ELECTRA SYNERGY : A MACHINE LEARNING APPROACH FOR EV CHARGING OPTIMIZATION

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	LIST OF FIGURES	i
	LIST OF SYMBOLS	ii
	LIST OF ABBREVIATIONS	vi
	ABSTRACT	vii
1.	**CHAPTER 1 : INTRODUCTION**	1
	1.1 GENERAL	1
	1.2 SCOPE OF THE PROJECT	2
	1.3 OBJECTIVE	2
	1.4 EXISTING SYSTEM	2
	1.4.1 EXISTING SYSTEM DISADVANTAGES	3
	1.5 LITERATURE SURVEY	3
	1.6 PROPOSED SYSTEM	6
	1.6.1 PROPOSED SYSTEM ADVANTAGES	6
2.	**CHAPTER 2 : PROJECT DESCRIPTION**	7
	2.1 GENERAL	7
	2.2 METHODOLOGIES	7
	2.2.1 MODULES NAME	7
	2.3 MODULES DESCRIPTION	8
	2.4 TECHNIQUE USED OR ALGORITHM USED	9
	2.4.1 PROPOSED SYSTEM ALGORITHM	9
	2.4.2 EXISTING SYSTEM ALGORITHM	10
3.	**CHAPTER 3 : REQUIREMENTS ENGINEERING**	11
	3.1 GENERAL	11
	3.2 HARDWARE REQUIREMENTS	11
	3.3 SOFTWARE REQUIREMENTS	12
	3.4 FUNCTIONAL REQUIREMENTS	13
	3.5 NON-FUNCTIONAL REQUIREMENTS	13

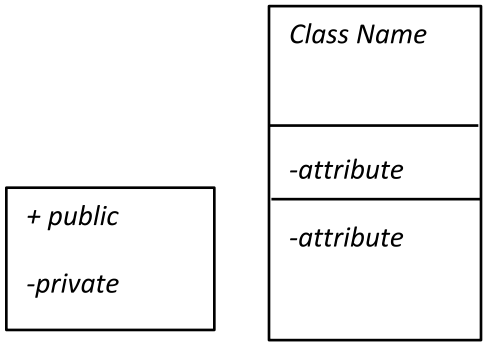
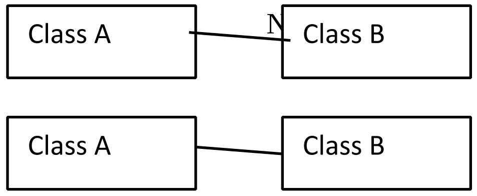
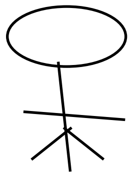
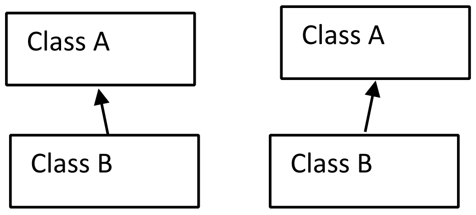
4.	**CHAPTER 4 : SYSTEM DESIGN**	14
	4.1 GENERAL	14
	4.2 UML DIAGRAMS	15
	4.2.1 USE CASE DIAGRAM	15
	4.2.2 CLASS DIAGRAM	16
	4.2.3 OBJECT DIAGRAM	17
	4.2.4 STATE DIAGRAM	18
	4.2.5 ACTIVITY DIAGRAM	19
	4.2.6 SEQUENCE DIAGRAM	21
	4.2.7 COLLABORATION DIAGRAM	22
	4.2.8 COMPONENT DIAGRAM	23
	4.2.9 DATA FLOW DIAGRAM	24
	4.2.10 DEPLOYMENT DIAGRAM	26
	4.2.11 SYSTEM ARCHITECTURE (TWO TIER)	27
5.	**CHAPTER 5 : DEVELOPMENT TOOLS**	29
	5.1 PYTHON	29
	5.2 HISTORY OF PYTHON	29
	5.3 IMPORTANCE OF PYTHON	29
	5.4 FEATURES OF PYTHON	30
	5.5 MACHINE LEARNING	31
	5.6 PYTHON LIBRARIES FOR MACHINE LEARNING	32
	5.7 FEATURES OFFERED BY PYTHON FOR ML	34
6.	**CHAPTER 6 : IMPLEMENTATION**	36
	6.1 GENERAL	36
	6.2 IMPLEMENTATION & SOURCE CODE	36
7.	**CHAPTER 7 : SNAPSHOTS**	54
	7.1 GENERAL	54
	7.2 VARIOUS SNAPSHOTS	54

