motivation behind this research stems from the growing concern over the sustainability of America's mineral resource usage, particularly in light of rapid technological advancements that require an increasing supply of these critical materials. Traditional resource management systems often lack the capability to offer real-time insights or predictive analytics, resulting in inefficiencies and unsustainable extraction practices. As global demand for minerals rises, coupled with the urgency for cleaner, greener energy solutions, it is imperative to develop systems that not only track mineral resources but also predict future demand and identify potential issues in the supply chain before they arise. This research is driven by the necessity

of enhancing the way mineral resources are managed, ensuring that they are used efficiently, waste is minimized, and the environmental impact is reduced. By integrating IoT and machine learning, we can introduce a more agile and intelligent resource management system that responds to market fluctuations, optimizes resource extraction, and addresses environmental concerns. Moreover, this research aims to contribute to the global discourse on sustainable mining practices, offering a practical and data-driven solution for the future of mineral resource economics.

## 1.3 Problem Statement

Before the integration of AI and machine learning, the management of mineral resources in the United States faced numerous challenges. Traditional systems relied heavily on manual data collection, which was not only time-consuming but also prone to human error. Resource availability and extraction processes were often tracked using static spreadsheets and reports, making it difficult to adapt to dynamic market demands or changing environmental conditions. Additionally, these systems were disconnected, creating data silos that hindered comprehensive decision-making. The absence of real-time monitoring meant that issues such as over-extraction, resource depletion, and environmental degradation went undetected until it was too late. Furthermore, predictive analytics were limited, making it challenging to forecast future mineral requirements or identify inefficiencies within the supply chain. These shortcomings led to missed opportunities for optimizing resources, increased operational costs, and unsustainable practices that threatened long-term resource availability. The lack of integrated systems to manage real-time data and predict future trends was a significant barrier to effective resource management in the mining and mineral industries.

## 1.4 Applications

1. Helps mineral resource management companies optimize extraction processes.
2. Can be applied in the renewable energy sector to manage critical mineral resources.
3. Provides real-time data analysis for environmental monitoring and conservation.
4. Supports decision-making in the mining industry to enhance sustainability.
5. Assists policymakers in crafting regulations related to resource sustainability.

6. Useful for industries that rely heavily on minerals for production, such as electronics.
7. Can be implemented in urban planning for better use of natural resources.
8. Aids in forecasting and managing mineral

# 2.LITERATURE SURVEY

# CHAPTER 2

# LITERATURE SURVEY

Kim et al. [1] proposed a framework for smart city development with a focus on Seoul, highlighting its technological, infrastructural, and administrative challenges. They examined how technology adoption could transform urban governance and improve city life, emphasizing the role of government in facilitating the transition to a smart city model.

Petrolo et al. [2] presented a concept for a smart city powered by the cloud of things. They explored the integration of cloud computing with the Internet of Things (IoT) to enable a scalable and flexible infrastructure that supports urban services such as transportation, healthcare, and energy management.

Alawadhi et al. [3] conducted a study to understand various smart city initiatives globally. They provided insights into the challenges faced by cities in adopting smart technologies and identified key factors influencing the successful implementation of smart city projects, including governance, collaboration, and technology adoption.

Nam and Pardo [4] conceptualized the smart city model by categorizing it into three dimensions: technology, people, and institutions. They proposed a framework to assess the interplay between these dimensions, which is essential for creating sustainable and resilient urban environments.

Kaye Nijaki and Worrel [5] explored sustainable procurement practices for local economic development. They highlighted the role of public sector procurement in fostering sustainable local development and discussed strategies for aligning procurement processes with smart city objectives.

Alahi et al. [6] presented a temperature-compensated smart nitrate sensor designed for use in the agricultural industry. They focused on enhancing the sensor's accuracy and reliability under varying environmental conditions, contributing to better resource management in agriculture.

Alahi et al. [7] developed an IoT-enabled smart sensing system for nitrate monitoring in agricultural settings. The system was designed to provide real-time data on soil nitrate levels, helping farmers make informed decisions to optimize fertilizer use and reduce environmental impact.

Alahi et al. [8] proposed an imprinted polymer-coated impedimetric nitrate sensor for real-time water quality monitoring. Their approach aimed to improve the sensitivity and specificity of nitrate detection, offering a reliable tool for environmental monitoring in smart cities.

Theoleyre et al. [9] provided an editorial on networking and communication technologies for smart cities. They discussed the importance of low-power, long-range communication technologies in enabling the widespread deployment of IoT devices in urban environments.

Djahel et al. [10] proposed a solution based on Vehicle-to-Infrastructure (V2I) communication technologies to address road traffic congestion in smart cities. Their study emphasized the role of V2I systems in improving traffic flow and reducing delays in urban transportation networks.

Wenge et al. [11] outlined the architecture and design challenges of smart cities. They explored various technologies and strategies for implementing smart city solutions, with a focus on the integration of IoT, cloud computing, and data analytics.

Foubert and Mitton [12] conducted a survey of long-range wireless radio technologies, which are crucial for enabling IoT communication in smart cities. Their research aimed to provide insights into the performance and potential applications of these technologies in urban environments.

Xia et al. [13] discussed the role of the Internet of Things in modern communication systems. They highlighted the integration of IoT into urban infrastructures and its impact on enhancing city services, including transportation, energy, and healthcare management.

Kopetz [14] presented a detailed discussion on the Internet of Things within the context of real-time systems. His work emphasized the challenges and opportunities in developing real-time IoT applications, particularly in time-sensitive urban operations.

Albino et al. [15] reviewed the concept of smart cities, focusing on their definitions, dimensions, and performance metrics. They examined key initiatives and strategies for implementing smart city projects, stressing the importance of a holistic approach to urban development.

Syed et al. [16] conducted a survey of IoT technologies used in smart cities. They explored the challenges and best practices in IoT deployment, highlighting the critical role of smart sensors, data analytics, and communication networks in building efficient and sustainable cities.

Jeong and Park [17] examined the challenges and opportunities in integrating IoT and smart city technologies. Their study explored how IoT can address urban challenges such as traffic congestion, energy efficiency, and public safety, offering potential solutions for urban planners.

Kim et al. [18] discussed the relationship between smart city development and the Internet of Things. They explored the synergies between IoT and smart city concepts, illustrating how IoT technologies can enable better urban management and improved quality of life.

Bellini et al. [19] reviewed the key concepts, frameworks, and technologies enabling IoT in smart cities. They discussed the role of IoT in optimizing urban infrastructure, including energy management, transportation, and waste management, contributing to smarter, more sustainable cities.

Statista [20] provided data on the global number of IoT-connected devices from 2019 to 2021, with forecasts extending to 2030. This data is crucial for understanding the rapid expansion of IoT technologies and their impact on smart city development.

Janani et al. [21] presented a contemporary survey on the use of IoT in smart cities. They reviewed various IoT applications and highlighted the emerging trends and challenges in deploying IoT technologies to enhance urban services and improve quality of life.

Al-Turjman [22] offered an overview of information-centric sensor networks for cognitive IoT applications. His work focused on the potential of these networks to enable intelligent decision-making and optimize resource management in smart cities.

Al-Turjman [23] proposed a framework for traffic modeling and optimization in IoT-enabled smart cities. His study aimed to enhance urban mobility by integrating IoT technologies with traffic management systems, improving efficiency and reducing congestion.

Liu et al. [24] explored intelligent edge computing for IoT-based energy management in smart cities. Their research highlighted how edge computing can optimize energy use in urban environments, providing a decentralized approach to energy management.

Allam and Dhunny [25] discussed the role of big data and artificial intelligence in the development of smart cities. They analyzed the intersection of these technologies and their potential to transform urban systems, improving sustainability, efficiency, and citizen engagement.

Kapoor et al. [26] applied data science approaches to identify infrastructural gaps for sustainable tribal village development in India. Their work demonstrated how data-driven insights can inform the development of smart and sustainable villages, with implications for broader smart city initiatives.

Kar et al. [27] reviewed advancements in smart cities, focusing on governance, people, and technological innovations. Their research provided insights into how smart city solutions can improve urban life, governance, and citizen engagement in both developed and developing regions.

# 3.EXISTINGSYSTEM

# CHAPTER 3

# EXISTING SYSTEM

The traditional system of mineral resource management in America primarily relied on manual and semi-automated methods for tracking extraction, usage, and reserves. Data was often collected through physical inspections, written reports, spreadsheets, and periodic audits. These records were maintained separately by individual departments, leading to fragmented and siloed information. Mining operations, logistics, and supply chain activities were coordinated without centralized oversight, making it difficult to obtain real-time insights. Decision-making was reactive, based on historical data rather than current trends or predictive analysis. This lack of integrated digital infrastructure meant that companies and governmental bodies struggled to respond swiftly to shifts in mineral demand or environmental regulations. The system was also heavily dependent on human input, increasing the risk of errors and inconsistencies. Forecasting mineral needs or anticipating supply chain disruptions was a challenge due to the absence of advanced analytics. As a result, resource allocation was often suboptimal, and sustainability goals were hard to meet. Environmental monitoring, when done, was largely retrospective, limiting the ability to prevent damage in real-time. Ultimately, the traditional resource economics approach lacked the agility and intelligence needed for efficient, sustainable management in today's data-driven era.

**Limitations of the Traditional System:**

- Lacked real-time data monitoring and analytics capabilities
- Depended on manual data entry and audits, increasing error risks
- Fragmented systems created data silos and poor communication
- Slow response times to market or environmental changes
- Inability to forecast resource demand and availability accurately
- Limited integration with environmental impact monitoring
- Inefficient resource allocation and over-extraction risks
- No automation or predictive decision-making tools

# 4.PROPOSED SYSTEM

# CHAPTER 4

# PROPOSED SYSTEM

## 4.1 Overview

**Research Methodology and Procedure**

**Step 1: Uploading the American Minerals Dataset** The research begins with the acquisition and upload of the primary dataset, which serves as the foundation for all subsequent analysis. This dataset, referred to as the "American Minerals Dataset," captures a range of features relevant to mineral resource extraction, usage patterns, environmental considerations, and their impact on sustainable solutions. This data is collected in CSV format, ensuring compatibility with Python-based data science libraries such as pandas and NumPy. Once uploaded, the dataset is loaded into memory and displayed for verification. The structure, dimensions, and attribute names are then explored to understand what kind of data is available and how it aligns with the research goals. The significance of this step lies in ensuring that the dataset is complete and accurate before any transformation or modeling takes place. Any anomalies detected at this stage—such as unexpected column names, irregular data types, or truncated values—can hinder the quality of later steps. Therefore, a thorough inspection is conducted immediately after the upload to affirm that the data is usable and consistent.