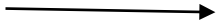
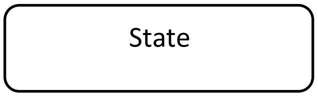
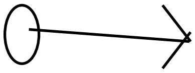
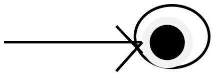
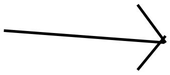
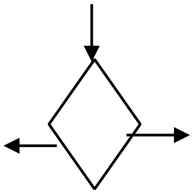
8.	**CHAPTER 8 : SOFTWARE TESTING**	59
	8.1 GENERAL	59
	8.2 DEVELOPING METHODOLOGIES	59
	8.3 TYPES OF TESTING	59
	8.3.1 UNIT TESTING	59
	8.3.2 FUNCTIONAL TEST	60
	8.3.3 SYSTEM TEST	60
	8.3.4 PERFORMANCE TEST	60
	8.3.5 INTEGRATION TESTING	60
	8.3.6 ACCEPTANCE TESTING	60
	8.4 BUILDING THE TEST PLAN	61
9.	**CHAPTER 9 : CONCLUSION**	62
	9.1 CONCLUSION	62
10.	**CHAPTER 10 : FUTURE ENHANCEMENTS**	63
	10.1 FUTURE ENHANCEMENTS & APPLICATIONS	63

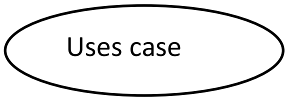
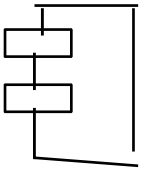
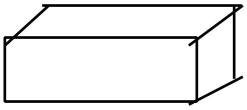
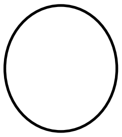
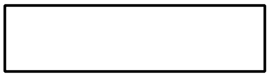
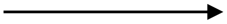
LIST OF FIGURES

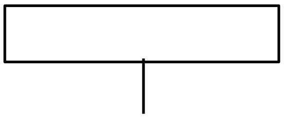
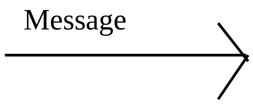
FIGURE NO.	NAME OF THE FIGURE	PAGE NO.
4.2.1	USE CASE DIAGRAM	15
4.2.2	CLASS DIAGRAM	16
4.2.3	OBJECT STATE DIAGRAM	17
4.2.4	STATE DIAGRAM	18
4.2.5	ACTIVITY DIAGRAM	19
4.2.6	SEQUENCE DIAGRAM	21
4.2.7	COLLABORATION DIAGRAM	22
4.2.8	COMPONENT DIAGRAM	23
4.2.9	DATA FLOW DIAGRAM	24
4.2.10	DEPLOYMENT DIAGRAM	26
4.2.11	SYSTEM ARCHITECTURE (TWO TIER)	27

LIST OF SYMBOLS

S. NO.	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class	 Class Name -attribute -attribute + public -private	Represents a collection of similar entities grouped together.
2.	Association		Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.
4.	Aggregation		Interaction between the system and external environment

5.	Relation (uses)	uses	Used for additional process communication.
6.	Relation (extends)	 extends	Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the processes.
9.	Initial State		Initial state of the object
10.	Final state		Final state of the object
11.	Control flow		Represents various control flow between the states.
12.	Decision box		Represents decision making process from a constraint

13.	Use case		Interact ion between the system and external environment.
14.	Component		Represents physical modules which are a collection of components.
15.	Node		Represents physical modules which are a collection of components.
16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
17.	External entity		Represents external entities such as keyboard, sensors, etc.
18.	Transition		Represents communication that occurs between processes.

19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message		Represents the message exchanged.

LIST OF ABBREVIATIONS

S. NO.	ABBREVATION	EXPANSION
1.	ML	Machine Learning
2.	SVM	Support Vector Machine
3.	KNN	K-Nearest Neighbor
4.	ANN	Artificial Neural Networks
5.	AI	Artificial Intelligence
6.	DNN	Deep Neural Networks
7.	MTDNN	Multitask Deep Neural Networks
8.	MNIST	Modified National Institute of Standards and Technology
9	CNN	Convolutional Neural Networks
10.	GAN	Generative Adversarial Networks
11.	NLP	Natural Language Processing
12.	VGG	Visual Geometry Group
13.	ADAM	Adaptive Moment Estimation
14.	ReLU	Rectified Linear Unit

ABSTRACT

In the realm of cutting-edge electric vehicle (EV) technology, "Electra Synergy" presents a pioneering methodology. This project introduces a revolutionary classification technique for EV charging session types and optimal charging strategy optimization, leveraging the K-Nearest Neighbours (KNN) algorithm and the Random Forest algorithm. This ensemble technique takes a unique approach diverging from conventional methods. It combines the simplicity, implementation ease, and adept handling of multi-class classification tasks offered by KNN with the robustness and accuracy of the Random Forest algorithm. By combining these algorithms, the project aims to enhance battery longevity, minimize charging expenditures, and tailor charging strategies to user preferences and operational requirements. The integration of real-time data and State-of-Charge (SoC) data, coupled with feedback loops, augments the model's adaptability, ensuring a refined charging regimen over time. This innovative approach promises to usher in a new era of sustainable and cost-effective EV charging practices, marking a paradigm shift in the field.

Furthermore, "Electra Synergy" harnesses a robust dataset comprising over 5000 elements, significantly surpassing previous endeavors. This larger dataset contributes to heightened predictive accuracy, marking a substantial advancement in comparison to existing projects. A pivotal aspect of our initiative lies in harnessing the power of machine learning models and the integration with web applications. This integration ensures an effortless and interactive user experience, allowing users to input diverse factors influencing EV charging optimization easily. This user-friendly interface enhances accessibility and provides clear and transparent charging recommendations, facilitating a rich user experience. By providing accurate real-world range information for EVs, "Electra Synergy" aims to empower users with reliable data for making well-informed decisions, countering manufacturers' tendency to inflate claimed ranges. This ensures that EV owners have reliable information for planning their journeys and fosters trust in the EV industry.

CHAPTER 1
INTRODUCTION

1.1 GENERAL

Electric Vehicle (EV) charging optimization faces challenges such as varying battery types, charging infrastructure limitations, and user preferences, requiring sophisticated solutionsfor efficient charging. The inadequacy of the older methods prompted our foray into the development of a sophisticated predictive model, poised to overcome the inefficiencies of the past. Transparency in EV charging, especially regarding real-world range information, has been lacking due to manufacturers' tendency to provide inflated claimed ranges, creating a need for accurate and reliable data. Previous models struggled to achieve the desired precision in predicting car prices, often resulting in suboptimal decisionmaking for consumers. Recognizing this gap, our project sets out to redefine precision in automotive valuation, offering a solution that goes beyond conventional limitations.

This project not only optimizes EV charging practices but also acknowledges the global imperative for sustainability. By integrating considerations for eco-friendly vehicles, the project promotes informed choices that contribute to a reduced carbon footprint and a more sustainable automotive future. Through encouraging the adoption of environmentally conscious vehicles, "Electra Synergy" aims to play a role in mitigating the environmental impact of the EV automotive industry. This project envisions a future where automotive valuation through predictive modeling is synonymous with accuracy, transparency, and fairness. We aim to revolutionize the field by setting new standards and providing solutions that address historical inefficiencies. By leveraging the KNN algorithm, our platform offers accurate and transparent charging recommendations, prioritizing user preferences, operational requirements, and environmental sustainability. "Electra Synergy" is dedicated to advancing the automotive industry's predictive modeling practices, ensuring optimal battery health, minimized charging costs, and a greener future.

1.2 SCOPE OF THE PROJECT

The scope involves leveraging machine learning to analyze charging session data, develop predictive models for optimal battery charging in electric vehicles. These strategies are then rigorously tested, refined, and integrated into electric vehicle charging systems for practical deployment. Continuous monitoring and adaptation are integral, allowing for ongoing improvements by updating models with new data, thereby enhancing the overall efficiency of the charging process.

1.3 OBJECTIVE

The objective of the 'Electra Synergy' project is to develop a user-friendly web application that leverages an ensemble approach, combining the use of K-Nearest Neighbours (KNN) and Random Forest algorithms. The application allows users to input EV parameters, such as battery capacity, charge duration, Charger Wattage, Driving Speed, etc., and receive accurate predictions of the vehicle range in the real world. This project aims to enhance battery health, minimize charging costs, and cater to user preferences or operational requirements. By leveraging real-time data and State-of-Charge (SoC) data integration, coupled with feedback loops, the model's adaptability is improved, ensuring enhanced charging recommendations over time. Utilizing a dataset collected from reputable sources, the goal is to provide individuals with a simple and effective tool for estimating EV range based on machine learning insights, enhancing the user experience in the process. The project seeks to revolutionize EV charging practices, promoting sustainability and cost-effectiveness in the EV industry.

1.4 EXISTING SYSTEM

The novelty of the proposed project consists in accurately forecasting the class of EV charge power profiles in terms of a reduced number of features (duration) which is a key parameter for EV charging stations and DSOs (Distribution System Operator). This is a change of the perspective in the traditional load power curve methodology which usually relies on time-series forecasting methods. The proposed model is validated on a real-world dataset containing information about charging events that occurred in more than 100 EV charging stations around the UK. Thanks to its simplicity and speed the proposed SVM algorithm, and thanks to the possibility to generalize the model, it could be implemented and fully work locally on any type of charging towers.

1.4.1 EXISTING SYSTEM DISADVANTAGES:

- It does not execute very well when the data set has more sound i.e. target classes are overlapping.
- In cases where the number of properties for each data point outstrips the number of training data specimens, the model will underperform.
- It doesn't perform well when we have large data set because the required training time is higher.

1.5 LITERATURE SURVEY

Title: Charging optimization for Li-ion battery in electric vehicles: A review

Author: C. Chen, Z. Wei, and A. C. Knoll

Year: 2022.

Description: Battery electric vehicles (BEVs) are advocated due to their environmental benign characteristic. However, the long charging time and the degradation caused by fast charging impede their further popularization. Extensive research has been carried out to optimize the charging process, such as minimizing charging time and aging, of lithium-ion batteries (LIBs). Motivated by this, a comprehensive review of existing charging optimization (ChgOp) techniques is provided in this article. First, the operation and models for LIBs are explained. Then, unexpected side effects, especially for the aging mechanism (AM) of LIB associated with unregulated fast charging, are scrutinized. This provides a solid theoretical foundation and forms the optimization problem. Following this endeavor, the general framework with critical concerns for ChgOp system design is overviewed. Within this horizon, the state-of-the-art ChgOp techniques, clustered into open- and close-loop categories, are reviewed systematically with their respective merits and shortcomings discussed. Finally, the development of an emerging charging control protocol with both real-time affordability and degradation consciousness is further discussed as an open outlook.