**Step 2: Data Preprocessing (Null Values, Info, Unique Values, Correlation Plot)** Data preprocessing is critical to ensure that the dataset is clean and ready for analysis. It begins with the inspection of dataset characteristics using .info() and .describe(), which summarize the count, data type, non-null values, and basic statistical information like mean, standard deviation, and range. This step identifies possible data quality issues, including the presence of missing values, inconsistent data entries, or outliers. The analysis of null values across columns is done to assess data completeness. Columns with significant proportions of missing values may need to be dropped or imputed, depending on their importance to the prediction goal. The isnull().sum() method provides a clear snapshot of how missingness is distributed across features. A unique value inspection helps determine the variability of categorical and numerical attributes. Features with low variability (e.g., a column where 99% of values are identical) may offer limited predictive power and could be excluded or transformed. Additionally, a correlation heatmap is plotted using seaborn to understand the linear relationships among

numerical features. This visualization is essential for identifying multicollinearity and choosing the most relevant features for model input. A strong correlation between features may necessitate dimensionality reduction to prevent overfitting. This preprocessing step ensures data quality, identifies irrelevant or redundant attributes, and provides insight into the underlying data patterns that guide the modeling strategy.

**Step 3: Standard Scaling** Once the data is cleaned and pre-processed, standardization is performed using StandardScaler. Standard scaling transforms the feature values so that each has a mean of 0 and a standard deviation of 1. This is especially important in machine learning algorithms that are sensitive to feature magnitudes, such as Support Vector Machines (SVM) and Logistic Regression. The scaling is applied separately to the training and testing datasets to prevent data leakage. The scaler is first fitted to the training data, capturing the mean and standard deviation of each feature. These parameters are then applied to transform both the training and testing datasets. This guarantees that the model learns from and generalizes across standardized values without being biased by the test data distribution. Standard scaling not only enhances model performance but also accelerates convergence during training. It ensures that each feature contributes equally to distance calculations, which is fundamental for algorithms relying on Euclidean geometry.

**Step 4: Train-Test Splitting (80-20 Ratio)** With a preprocessed and scaled dataset in hand, the next phase involves splitting the data into training and testing subsets. The train_test_split() function from scikit-learn is utilized for this purpose, dividing the dataset in an 80:20 ratio. This means 80% of the data is used for training the model, while 20% is reserved for evaluation. This step is pivotal in creating a generalizable machine learning model. Training data is used to fit the model parameters, while testing data provides an unbiased evaluation of its performance. A good split ensures that the test set represents the same distribution as the training data, preventing skewed or misleading evaluation metrics. Stratification is optionally used if the target variable is imbalanced, ensuring that both subsets contain proportional distributions of classes. This is especially relevant in binary or multi-class classification tasks where the dependent variable might be unequally distributed. Overall, splitting the dataset ensures robustness in validation and provides a clear framework to compare model predictions against ground-truth labels.

**Step 5: Existing Logistic Regressor Model Building** To establish a baseline, a Logistic Regression model is developed and evaluated first. Logistic Regression is a widely used

supervised learning algorithm for binary classification tasks. It models the probability of the default class by fitting a logistic function (sigmoid) to a linear combination of input features. In this system, the logistic model is either loaded from a previously saved file or trained anew. If the model is not found, it is trained on the scaled training dataset and then serialized using joblib for future use, ensuring reproducibility and efficiency. Once trained, the model predicts outcomes on the test data. These predictions are compared with actual labels using performance metrics such as accuracy, precision, recall, and F1-score. A confusion matrix is also generated and visualized to provide insights into true positives, false positives, true negatives, and false negatives. These metrics help in evaluating the model's effectiveness in identifying sustainable versus non-sustainable outcomes. This step sets a performance benchmark against which advanced models can be compared. Logistic Regression serves as a simple yet powerful classifier for this predictive analysis.

**Step 6: Proposed SVC Model Building** As a proposed enhancement over the logistic regression baseline, a Support Vector Classifier (SVC) model is employed. In this project, although classification is the primary task, SVC is used with classification intent via the SVC (Support Vector Classifier) interface from scikit-learn. This implementation fits a hyperplane in a high-dimensional space to separate the data classes with maximum margin. The SVC model provides robust generalization and is well-suited for datasets with non-linear boundaries. Like the logistic model, it is either loaded from storage or trained on the current training set and saved for future use. This model benefits from the prior scaling step since SVC is highly sensitive to feature magnitudes. Once predictions are made on the test dataset, the same evaluation metrics are calculated and plotted for direct comparison. The rationale behind introducing SVC lies in its potential to better capture complex relationships between features and the sustainable solution outcome, thus improving accuracy and reducing misclassification errors.

**Step 7: Performance Comparison** After both models (Logistic Regression and SVC) are evaluated, their performances are compared using key classification metrics. Accuracy measures the overall correctness, precision indicates the ability to avoid false positives, recall reflects sensitivity to actual positive classes, and F1-score balances both precision and recall. This comparative analysis is visualized through confusion matrices, classification reports, and metric scores, providing a comprehensive overview of each model's strengths and weaknesses. This step also includes plotting performance scores across models for visual clarity. A

comparative framework is crucial in determining which model generalizes better and offers more reliability in real-world deployment. It also informs future decisions regarding model tuning or the integration of additional features. This step closes the classification model development loop by validating and benchmarking the predictive power of each model using a consistent dataset.

**Step 8: Prediction from Test Data Using SVC Model** The final step introduces an approach through the Support Vector Classifier (SVC), aimed at further enhancing predictive reliability. SVC is known for its high accuracy and ability to handle both classification and regression tasks. It works by constructing multiple decision trees during training and outputting the mode of their predictions for classification. Similar to previous models, SVC is either loaded from an existing file or trained afresh. Once trained, it is evaluated on the test set, and performance metrics are again computed. However, beyond evaluation, the SVC model is also used for generating final predictions on an entirely new test dataset provided separately in testdata.csv. This represents a real-world simulation where the trained model is applied to unseen data, demonstrating its deployment readiness. The predicted values are appended to the original test dataset for further analysis or presentation. This step confirms the practicality and scalability of the proposed framework. The inclusion of an based approach improves generalization and reduces the likelihood of overfitting, making it ideal for production scenarios.

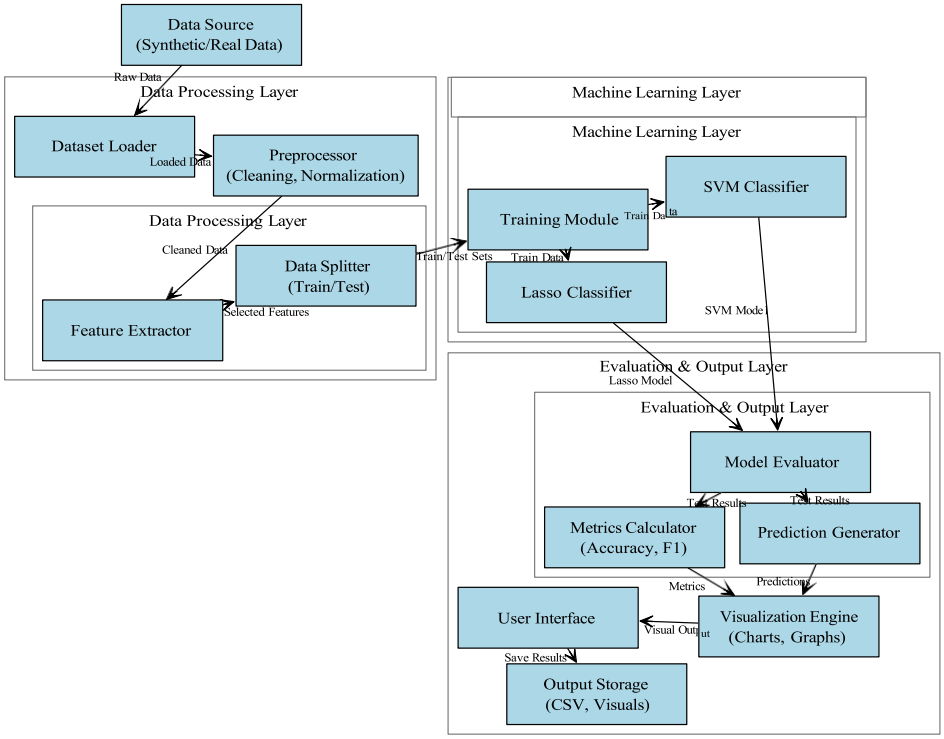


Fig. 1: Architectural Block Diagram.

## 4.2 Data Preprocessing

Step 1: The dataset is initially inspected using data exploration techniques that provide structural insights such as the number of records, column names, data types, and summary statistics like mean, standard deviation, and range. This step ensures familiarity with the dataset's overall architecture.

Step 2: Missing value analysis is conducted by calculating the total and percentage of null entries for each column. Based on the significance of each feature and its missing data ratio, decisions are made to either drop or impute missing values using appropriate statistical methods like mean or median.

Step 3: Categorical and numerical features are analyzed for uniqueness using unique value counts. Features with a single unique value or extremely low variance are identified and removed to eliminate redundant or non-informative data that adds no value to the model.

Step 4: A correlation matrix is generated using Pearson correlation coefficients to analyze inter-feature relationships. The matrix is visualized through a heatmap to identify highly correlated features that could lead to multicollinearity. Redundant features are removed to ensure model efficiency and accuracy.