Title: Pre diction of EV charging behavior using machine learning.

Author: S. Shahriar, A. R. Al-Ali, A. H. Osman, S. Dhou, and M. Nijim

Year: 2021.

Description: As a key pillar of smart transportation in smart city applications, electric vehicles (EVs) are becoming increasingly popular for their contribution in reducing greenhouse gas emissions. One of the key challenges, however, is the strain on power grid infrastructure that comes with large-scale EV deployment. The solution to this lies in utilization of smart scheduling algorithms to manage the growing public charging demand. Using data-driven tools and machine learning algorithms to learn the EV charging behavior can improve scheduling algorithms. Researchers have focused on using historical charging data for predictions of behavior such as departure time and energy needs. However, variables such as weather, traffic, and nearby events, which have been neglected to a large extent, can perhaps add meaningful representations, and provide better predictions. Therefore, in this paper we propose the usage of historical charging data in conjunction with weather, traffic, and events data to predict EV session duration and energy consumption using popular machine learning algorithms including random forest, SVM, XGBoost and deep neural networks. The best predictive performance is achieved by an ensemble learning model, with SMAPE scores of 9.9% and 11.6% for session duration and energy consumptions, respectively, which improves upon the existing works in the literature. In both predictions, we demonstrate a significant improvement compared to previous work on the same dataset and we highlight the importance of traffic and weather information for charging behavior predictions.

Title: About Optimized Battery Charging

Author: W. Shen

Year: 2022.

Description: A battery's lifespan is related to its chemical age, which is more than just the length of time since the battery was assembled. A battery's chemical age results from a complex combination of several factors, including temperature history and charging pattern. All rechargeable batteries are consumable components that become less effective as they chemically age. As lithium-ion batteries chemically age, the amount of charge they can hold diminishes, resulting in reduced battery life and reduced peak performance. Find out more about iPhone battery and performance and how to maximise battery performance and lifespan

Title: Towards a smarter battery management system: A critical review on optimal charging methods of lithium ion batteries

Author: Q. Lin, J. Wang, R. Xiong,

Year: 2019

Description: Energy storage system (ESS) technology is still the logjam for the electric vehicle (EV) industry. Lithium-ion (Li-ion) batteries have attracted considerable attention in the EV industry owing to their high energy density, lifespan, nominal voltage, power density, and cost. In EVs, a smart battery management system (BMS) is one of the essential components; it not only measures the states of battery accurately, but also ensures safe operation and prolongs the battery life. The accurate estimation of the state of charge (SOC) of a Li-ion battery is a very challenging task because the Li-ion battery is a highly time variant, non-linear, and complex electrochemical system. This paper explains the workings of a Li-ion battery, provides the main features of a smart BMS, and comprehensively reviews its SOC estimation methods. These SOC estimation methods have been classified into four main categories depending on their nature. A critical explanation, including their merits, limitations, and their estimation errors from other studies, is provided. Some recommendations depending on the development of technology are suggested to improve the online estimation.

1.6 PROPOSED SYSTEM

In this system, we implement an advanced EV charging optimization system using an ensemble approach with the K-Nearest Neighbors (KNN) and Random Forest algorithms. Our system offers a seamless user experience with features such as real-time data integration and feedback loops. Beyond its standalone capabilities, Electra Synergy features web integration, allowing users to access the optimization service through a user-friendly web interface. This integration enhances accessibility, enabling users to input charging parameters effortlessly and receive optimized strategies in real-time. Whether through direct data integration or interactive manual entry, the system's web integration ensures a versatile and dynamic platform for efficient EV charging management.

1.6.1 PROPOSED SYSTEM ADVANTAGES:

- Robust to Overfitting : The ensemble nature of Random Forests helps mitigate overfitting, making them more robust and less sensitive to noise in the data.
- KNN stores the entire training dataset and directly uses it during inference.
- As new data is added, KNN can easily adapt without needing to retrain the model.
- KNN doesn't make any assumptions about the underlying data distribution, making it suitable for diverse datasets.
- Feature Importance: Random Forests can provide insights into feature importance, helping to identify which variables are most influential in making predictions.
- Handles Missing Values: Random Forests can handle missing values in the dataset, maintaining accuracy even when data is incomplete.
- High accuracy rate (over 95 %)
- Enhanced Precision and Accuracy
- User-Friendly Web Integration
- Adaptability to Diverse Data Sources
- Real-Time Predictions
- Transparency and Trust

CHAPTER 2

PROJECT DESCRIPTION

2.1 GENERAL

Using machine learning to optimize battery charging in electric vehicles involves a comprehensive approach. It begins with collecting diverse data, encompassing charging sessions, battery specifics, driving habits, and environmental conditions. This data serves as the foundation for creating machine learning models. These models are designed to forecast the most efficient charging parameters based on historical data and real-time inputs. Once developed, these models facilitate the implementation of optimized strategies, such as scheduling charges during off-peak hours or adjusting rates according to grid demand, ultimately aiming to extend battery life and ensure cost-effectiveness. These strategies are then rigorously tested, refined, and integrated into electric vehicle charging systems for practical deployment. Continuous monitoring and adaptation are integral, allowing for ongoing improvements by updating models with new data, thereby enhancing the overall efficiency of the charging process.

2.2 METHODOLOGIES

2.2.1 MODULE NAMES

- Data Collection
- Data Analysis
- Data Pre-Processing
- Model Training & Testing
- Model Performance Evalutaion
- Web Interface Development
- Flask Integration
- Testing & Validation
- Result
- Deployment

2.3 MODULE DESCRIPTION

Data Collection : Data collection in the context of optimizing battery charging for electric vehicles involves gathering diverse information relevant to the charging process, vehicle usage, and environmental factors. We focused on gathering information such as start plugin hour, end plugout hour, and duration of charging sessions, ensuring a comprehensive dataset for accurate modeling.

Data Analysis : Data analysis, within the context of optimizing battery charging for electric vehicles, involves examining the collected data to derive meaningful insights and patterns that can inform the development of charging optimization strategies.

Data Pre-Processing : It typically involves cleaning, transforming, and organizing raw data to make it suitable for analysis. This may include tasks such as handling missing values, encoding categorical variables, scaling numerical features, and removing outliers to prepare the data for machine learning algorithms., it is pivotal to identify and correctly handle the missing values, failing to do this, you might draw inaccurate and faulty conclusions and inferences from the data.

Model Training & Testing : The training process is iterative and involves experimentation and fine-tuning to develop a model that accurately predicts optimal charging parameters for electric vehicles based on historical data. In this we use 90% of data for training and remaining 10 % of data for testing purpose.

Model Performance Evaluation : It involves assessing how well a machine learning model performs its task. This includes checking its accuracy, precision, or other specific measures relevant to the problem it's solving, like predicting optimal charging for electric vehicles. It's like testing the model's abilities with new data it hasn't seen before to ensure it doesn't just memorize previous information but can make accurate predictions.

Web Interface Development : Our web interface development prioritizes an intuitive platform using HTML5, CSS3, and JavaScript (ES6) for seamless input of EV details. The design, featuring a clean layout and responsive elements, enhances accessibility, while interactive features provide a dynamic user experience. Optimized for Google Chrome, the interface seamlessly integrates with the Flask backend, ensuring efficient communication with the machine learning model for accurate EV charging optimization.

Flask Integration : In the Flask integration module, we leverage the Flask framework to seamlessly connect the web interface with the machine learning model. The Flask application, hosted on the backend, handles HTTP requests from the frontend, facilitating a smooth flow of data between the user interface and the model. Through the Flask routes, user inputs are processed, and predictions from the Regressor model are returned to the interface in real-time. This integration ensures a robust and responsive connection, allowing users to experience efficient and accurate EV charging Optimization strategies within the web application.

Testing & Validation : The testing and validation module rigorously evaluates the machine learning model, ensuring accuracy and reliability in predicting the results. Utilizing diverse validation techniques, this phase verifies the model's performance under different scenarios, providing confidence in its predictive capabilities. Thorough testing ensures the robustness of our system, enhancing its reliability for real-world deployment.

Result : In essence, "results" encompass the outcomes generated by applying machine learning models or data analysis techniques, aiding in decision-making processes, insights generation, and guiding further steps toward achieving the objectives set for optimizing battery charging in electric vehicles or any other relevant task.

Deployment : The deployment phase finalizes the project, ensuring the entire system operates seamlessly in a production environment. This stage encompasses the integration of all modules, making the system accessible to end-users. By optimizing performance and resolving any potential deployment issues, we ensure a reliable and user-ready application for EV charging optimization.

2.4 TECHNIQUE USED OR ALGORITHM USED

2.4.1 PROPOSED SYSTEM ALGORITHM

K-Nearest Neighbor & Random Forest Algorithms (Ensemble Approach) :

Single machine learning classifier approach that has been used in all previous researches was also tested in this research. The whole data set collected in this research has been split into training (90%) and testing (10%) subsets and an Ensemble Regressor Model is built using both the KNN (K-Nearest Neighbor) algorithm and the Random Forest algorithm. The ensemble approach combining K-Nearest Neighbors (KNN) and Random Forest algorithms was developed as a robust method for improving classification accuracy and generalization in machine learning tasks. Random Forest, pioneered by Leo

Breiman and Adele Cutler, operates by constructing multiple decision trees during training and outputting the mode of the classes as the prediction. This approach reduces overfitting and increases the overall accuracy by averaging the predictions of individual trees. On the other hand, KNN, conceptualized by Evelyn Fix and Joseph Hodges, relies on the similarity of instances to make predictions, where the class of an unseen instance is determined by the majority class among its K nearest neighbors. Combining these algorithms in an ensemble model enhances their strengths, providing a more robust and accurate prediction model capable of handling complex datasets and improving performance across various machine learning tasks.