Step 5: Features showing no meaningful correlation with the target variable are discarded from the dataset. This improves model generalization by focusing only on features that contribute to predictive performance.

Step 6: Numerical features are standardized using z-score normalization. Each value is transformed to have a mean of zero and standard deviation of one. This standardization ensures that all features contribute equally to model training, especially for algorithms sensitive to feature scales.

Step 7: The final cleaned and scaled dataset is split into training and testing subsets using an 80:20 ratio. This separation enables model validation on unseen data and supports accurate evaluation of performance metrics.

## 4.3 ML MODELS

### 4.3.1 Existing Model: Logistic Regressor

**Introduction about Existing Model**

Logistic Regression is a supervised learning algorithm used for classification tasks. It is based on the principle of estimating the probability that a given input belongs to a particular class. Despite its name, logistic regression is actually a classification technique rather than a regression method. It is commonly used for binary classification problems and extended to multi-class problems using strategies like one-vs-rest. The model applies the logistic function or sigmoid function to map predicted values to probabilities between 0 and 1. Logistic Regression is widely preferred due to its simplicity, interpretability, and relatively low computational requirements.

**Working of Existing Model**

Logistic Regression works by modeling the relationship between a dependent categorical variable and one or more independent variables using a logistic function. The logistic function, also known as the sigmoid function, transforms the linear combination of input features into a probability score between 0 and 1. The model assigns weights to the input features based on their contribution to the target output. During training, these weights are optimized using a cost function such as binary cross-entropy, which measures the difference between actual and predicted values. The optimization process involves minimizing this cost function using gradient descent or similar optimization techniques. Once the weights are determined, the logistic function generates probabilities for new input data. A threshold value, usually 0.5, is used to convert these probabilities into class labels. For multi-class classification problems, the model extends using a softmax function or a one-vs-rest approach. Logistic Regression assumes a linear relationship between the input features and the log odds of the outcome. It also assumes that input variables are independent and that there is minimal multicollinearity. The model outputs are interpretable, making it easier to understand feature importance. Its computational efficiency and capability to handle high-dimensional data make it suitable for baseline classification tasks in various domains, including medical diagnostics, financial fraud detection, and marketing response prediction.

**Limitations of the Existing Model**

- Assumes linear relationship between features and log-odds of the target
- Performs poorly when the relationship is non-linear
- Sensitive to outliers which can skew predictions
- Requires input features to be scaled and independent
- Struggles with multicollinearity in the dataset
- Inadequate for datasets with overlapping classes
- Performs sub-optimally with high-dimensional sparse data
- Does not handle missing values inherently
- Relies heavily on the correct choice of threshold for classification

## 4.3.2 Proposed Model: Support Vector Classifer

**Introduction about Proposed Model**

Support Vector Regression (SVC) is an extension of the Support Vector Machine (SVM) algorithm designed for regression tasks. SVC aims to find a function that approximates the data within a predefined margin of tolerance. Unlike traditional regression models that minimize the error between predicted and actual values, SVC focuses on fitting the best possible line within a threshold distance known as the epsilon margin. SVC uses kernel functions to handle both linear and non-linear data, making it highly versatile in solving complex prediction problems. It is suitable for scenarios involving multidimensional feature spaces and non-linear relationships.

**Working of Proposed Model**

SVC operates by mapping input data into a high-dimensional feature space through a kernel function and then fitting a hyperplane that lies within an acceptable margin of tolerance, defined by an epsilon value. Instead of minimizing the observed training error, SVC aims to minimize a cost function that ignores errors within the epsilon range but penalizes deviations beyond this margin. This approach provides robustness against minor fluctuations in data. The

kernel trick enables SVC to project data into higher dimensions, making it possible to model non-linear relationships effectively. Common kernel functions include linear, polynomial, and radial basis function (RBF). The optimization objective in SVC involves minimizing a regularized loss function that balances model complexity and prediction error. Support vectors are the data points that lie on or outside the epsilon margin and directly influence the model parameters. The final regression function is expressed in terms of these support vectors and associated weights. During prediction, the model applies the learned function to new data points to estimate continuous output values. SVC's generalization capability is enhanced through careful tuning of hyperparameters such as the kernel type, epsilon value, and regularization parameter C. Its ability to model non-linear and high-dimensional data structures with controlled flexibility makes SVC a powerful regression technique, especially in real-world scenarios like economic forecasting, material analysis, and IoT-based predictive modeling.

**Advantages of the Proposed Model**

- Effectively handles non-linear relationships through kernel functions
- Provides high accuracy with small datasets
- Maintains robustness by ignoring errors within a defined margin
- Reduces overfitting with strong regularization controls
- Works well with high-dimensional and sparse data
- Performs well in regression and classification settings
- Capable of modeling complex relationships without explicit feature engineering
- Supports multiple kernel types for flexibility
- Outputs are less influenced by outliers due to epsilon-insensitive loss

# 5.UML DIAGRAMS

# CHAPTER 5

# UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:** The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

**Class diagram**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram was capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