The model built undergoes training on a split dataset, using a majority for learning patterns and a remaining portion for rigorous testing, ensuring its ability to generalize to new data. Remarkably, the Ensemble approach achieves an impressive 98% accuracy, attributed to its ensemble nature, which minimizes overfitting and enhances overall robustness. The decision to choose the ensemble approach, combining K-Nearest Neighbors (KNN) and Random Forest algorithms, was intentional and driven by its suitability for handling diverse factors influencing EV charging optimization. Also the ability to handle diverse features in our dataset and its excellent generalization capabilities. This deliberate choice has not only contributed to the model's high accuracy but also ensures its reliability in providing precise predictions for a wide range of EV configurations in EV charging optimization system.

2.4.2 EXISTING SYSTEM ALGORITHM

SVM (Support Vector Machines) :

It is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates different classes in the feature space. The hyperplane is chosen to maximize the margin, which is the distance between the hyperplane and the nearest data point of each class, making the classification robust. SVM can handle both linear and non-linear data by using different kernel functions to transform the data into higher-dimensional spaces where a hyperplane can be more easily defined. It aims to find the optimal hyperplane that separates the classes with the largest margin, which helps in generalizing well to unseen data. The algorithm relies on a subset of training data points called support vectors, which are the data points closest to the hyperplane and are crucial in defining its orientation.

CHAPTER 3

REQUIREMENTS ENGINEERING

3.1 GENERAL

These are the requirements for doing the project. Without using these tools and software's we can't do the project. So we have two requirements to do the project.

They are-

1. Hardware Requirements.
2. Software Requirements.

3.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

● Processor:	Intel Core i3-8100 (Quad Core CPU or Higher)
● RAM:	8 GB DDR4 2666Mt/s or higher
● Storage:	256 GB SATA HDD at 7200 RPM (preferably PCIe Gen3 SSD for faster read-write speeds)
● GPU:	NVIDIA GeForce GTX 1000 series CUDA-enabled GPU for faster model training (Optional but recommended for large datasets)

3.3 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

- Operating System: Windows 7/ MacOS X 10.13 or higher/ Ubuntu 18.04 or higher
- IDE: Microsoft Visual Studio /Anaconda/Jupyter/ Spyder (IDE of choice)
- Python Environment: Python 3 (version 3.9.18)
- Libraries for Data Processing: Pandas (version 2.2.1) , Matplotlib (version 3.8.0) NumPy (version 1.26.4) , Seaborn (version 0.12.2) Pickle (version 4)
- Machine Learning Libraries: Scikit-learn (version 1.3.0)
- Database: Local Storage (No Separate Database)
- Framework: Flask (version 2.2.5)
- Frontend (Web Application): HTML5, CSS3, JavaScript (ES6) (Preferably Google Chrome Browser)

3.4 FUNCTIONAL REQUIREMENTS

In this project, a functional requirement introduces a combination of regressor methods for EV charging optimization, focusing specifically on the implementation of the KNN (K-Nearest Neighbor) and the Random Forest Algorithm in an ensemble approach. Our model utilizes a diverse set of parameters such as EV specifications, features, and historical data.The Regressor is then applied to these parameters, leveraging its power to capture complex relationships within the dataset. Through this iterative refinement process, the model achieves high accuracy in EV charging optimization, showcasing the effectiveness of the ensemble model in providing nuanced and precise estimations based on a comprehensive set of features.

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute-based deduplication systems by resorting to the hybrid cloud architecture.

3.5 NON-FUNCTIONAL REQUIREMENTS

The major non-functional Requirements of the system are as follows

Usability : The system is designed with a completely automated process hence there is no or less user intervention.

Reliability : The system is more reliable because of the qualities that are inherited from the chosen platform python. The code built by using python is more reliable.

Performance : This system is developing in the high level languages and using the advanced front-end and back-end technologies it will give response to the end user on the client system within very less time.

Supportability : The system is designed to be cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is having python3, built into the system.

Implementation : The system is implemented in a web environment using flask framework. Flask handles communication through HTTP request-response cycles. When a user interacts with the web interface, Flask processes the input through predefined routes, triggering HTTP requests. These requests are then handled by Flask, facilitating communication with the machine learning model, and the corresponding HTTP responses deliver the predicted car prices back to the user interface in real-time.

CHAPTER 4

DESIGN ENGINEERING

4.1 GENERAL

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of the project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished products.