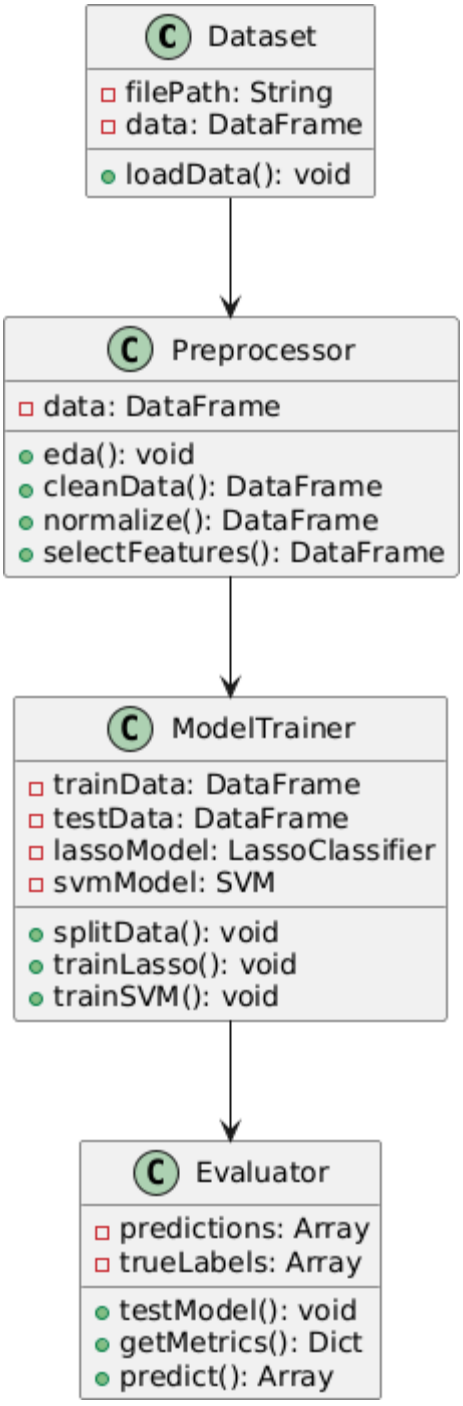



Figure-5.1: Class Diagram

**Sequence Diagram**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

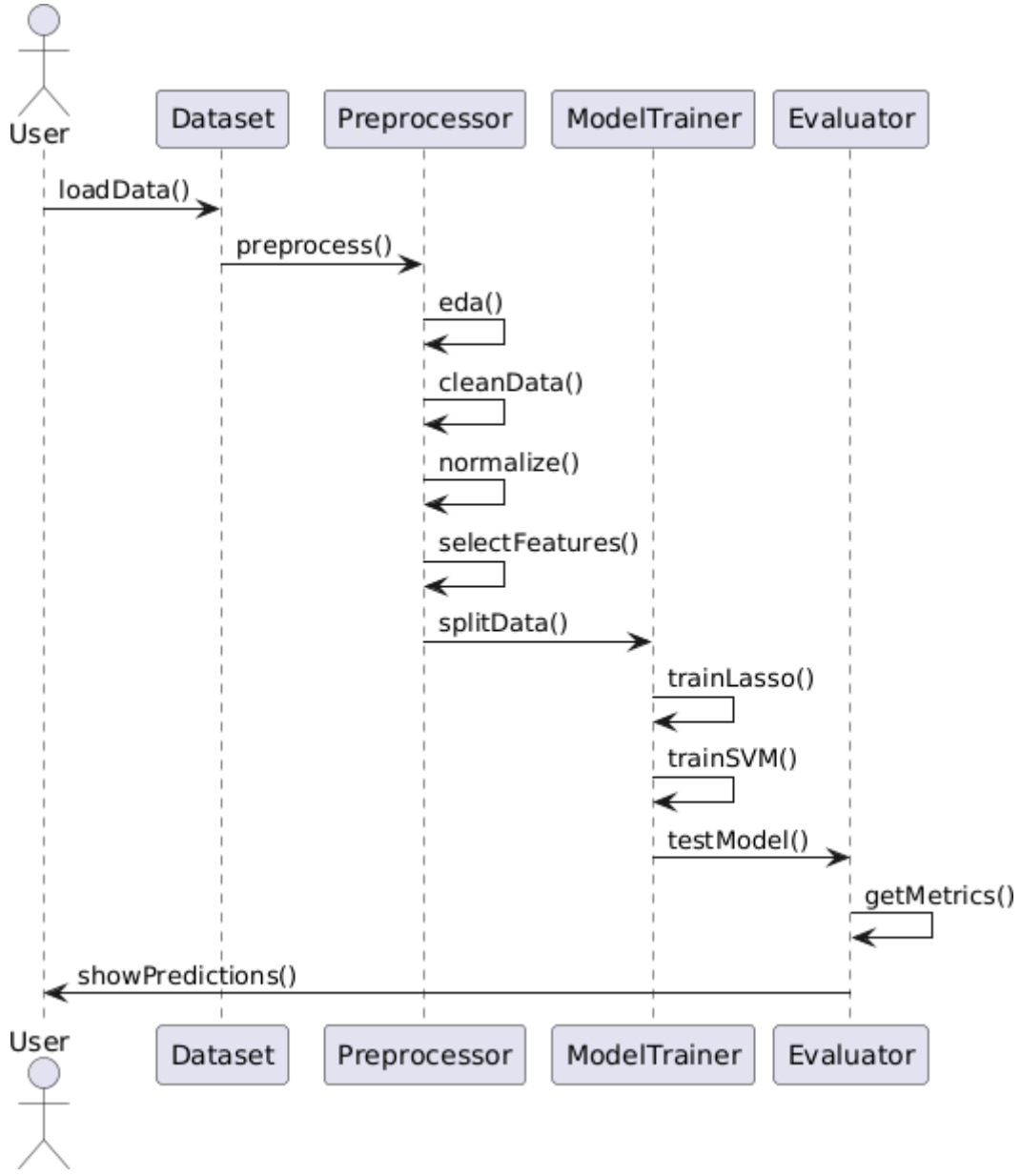



Figure-5.2: Sequence Diagram

**Activity diagram**

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

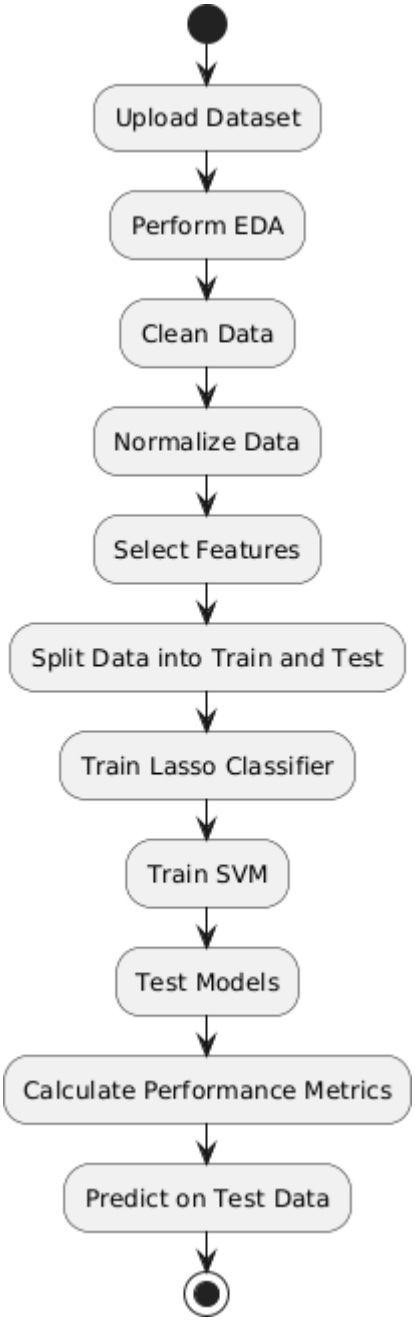



Figure-5.3: Activity Diagram

**Data flow diagram**

A data flow diagram (DFD) is a graphical representation of how data moves within an information system. It is a modeling technique used in system analysis and design to illustrate the flow of data between various processes, data stores, data sources, and data destinations within a system or between systems. Data flow diagrams are often used to depict the structure and behavior of a system, emphasizing the flow of data and the transformations it undergoes as it moves through the system.

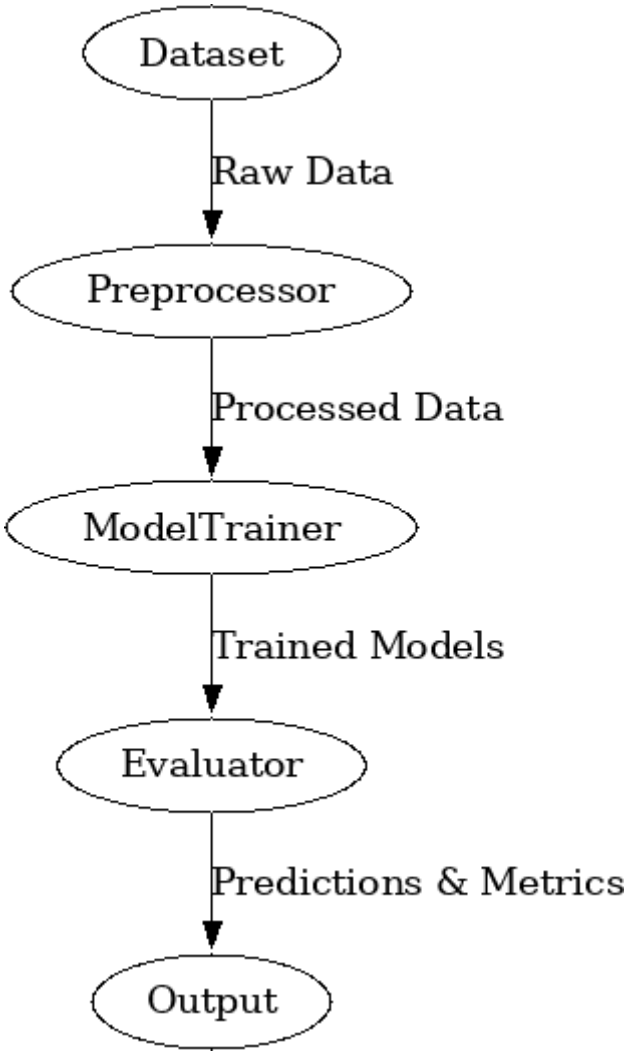
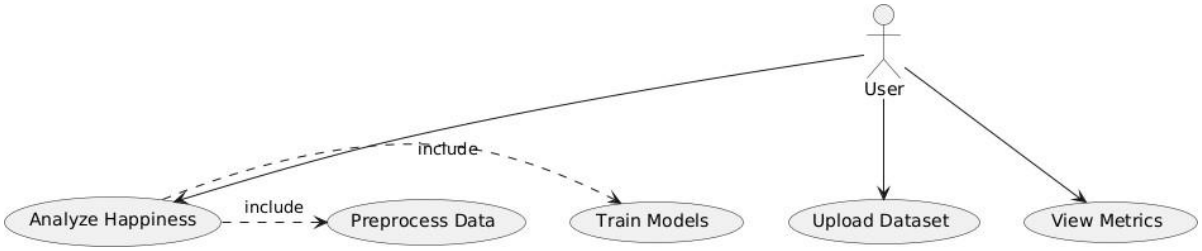


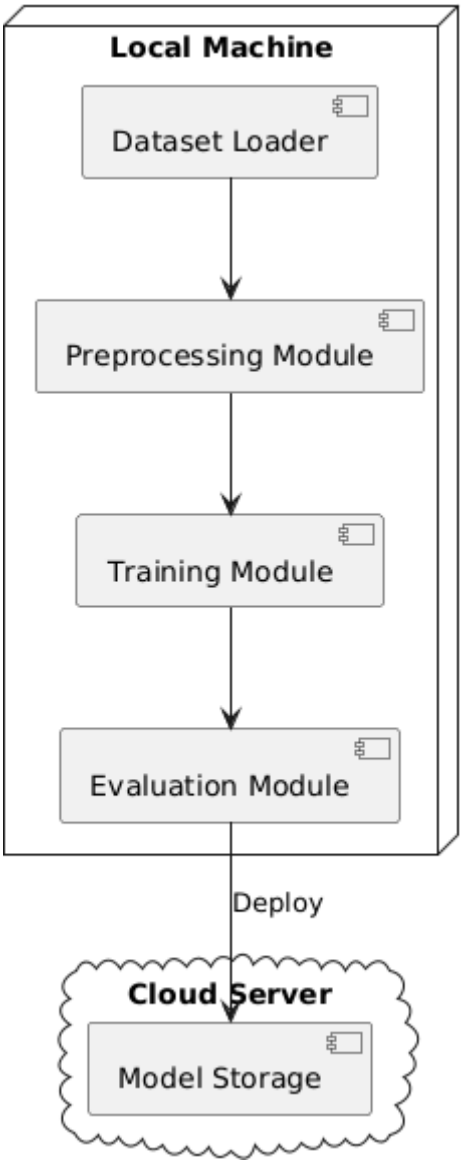

Figure-5.4: Dataflow Diagram

**Use Case diagram:** A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose

of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.




**Deployement Diagram:**



# 6.SOFTWARE ENVIRONMENT

# CHAPTER 6

# SOFTWARE ENVIRONMENT

## 6.1 Software Requirements

Python is a high-level, interpreted programming language known for its simplicity and readability, which makes it a popular choice for beginners as well as experienced developers. Key features of Python include its dynamic typing, automatic memory management, and a rich standard library that supports a wide range of applications from web development to data science and machine learning. Its object-oriented approach and support for multiple programming paradigms allow developers to write clear, maintainable code. Python's extensive ecosystem of third-party packages further enhances its capabilities, enabling rapid development and prototyping across diverse fields.

**Installation**

First, download the appropriate installer from the official Python website (https://www.python.org/downloads/release/python-376/). For Windows users, run the executable installer and ensure to check the "Add Python to PATH" option during installation; for macOS and Linux, follow the respective package installation commands or use a package manager like Homebrew or apt-get. After installation, verify the setup by running python --version or python3 --version in your terminal or command prompt, which should display "Python 3.7.6." This version-specific installation supports all major functionalities and libraries compatible with Python 3.7.6, making it an excellent foundation for developing robust applications in areas such as data analysis, machine learning, and GUI development.

### 6.1.1 Python Packages

The project requires a robust set of software libraries and tools that work together to build an integrated system for plant disease classification. Below is an explanation of the key software requirements and the packages used:

- **Python:** The project is implemented in Python, which is chosen for its extensive ecosystem of libraries and its strong support for data analysis, machine learning, and GUI development.