4.2 UML DIAGRAMS

4.2.1 USE CASE DIAGRAM

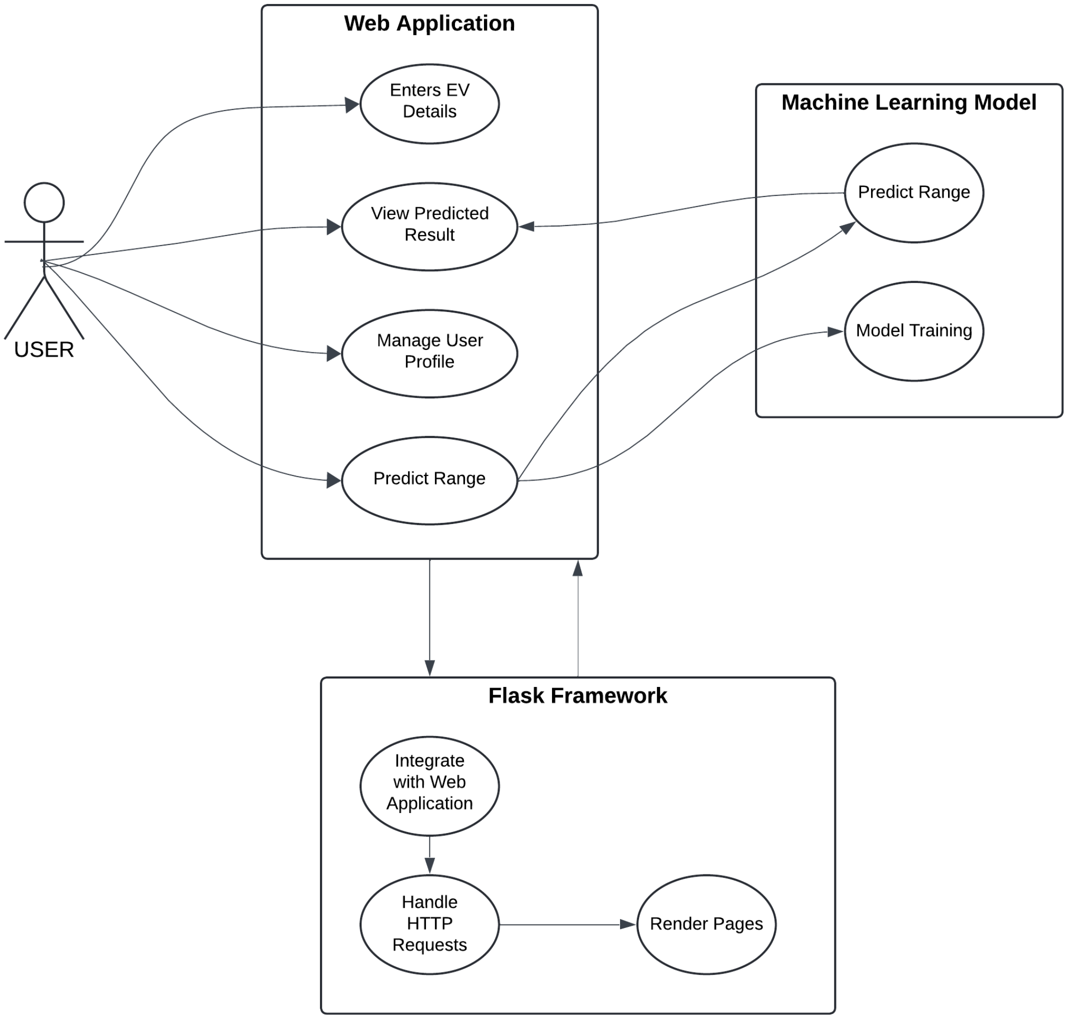


Fig 4.2.1 (Use Case Diagram for EV charging optimization)

EXPLANATION:

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. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