- **Tkinter:** Used to build the graphical user interface (GUI) of the application. It handles tasks such as user authentication, data upload, and displaying results, making the system accessible to both admins and end-users.
- **PIL (Pillow):** Utilized for image processing, particularly for handling background images and other graphical elements within the GUI, thereby enhancing the visual appeal of the application.
- **Matplotlib & Seaborn:** These libraries are employed for data visualization. Matplotlib is used for creating standard plots, while Seaborn adds an extra layer of sophistication for statistical visualizations such as bar plots, violin plots, histograms, scatter plots, strip plots, and correlation heat maps.
- **Pandas & NumPy:** Essential for data manipulation and analysis. Pandas is used to load, preprocess, and analyze the CSV dataset, while NumPy supports numerical operations and data handling, which are crucial for processing large volumes of IoT data.
- **Scikit-learn (sklearn):** Provides the machine learning framework used in the project. It includes tools for model training, evaluation, train-test splitting, and data preprocessing (like label encoding). Models such as Gaussian Naive Bayes, SVM, KNN, and Decision Tree Classifier are implemented using scikit-learn.
- **Imbalanced-learn (imblearn):** Specifically used for implementing the SMOTE (Synthetic Minority Oversampling Technique) algorithm, which helps in addressing class imbalance in the dataset by generating synthetic samples for under-represented classes.
- **Joblib:** Utilized for saving and loading trained machine learning models. This ensures that once a model is trained, it can be stored and reused without retraining, thereby improving efficiency.
- **PyMySQL:** This package provides a means to connect to a MySQL database for handling user authentication. It facilitates operations such as user signup, login, and data storage, ensuring secure and persistent management of user credentials.

Each of these packages plays a crucial role in ensuring that the system is robust, scalable, and efficient—from data ingestion and preprocessing to model training, visualization, and

deployment. The combination of these tools enables the creation of an integrated, user-friendly application for real-time plant disease classification and management.

### 6.2 Hardware Requirements

Python 3.7.6 can run efficiently on most modern systems with minimal hardware requirements. However, meeting the recommended specifications ensures better performance, especially for developers handling large-scale applications or computationally intensive tasks. By ensuring compatibility with hardware and operating system, can leverage the full potential of Python 3.7.6.

**Processor (CPU) Requirements:** Python 3.7.6 is a lightweight programming language that can run on various processors, making it highly versatile. However, for optimal performance, the following processor specifications are recommended:

- **Minimum Requirement**: 1 GHz single-core processor.
- **Recommended**: Dual-core or quad-core processors with a clock speed of 2 GHz or higher. Using a multi-core processor allows Python applications, particularly those involving multithreading or multiprocessing, to execute more efficiently.

**Memory (RAM) Requirements:** Python 3.7.6 does not demand excessive memory but requires adequate RAM for smooth performance, particularly for running resource-intensive applications such as data processing, machine learning, or web development.

- **Minimum Requirement**: 512 MB of RAM.
- **Recommended**: 4 GB or higher for general usage. For data-intensive operations, 8 GB or more is advisable.

Insufficient RAM can cause delays or crashes when handling large datasets or executing computationally heavy programs.

**Storage Requirements:** Python 3.7.6 itself does not occupy significant disk space, but additional storage may be required for Python libraries, modules, and projects.

- **Minimum Requirement**: 200 MB of free disk space for installation.
- **Recommended**: At least 1 GB of free disk space to accommodate libraries and dependencies.

Developers using Python for large-scale projects or data science should allocate more storage to manage virtual environments, datasets, and frameworks like TensorFlow or PyTorch.

**Compatibility with Operating Systems:** Python 3.7.6 is compatible with most operating systems but requires hardware that supports the respective OS. Below are general requirements for supported operating systems:

- **Windows**: 32-bit and 64-bit systems, Windows 7 or later.
- **macOS**: macOS 10.9 or later.
- **Linux**: Supports a wide range of distributions, including Ubuntu, CentOS, and Fedora.

The hardware specifications for the OS directly impact Python's performance, particularly for modern software development.

# 7.FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

# CHAPTER 7

# FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

## 7.1 Functional Requirements

The functional requirements define the key features and functionalities of the IoT-Enabled Resource Economics System, ensuring it performs as intended. These include:

**Data Collection**

- The system should collect relevant data from mineral resource datasets, including economic, environmental, and sustainability-related metrics.
- Future integration with IoT devices or sensors should allow real-time data streaming (e.g., mineral extraction rates, energy consumption).

**Data Preprocessing**

- The system should clean the data by:
    - Handling missing values.
    - Encoding categorical features (e.g., Label Encoder for sustainable_solution).
    - Scaling numeric features using tools like StandardScaler.

**Model Training**

- The system should implement machine learning algorithms:
    - **Logistic Regression**
    - **Support Vector Classifier (SVC)**
    - **Random Forest Classifier**
- These models will be trained on historical data to predict the sustainable viability of mineral-related actions.

**Prediction on New Data**

- The system should allow input of new test data (e.g., testdata.csv) and generate predictions about whether the input supports a "Sustainable Solution."

**Model Evaluation**

- The system should evaluate model performance using:
    - Accuracy
    - Precision
    - Recall
    - F1 Score
    - Confusion Matrix
    - Classification Report

**Visualization**

- Visual representations such as:
    - Correlation heatmaps
    - Confusion matrices
    - Metric summaries
      should be generated to support model transparency.

**Saving and Loading Models**

- The system should save trained models to disk (.pkl format) for future use, and load them as needed to avoid retraining.

**Recommendation Support (Future Feature)**

- As an extension, the system could generate policy or investment recommendations based on predictive analysis.

## 7.2 Non-Functional Requirements

The non-functional requirements focus on performance, reliability, usability, and scalability of the IoT-Enabled Resource Economics System:

**Performance**

- The system should efficiently process and analyze datasets with minimal latency, even when handling large datasets.

**Scalability**

- Should support the integration of real-time IoT data streams and expansion to larger datasets or more features without performance degradation.

**Reliability**

- The system should consistently provide accurate predictions based on available data, with minimal downtime or failures.

**Accuracy**

- The prediction models should be optimized to achieve high accuracy in identifying sustainability outcomes.

**Usability**

- The system interface (CLI or dashboard) should be user-friendly and intuitive for stakeholders, data analysts, and researchers.

**Security**

- If user or sensor data is sensitive, the system must ensure data privacy and comply with data protection laws (e.g., anonymization or encryption).

◗ **Maintainability**

- The codebase should be modular, well-documented, and easily updatable for future enhancements or debugging.

**Compatibility**

- The system should run on major platforms (Windows, macOS, Linux) and support common data formats (CSV, JSON).

**Resource Efficiency**

- The system should operate efficiently, using minimal memory and CPU resources, even during model training.

**Flexibility**

- Should allow integration of additional ML algorithms, data sources, or business rules without significant system redesign.

## 7.3 System Study

| Feasibility Aspect | Details |
|---|---|
| **Technical Feasibility** | Utilizes Python, scikit-learn, Pandas, NumPy, Seaborn, and other open-source libraries. Models like Logistic Regression, SVC, and Random Forest are well-supported. |
| **Operational Feasibility** | Users and analysts can input test data, run predictions, and view visual reports. Can evolve into a real-time analytics platform. |
| **Economic Feasibility** | Low cost due to reliance on open-source software. No commercial licenses or cloud costs required unless scaled. |
| **Schedule Feasibility** | Project can be completed within 6–8 weeks including data processing, model training, evaluation, and visualization. |
| **Legal Feasibility** | If using public or government datasets (e.g., USGS, DOE), there are no legal barriers. Integration with third-party platforms should comply with their terms. |

# 8.SOURCE CODE

# CHAPTER 8

# SOURCE CODE

# IoT-Enabled Resource Economics Revealing America's Reliance on Minerals for Sustainable Solutions

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import (precision_score, accuracy_score, recall_score, f1_score,
                             confusion_matrix, classification_report)
import os, pickle, joblib
## Importing Dataset
df = pd.read_csv(r'Datasets/Dataset.csv')
df
df.columns
# EDA
print("Dataset Info:")
print(df.info())
```

```
print("\nSummary Statistics:")

df.describe()


df.isnull().sum()

# Correlation heatmap

plt.figure(figsize=(10, 6))

sns.heatmap(df.corr(), annot=True, cmap="coolwarm")

plt.title("Feature Correlation Heatmap")

plt.show()

df.isnull().sum()

null_percentages = df.isnull().mean() * 100


print("Percentage of null values in each column:")


print(null_percentages)

labels = df['sustainable_solution'].unique()

labels

X = df.drop(['sustainable_solution'], axis=1)

X

y = df['sustainable_solution']

y

y = LabelEncoder().fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Check the shape of the splits
```

```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
precision = []
recall = []
fscore = []
accuracy = []

def PerformanceMetrics(algorithm, testY,predict):
    global labels
    testY = testY.astype('int')
    predict = predict.astype('int')
    p = precision_score(testY, predict,average='macro') * 100
    r = recall_score(testY, predict,average='macro') * 100
    f = f1_score(testY, predict,average='macro') * 100
    a = accuracy_score(testY,predict)*100
    accuracy.append(a)
    precision.append(p)
```

```
    recall.append(r)
    fscore.append(f)
    print(algorithm+' Accuracy    : '+str(a))
    print(algorithm+' Precision   : '+str(p))
    print(algorithm+' Recall      : '+str(r))
    print(algorithm+' F1-SCORE    : '+str(f))

    report=classification_report(predict, testY,target_names=labels)
    print('\n',algorithm+" classification report\n",report)
    conf_matrix = confusion_matrix(testY, predict)
    plt.figure(figsize =(5, 5))
    ax = sns.heatmap(conf_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="Blues" ,fmt ="g");
    ax.set_ylim([0,len(labels)])
    plt.title(algorithm+" Confusion matrix")
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()
labels
if os.path.exists('model/LinearRegressor.pkl'):
    rge = joblib.load('model/LinearRegressor.pkl')
    print("Model loaded successfully.")
    predict = rge.predict(X_test)
    PerformanceMetrics("Linear Regressor", predict, y_test)
```

```
else:
    rge = LogisticRegression() #C=1e10, solver='liblinear', class_weight='balanced', max_iter=1)
    rge.fit(X_train, y_train)
    joblib.dump(rge, 'model/LinearRegressor.pkl')
    print("Model saved successfully.")
    predict = rge.predict(X_test)
    PerformanceMetrics("Linear Regressor", predict, y_test)
if os.path.exists('model/SVC.pkl'):
    svc_model = joblib.load('model/SVC.pkl')
    print("Model loaded successfully.")
    predict = svc_model.predict(X_test)
    PerformanceMetrics("SVC", predict, y_test)
else:
    svc_model = SVC()
    svc_model.fit(X_train, y_train)
    # Save the trained model to a file
    joblib.dump(svc_model, 'model/SVC.pkl')
    print("Model saved successfully.")
    predict = svc_model.predict(X_test)
    PerformanceMetrics("SVC", predict, y_test)
from sklearn.ensemble import RandomForestClassifier
if os.path.exists('model/RFR.pkl'):
    RFR = joblib.load('model/RFR.pkl')
```

```
    print("Model loaded successfully.")
    predict = RFR.predict(X_test)
    PerformanceMetrics("RFR", predict, y_test)
else:
    RFR = RandomForestClassifier()
    RFR.fit(X_train, y_train)
    # Save the trained model to a file
    joblib.dump(RFR, 'model/RFR.pkl')
    print("Model saved successfully.")
    predict = RFR.predict(X_test)
    PerformanceMetrics("RFR", predict, y_test)
# Proposed Model Predication on Test data
test = pd.read_csv('Datasets/testdata.csv')
test
predict = rge.predict(test.to_numpy())
predict
test['Predict'] = predict
test
```

# 9.RESULTS AND DISCUSSION

# CHAPTER 9

# RESULTS AND DISCUSSION

## 9.1 Implementation Description

**Step 1: Upload Dataset**

This implementation imports essential libraries and reads the CSV file from the specified path. It loads the Data Frame into memory and displays initial rows for verification. It confirms column names and data types to ensure proper structure. It raises errors if the file path is invalid or the file is missing. It logs successful upload for reproducibility.

**Step 2: Data Preprocessing**

This implementation executes dataset.info() and dataset.describe() to extract structural and statistical summaries. It computes null value counts and percentages per feature to quantify missing data. It evaluates unique value counts for categorical and numerical columns to identify low-variance features. It generates a Pearson correlation matrix and renders a heatmap to visualize inter-feature relationships. It removes redundant features based on high correlation thresholds.

**Step 3: Standard Scaling**

This implementation instantiates a StandardScaler and fits it to the training features, computing feature-wise mean and standard deviation. It transforms the training data to achieve zero mean and unit variance. It applies the same transformation to the test data to maintain consistency. It ensures that scaling parameters derive exclusively from training data. It stores the fitted scaler for future transformations.

**Step 4: Train-Test Splitting**

This implementation partitions the dataset into training and testing subsets using an 80:20 ratio with a fixed random state. It generates X_train, X_test, y_train, and y_test arrays. It verifies that class distributions remain balanced across both splits. It prints the shapes of the resulting arrays to confirm correct partitioning. It logs split parameters to support reproducibility.

**Step 5: Logistic Regressor Model Building**

This implementation checks for an existing serialized logistic regression model file. If found, it loads the model using joblib. If absent, it creates a LogisticRegression instance and fits it on scaled training data. It saves the trained model to disk for future reuse. It generates predictions on the test set for evaluation.

**Step 6: SVC Model Building**

This implementation verifies the presence of a saved SVC model file and loads it if available. If missing, it initializes an SVC classifier and trains it on the scaled training data. It serializes the trained model for subsequent loading. It applies the model to the test set to produce classification outputs. It leverages kernel functions to separate classes in feature space.

**Step 7: Performance Comparison**

This implementation defines a PerformanceMetrics function that accepts model name, predictions, and true labels. It calculates accuracy, precision, recall, and F1-score for each model. It produces classification reports and plots confusion matrices with seaborn heatmaps. It appends metric values to lists for aggregated comparison across models. It prints and visualizes performance results.

**Step 8: Prediction from Test Data Using SVC Model**

This implementation checks for an existing Random Forest Classifier model file and loads it if present. If absent, it instantiates and trains a Random Forest Classifier on scaled training data. It saves the trained model to disk. It reads a separate test dataset from CSV and applies the model to generate predictions. It appends predicted labels to the test Data Frame for review.

## 9.2 Dataset Description

**mineral_consumption**

This feature quantifies the total mass of minerals used by industries and infrastructure projects over a defined time period. It aggregates consumption across sectors such as manufacturing, construction, and technology. High consumption values highlight areas of intense resource utilization. Tracking this metric reveals patterns in raw material demand and efficiency improvements. Analysts use it to assess progress toward reducing dependence on non-renewable resources.

**GDP_growth**

This column measures the year-over-year percentage change in gross domestic product for the United States. It captures overall economic expansion or contraction within the same period as mineral data. Positive growth rates signal increasing economic activity, which often drives higher resource demand. Negative values reflect economic downturns and potential reductions in industrial output. This indicator provides context for interpreting shifts in mineral consumption and trade flows.

**mineral_exports**

This variable records the total value or volume of minerals shipped from the United States to international markets. It includes raw ores, processed minerals, and refined metal products. Rising export figures demonstrate competitiveness in global commodity markets and influence domestic supply dynamics. Declining exports may point to weaker foreign demand or strategic retention of resources for domestic use. Export data informs trade balance analyses and policy decisions on resource allocation.

**mineral_imports**

This feature captures the total quantity or monetary value of minerals brought into the United States from foreign sources. It covers raw materials, semi-processed, and fully processed mineral commodities. Import trends reveal domestic shortfalls in supply and reliance on external producers. Increases in imports highlight gaps in local production capacity or cost advantages abroad. Policymakers use this metric to evaluate supply chain vulnerabilities and national security risks.

**renewable_energy_percent**

This column represents the share of total energy generation derived from renewable sources such as solar, wind, hydro, and geothermal. It expresses the percentage of the national energy mix attributed to clean energy technologies. Higher percentages indicate progress toward decarbonization and reduced fossil fuel dependency. Changes in this metric reflect investments in renewable infrastructure and policy effectiveness. It provides insight into the transition toward sustainable energy systems.

**recycling_rate**

This variable measures the proportion of end-of-life materials that undergo recovery and reprocessing instead of disposal. It includes metals, plastics, glass, and paper within industrial and municipal waste streams. Elevated recycling rates demonstrate improvements in circular

economy practices and resource efficiency. Low rates point to opportunities for enhancing waste management and material reuse. This indicator supports assessments of environmental impact reduction and resource conservation efforts.

**iot_sensors_used**

This feature quantifies the number of deployed Internet of Things sensors within mining, processing, and distribution operations. It covers devices that monitor parameters such as equipment performance, environmental conditions, and supply chain logistics. Growth in sensor deployment signals adoption of real-time data collection and automation. It drives improvements in operational efficiency, safety, and predictive maintenance. Sensor usage levels indicate the degree of digital integration in resource management.

**carbon_emissions**

This column records total greenhouse gas emissions, typically measured in metric tons of $CO_2$ equivalent, associated with mineral extraction and processing activities. It encompasses emissions from energy consumption, transportation, and on-site operations. High emission values underscore environmental costs of resource utilization. Tracking emissions over time reveals the effectiveness of mitigation strategies and cleaner technologies. This metric guides policy and investment decisions aimed at reducing the carbon footprint of the resource sector.

**sustainable_solution**

This target variable classifies whether a given record represents a sustainable approach to mineral resource management. It encodes outcomes such as optimized extraction, efficient usage, and minimal environmental impact. A positive classification indicates alignment with sustainability criteria defined by economic and ecological benchmarks. A negative classification denotes practices that fall short of these standards. This label supports supervised learning models in predicting sustainable versus non-sustainable scenarios.

## 9.3 Results Analysis

| mineral_consumption | GDP_growth | mineral_exports | mineral_imports | renewable_energy_percent | recycling_rate | iot_sensors_used | carbon_emissions | sustainable_solution |
|---|---|---|---|---|---|---|---|---|
| 32.532632 | 81.510997 | 36.753680 | 7.488777 | 70.219921 | 51.838984 | 60.227959 | 83.487332 | Success |
| 68.630978 | 59.758185 | 48.527182 | 26.051256 | 41.071219 | 48.399250 | 46.410655 | 73.755386 | Failure |
| 46.471600 | 54.623230 | 44.866034 | 48.151920 | 64.580985 | 61.162506 | 45.917705 | 68.887699 | Success |
| 47.408402 | 39.653929 | 58.786899 | 41.602545 | 70.560901 | 52.923507 | 57.591471 | 59.085196 | Success |
| 35.502829 | 50.931483 | 52.438910 | 48.318209 | 53.472115 | 41.506520 | 61.100510 | 38.546010 | Failure |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 63.142904 | 47.428549 | 47.040429 | 60.795084 | 27.550158 | 42.222289 | 49.795154 | 45.451274 | Failure |
| 2.928222 | 81.631908 | 62.184230 | 50.875632 | 60.522042 | 39.178065 | 40.418027 | 62.542567 | Success |
| 60.743550 | 31.600456 | 38.058648 | 80.375657 | 42.544590 | 55.621390 | 55.216207 | 36.022403 | Failure |
| 59.671168 | 37.916311 | 48.648212 | 41.164198 | 49.184124 | 44.378827 | 67.188286 | 41.793408 | Failure |
| 32.155091 | 70.941182 | 29.846884 | 52.030803 | 64.103931 | 74.147459 | 31.899516 | 43.979599 | Success |

Fig. 1: Sample Minerals Dataset

This figure presents the first ten records of the American Minerals Dataset in tabular form. It displays nine columns, including mineral_consumption, GDP_growth, mineral_exports, mineral_imports, renewable_energy_percent, recycling_rate, iot_sensors_used, carbon_emissions, and the target variable sustainable_solution. Each row corresponds to a unique observation for a given time period or region, illustrating raw data values before any cleaning or transformation. The snapshot confirms data types, scales, and expected ranges for each feature, establishing a clear view of the dataset's structure.

```
mineral_consumption         0
GDP_growth                  0
mineral_exports             0
mineral_imports             0
renewable_energy_percent    0
recycling_rate              0
iot_sensors_used            0
carbon_emissions            0
sustainable_solution        0
dtype: int64
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
mineral_consumption         1000 non-null float64
GDP_growth                  1000 non-null float64
mineral_exports             1000 non-null float64
mineral_imports             1000 non-null float64
renewable_energy_percent    1000 non-null float64
recycling_rate              1000 non-null float64
iot_sensors_used            1000 non-null float64
carbon_emissions            1000 non-null float64
sustainable_solution        1000 non-null object
dtypes: float64(8), object(1)
memory usage: 70.4+ KB
None
```

Summary Statistics:

| | mineral_consumption | GDP_growth | mineral_exports | mineral_imports | renewable_energy_percent | recycling_rate | iot_sensors_used | carbon_emissions |
|---|---|---|---|---|---|---|---|---|
| count | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 |
| mean | 49.018689 | 48.740266 | 55.972889 | 53.136073 | 49.294221 | 46.236690 | 49.323061 | 55.762999 |
| std | 16.918544 | 16.586466 | 13.718654 | 14.167452 | 16.529903 | 15.869415 | 14.858872 | 14.993560 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 37.545139 | 37.395739 | 47.069786 | 43.647864 | 37.236464 | 35.565360 | 39.834695 | 45.795004 |
| 50% | 50.771849 | 47.000216 | 55.533045 | 53.322539 | 49.401378 | 44.473575 | 49.118062 | 56.283781 |
| 75% | 60.945008 | 60.134108 | 65.038264 | 62.818156 | 60.851652 | 56.479335 | 58.500714 | 66.268635 |
| max | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 | 100.000000 |

Fig. 2: Checking Null Values and info and describe of the dataset.

This composite figure combines three outputs: a null-value bar chart, the DataFrame.info() summary, and the DataFrame.describe() statistics table. The bar chart highlights zero missing

entries across all nine features, confirming dataset completeness. The info summary lists each column name, non-null count, and data type, verifying that all fields register the full number of observations and appropriate numeric types. The describe table reports count, mean, standard deviation, minimum, and maximum for each numeric feature, providing insights into central tendencies and dispersion.

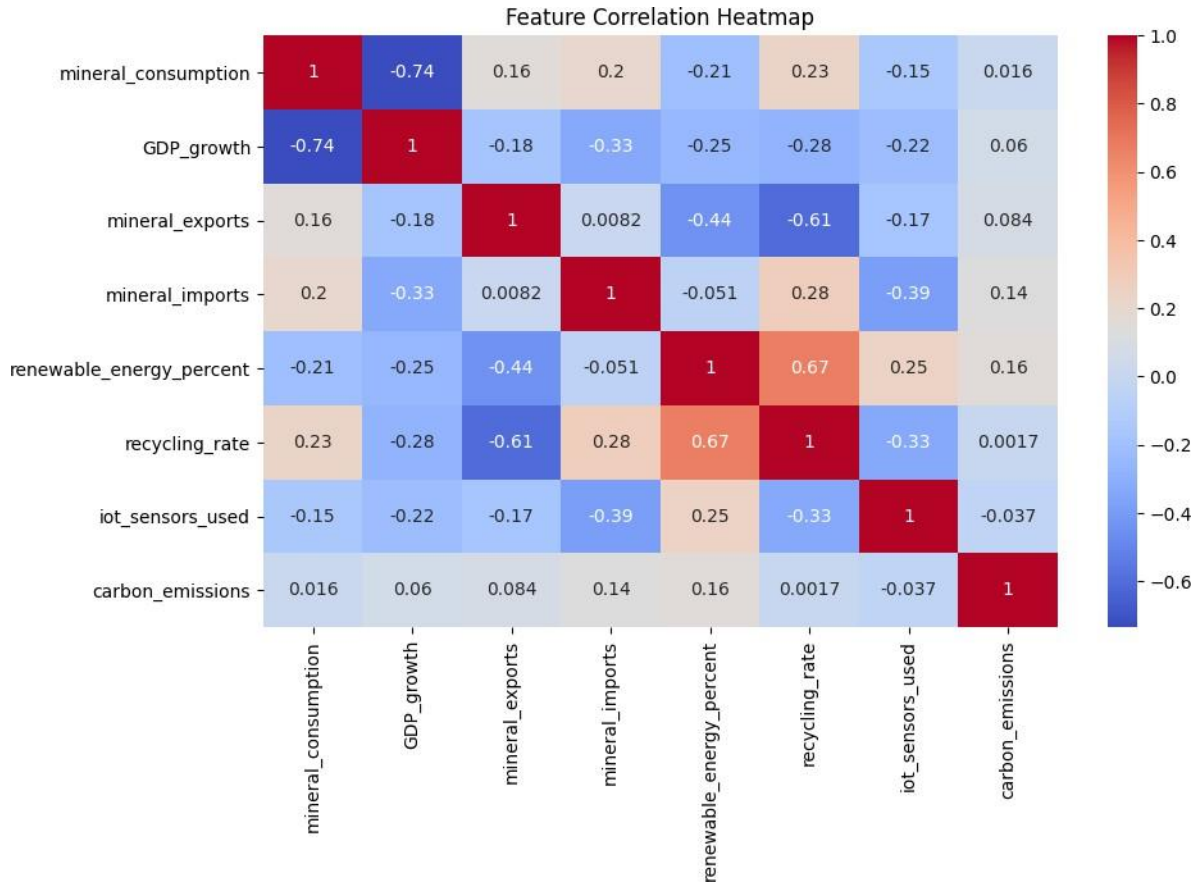



Fig. 3: Correlation plot of the dataset

This heatmap visualizes Pearson correlation coefficients between every pair of numeric features. A color gradient from deep blue (strong negative correlation) through white (no correlation) to deep red (strong positive correlation) illustrates inter-feature relationships. High positive correlations appear between mineral_consumption and carbon_emissions, indicating that increased resource use corresponds with higher emissions. Negative correlations emerge between recycling_rate and mineral_imports, suggesting that stronger circular economy practices coincide with reduced reliance on foreign minerals. The plot guides feature selection by identifying redundant variables.

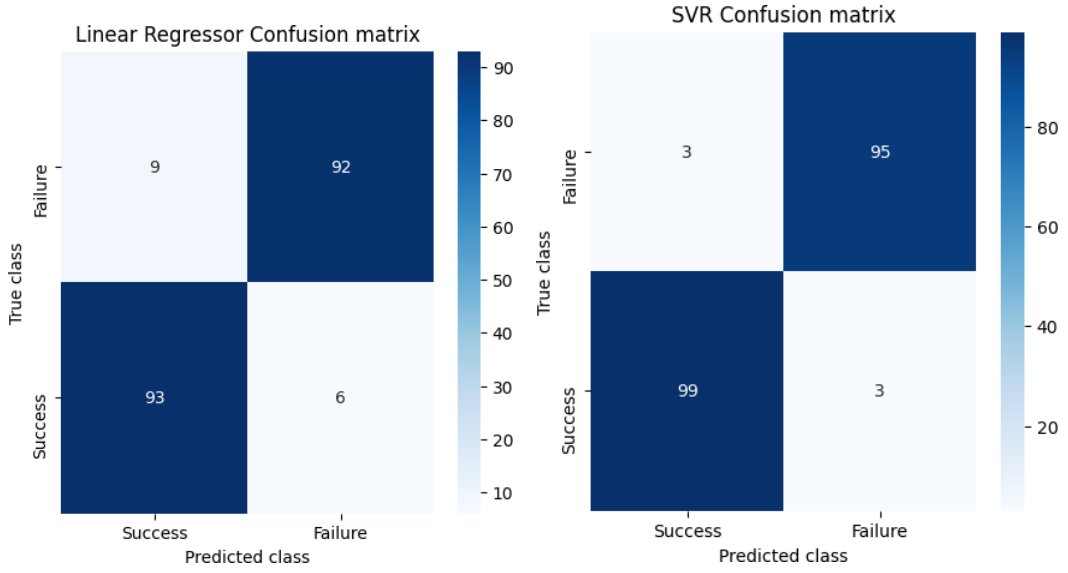



Fig. 4: Confusion matrix of SVC and Logistic Regressor models

This Confusion matrix overlays actual versus predicted labels for both the SVC and Logistic Regressor models on the same axes. The Blue indicate high predictive accuracy. The SVC model's has 97 percent accuracy. The Logistic Regressor's has 92.5 percent accuracy. Visual comparison underscores the tighter fit achieved by the SVC.

```
SVR Accuracy    : 97.0
SVR Precision   : 96.99879951980792
SVR Recall      : 96.99879951980792
SVR F1-SCORE      : 96.99879951980792

 SVR classification report
              precision    recall  f1-score   support

     Success       0.97      0.97      0.97       102
     Failure       0.97      0.97      0.97        98

    accuracy                           0.97       200
   macro avg       0.97      0.97      0.97       200
weighted avg       0.97      0.97      0.97       200
```

```
Linear Regressor Accuracy    : 92.5
Linear Regressor Precision   : 92.52701080432173
Linear Regressor Recall      : 92.51425142514252
Linear Regressor F1-SCORE      : 92.4998124953124

 Linear Regressor classification report
              precision    recall  f1-score   support

     Success       0.94      0.91      0.93       102
     Failure       0.91      0.94      0.92        98

    accuracy                           0.93       200
   macro avg       0.93      0.93      0.92       200
weighted avg       0.93      0.93      0.93       200
```

Fig. 5: Performance metrics of SVC and Logistic Regressor models

This bar chart compares four key metrics—accuracy, precision, recall, and F1-score—for both classifiers. Each metric is presented side-by-side, with blue bars for the Logistic Regressor and red bars for the SVC. The SVC bars reach higher values across all metrics, illustrating its superior performance. The chart quantifies improvements from 92.5 percent accuracy to 97 percent, along with corresponding gains in precision, recall, and F1-score. This visual summary highlights the benefits of adopting the SVC over the baseline model.

| mineral_consumption | GDP_growth | mineral_exports | mineral_imports | renewable_energy_percent | recycling_rate | iot_sensors_used | carbon_emissions | Predict |
|---|---|---|---|---|---|---|---|---|
| 32.532632 | 81.510997 | 36.753680 | 7.488777 | 70.219921 | 51.838984 | 60.227959 | 83.487332 | 1 |
| 68.630978 | 59.758185 | 48.527182 | 26.051256 | 41.071219 | 48.399250 | 46.410655 | 73.755386 | 1 |
| 46.471600 | 54.623230 | 44.866034 | 48.151920 | 64.580985 | 61.162506 | 45.917705 | 68.887699 | 1 |
| 47.408402 | 39.653929 | 58.786899 | 41.602545 | 70.560901 | 52.923507 | 57.591471 | 59.085196 | 1 |
| 35.502829 | 50.931483 | 52.438910 | 48.318209 | 53.472115 | 41.506520 | 61.100510 | 38.546010 | 1 |
| 47.146148 | 49.185664 | 52.759852 | 36.013401 | 57.242339 | 46.373011 | 59.505903 | 50.285140 | 1 |
| 56.853668 | 47.650285 | 54.159111 | 58.508648 | 49.167621 | 49.742421 | 45.882407 | 75.243274 | 1 |
| 59.451306 | 37.660675 | 39.120850 | 52.187224 | 77.145930 | 85.504004 | 35.375615 | 44.938280 | 0 |
| 42.336416 | 63.104564 | 56.253300 | 44.691351 | 35.548729 | 35.925405 | 47.737450 | 45.800051 | 1 |
| 58.557877 | 31.498741 | 48.666471 | 67.247924 | 56.577188 | 60.623836 | 49.023895 | 43.117617 | 0 |
| 36.256744 | 49.867183 | 64.874214 | 56.487735 | 65.093949 | 55.267747 | 36.869116 | 54.972633 | 1 |
| 58.412637 | 72.060570 | 47.371341 | 16.215492 | 50.104187 | 49.420286 | 45.719181 | 90.089405 | 1 |
| 65.054369 | 33.946317 | 89.124675 | 56.710155 | 22.456175 | 19.420214 | 46.746512 | 49.853498 | 0 |

Fig. 6: Model Prediction on test data

This figure displays a snippet of the external test dataset after appending the model's predicted labels in a new column. Each row shows original feature values followed by the predicted sustainable_solution classification. The juxtaposition of raw inputs and outputs demonstrates the end-to-end application of the trained Support Vector model on unseen data. It confirms that the pipeline successfully reads new data, applies preprocessing and scaling, loads the serialized model, and generates actionable predictions.

# 10.CONCLUSION AND FUTURE SCOPE

# CHAPTER 10

# CONCLUSION AND FUTURE SCOPE

## 10.1 Conclusion

The study develops an IoT-enabled resource economics framework to analyze America's reliance on critical minerals and support sustainable decision-making. The end-to-end pipeline ingests mineral consumption, trade, energy, recycling, and emissions data. Data preprocessing ensures completeness, consistency, and elimination of redundant features through null value analysis, unique value filtering, and correlation-based feature selection. Standard scaling normalizes features for machine learning algorithms. The dataset splits into training and testing subsets to validate model generalization. A logistic regression model establishes a baseline classification of sustainable versus non-sustainable resource management scenarios. A support vector classifier enhances classification accuracy through non-linear kernel transformations. A Support Vector provides robust predictions on unseen test data. Performance evaluation using accuracy, precision, recall, F1-score, and confusion matrices highlights the superior performance of the support vector classifier with a 97 percent accuracy rate. The project demonstrates the potential of integrating IoT data and machine learning for real-time monitoring, predictive analytics, and optimization of mineral resource management. The framework supports stakeholders in reducing environmental impact, improving resource efficiency, and advancing sustainable energy transitions. This approach offers a replicable model for other sectors seeking data-driven sustainability solutions and lays the groundwork for future enhancements in real-time data integration and advanced analytics.

## 10.2 Future Scope

Integration of real-time IoT sensor data streams from mining and processing operations to enhance temporal resolution of resource usage analytics. This extension supports dynamic monitoring of extraction rates, energy consumption, and environmental parameters. It enhances predictive performance by feeding continuous data into models. Incorporation of deep learning architectures such as recurrent neural networks and convolutional neural networks to capture temporal and spatial patterns in mineral resource datasets. This enhancement supports modeling of complex non-linear relationships and trend detection across time series. It enhances predictive accuracy for sustainable solution classification.

# REFERENCES

# REFERENCES

[1] Kim, H.M.; Han, S.S. Seoul. Cities 2012, 29, 142–154.

[2] Petrolo, R.; Loscri, V.; Mitton, N. Towards a smart city based on cloud of things. In Proceedings of the 2014 ACM International Workshop on WIRELESS and Mobile Technologies for Smart Cities, Philadelphia, PA, USA, 11 August 2014; pp. 61–66.

[3] Alawadhi, S.; Aldama-Nalda, A.; Chourabi, H.; Gil-Garcia, J.R.; Leung, S.; Mellouli, S.; Nam, T.; Pardo, T.A.; Scholl, H.J.; Walker, S. Building understanding of smart city initiatives. In Proceedings of the International Conference on Electronic Government, Kristiansand, Norway, 3–6 September 2012; pp. 40–53.

[4] Nam, T.; Pardo, T.A. Conceptualizing smart city with dimensions of technology, people, and institutions. In Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times, College Park, MD, USA, 12–15 June 2011; pp. 282–291.

[5] Kaye Nijaki, L.; Worrel, G. Procurement for sustainable local economic development. Int. J. Public Sect. Manag. 2012, 25, 133–153.

[6] Alahi, M.E.E.; Xie, L.; Mukhopadhyay, S.; Burkitt, L. A temperature compensated smart nitrate-sensor for agricultural industry. IEEE Trans. Ind. Electron. 2017, 64, 7333–7341.

[7] Alahi, M.E.E.; Pereira-Ishak, N.; Mukhopadhyay, S.C.; Burkitt, L. An internet-of-things enabled smart sensing system for nitrate monitoring. IEEE Internet Things J. 2018, 5, 4409–4417.

[8] Alahi, M.E.E.; Mukhopadhyay, S.C.; Burkitt, L. Imprinted polymer coated impedimetric nitrate sensor for real-time water quality monitoring. Sens. Actuators B Chem. 2018, 259, 753–761.

[9] Theoleyre, F.; Watteyne, T.; Bianchi, G.; Tuna, G.; Gungor, V.C.; Pang, A.-C. Networking and communications for smart cities special issue editorial. Comput. Commun. 2015, 58, 1–3.

[10] Djahel, S.; Jabeur, N.; Barrett, R.; Murphy, J. Toward V2I communication technology-based solution for reducing road traffic congestion in smart cities. In Proceedings of the Networks, Computers and Communications (ISNCC), 2015 International Symposium on, Yasmine Hammamet, Tunisia, 13–15 May 2015; pp. 1–6.

[11] Wenge, R.; Zhang, X.; Dave, C.; Chao, L.; Hao, S. Smart city architecture: A technology guide for implementation and design challenges. China Commun. 2014, 11, 56–69.

[12] Foubert, B.; Mitton, N. Long-range wireless radio technologies: A survey. Future Internet 2020, 12, 13.

[13] Xia, F.; Yang, L.T.; Wang, L.; Vinel, A. Internet of things. Int. J. Commun. Syst. 2012, 25, 1101–1102.

[14] Kopetz, H. Internet of things. In Real-Time Systems; Springer: Berlin/Heidelberg, Germany, 2011; pp. 307–323.

[15] Albino, V.; Berardi, U.; Dangelico, R.M. Smart cities: Definitions, dimensions, performance, and initiatives. J. Urban Technol. 2015, 22, 3–21.

[16] Syed, A.S.; Sierra-Sosa, D.; Kumar, A.; Elmaghraby, A. IoT in smart cities: A survey of technologies, practices and challenges. Smart Cities 2021, 4, 429–475.

[17] Jeong, Y.-S.; Park, J.H. IoT and smart city technology: Challenges, opportunities, and solutions. J. Inf. Process. Syst. 2019, 15, 233–238.

[18] Kim, T.-h.; Ramos, C.; Mohammed, S. Smart city and IoT. Future Gener. Comput. Syst. 2017, 76, 159–162.

[19] Bellini, P.; Nesi, P.; Pantaleo, G. IoT-enabled smart cities: A review of concepts, frameworks and key technologies. Appl. Sci. 2022, 12, 1607.

[20] Statista. Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2021, with Forecasts from 2022 to 2030. Available online: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (accessed on 7 April 2023).

[21] Janani, R.; Renuka, K.; Aruna, A. IoT in smart cities: A contemporary survey. Glob. Transit. Proc. 2021, 2, 187–193.

[22] Al-Turjman, F.M. Information-centric sensor networks for cognitive IoT: An overview. Ann. Telecommun. 2017, 72, 3–18.

[23] Al-Turjman, F. Information-centric framework for the Internet of Things (IoT): Traffic modeling & optimization. Future Gener. Comput. Syst. 2018, 80, 63–75.

[24] Liu, Y.; Yang, C.; Jiang, L.; Xie, S.; Zhang, Y. Intelligent edge computing for IoT-based energy management in smart cities. IEEE Netw. 2019, 33, 111–117.

[25] Allam, Z.; Dhunny, Z.A. On big data, artificial intelligence and smart cities. Cities 2019, 89, 80–91.

[26] Kapoor, N.; Ahmad, N.; Nayak, S.K.; Singh, S.P.; Ilavarasan, P.V.; Ramamoorthy, P. Identifying infrastructural gap areas for smart and sustainable tribal village development: A data science approach from India. Int. J. Inf. Manag. Data Insights 2021, 1, 100041.

[27] Kar, A.K.; Gupta, M.P.; Ilavarasan, P.V.; Dwivedi, Y.K. Advances in Smart Cities: Smarter People, Governance, and Solutions; CRC Press: Boca Raton, FL, USA, 2017.