4.2.2 CLASS DIAGRAM

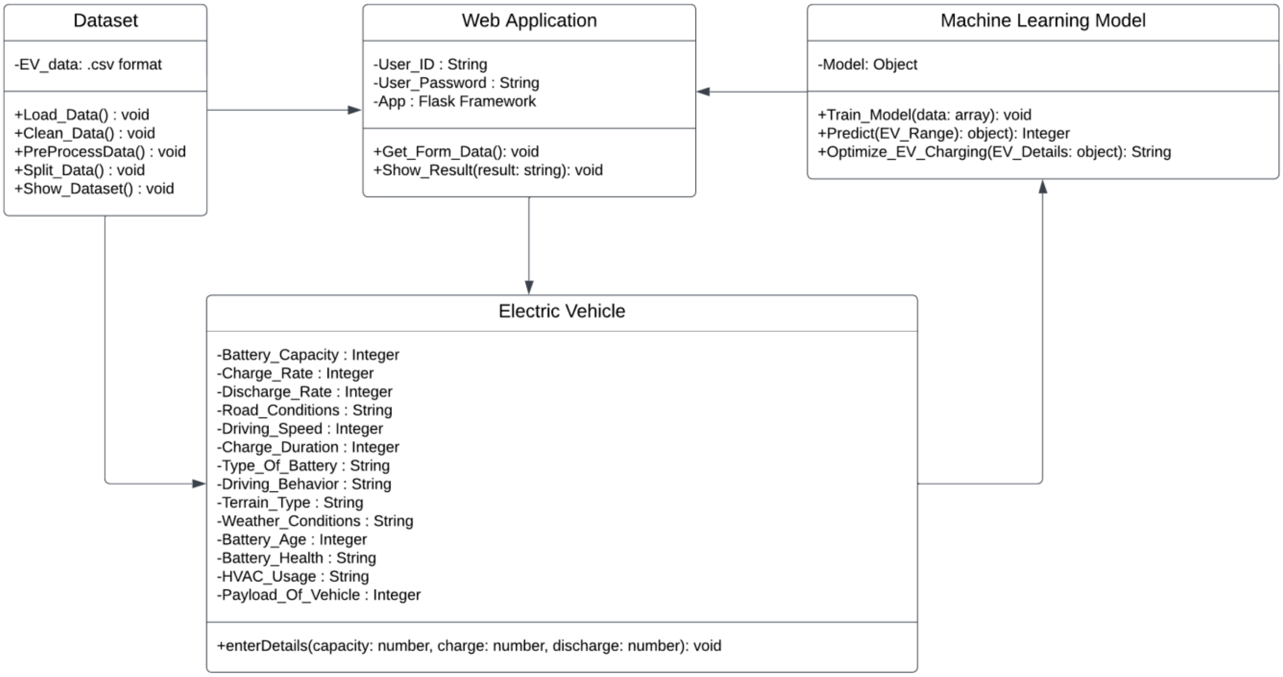

Dataset
-EV_data: .csv format
+Load_Data() : void
+Clean_Data() : void
+PreProcessData() : void
+Split_Data() : void
+Show_Dataset() : void
Web Application
-User_ID : String
-User_Password : String
-App : Flask Framework
+Get_Form_Data(): void
+Show_Result(result: string): void
Machine Learning Model
-Model: Object
+Train_Model(data: array): void
+Predict(EV_Range): object): Integer
+Optimize_EV_Charging(EV_Details: object): String
Electric Vehicle
-Battery_Capacity : Integer
-Charge_Rate : Integer
-Discharge_Rate : Integer
-Road_Conditions : String
-Driving_Speed : Integer
-Charge_Duration : Integer
-Type_Of_Battery : String
-Driving_Behavior : String
-Terrain_Type : String
-Weather_Conditions : String
-Battery_Age : Integer
-Battery_Health : String
-HVAC_Usage : String
-Payload_Of_Vehicle : Integer
+enterDetails(capacity: number, charge: number, discharge: number): void

Fig 4.2.2 (Class Diagram for EV charging optimization)

EXPLANATION:

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

4.2.3 OBJECT DIAGRAM

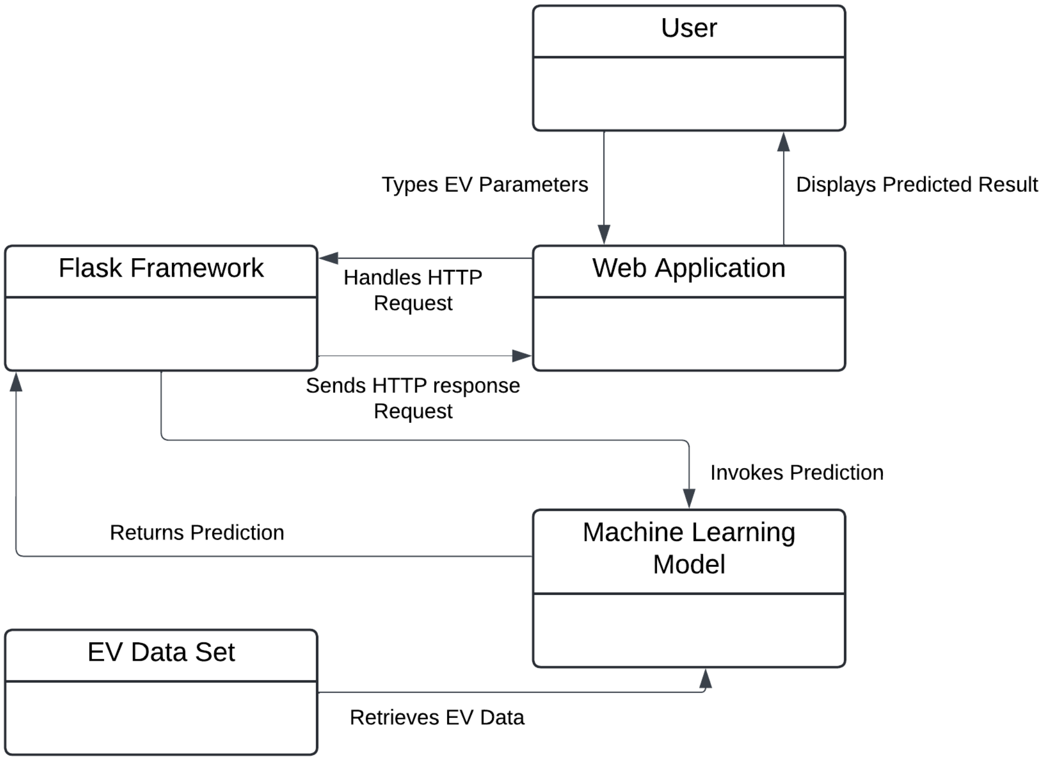


Fig 4.2.3 (Object Diagram for EV charging optimization)

EXPLANATION:

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

4.2.4 STATE DIAGRAM

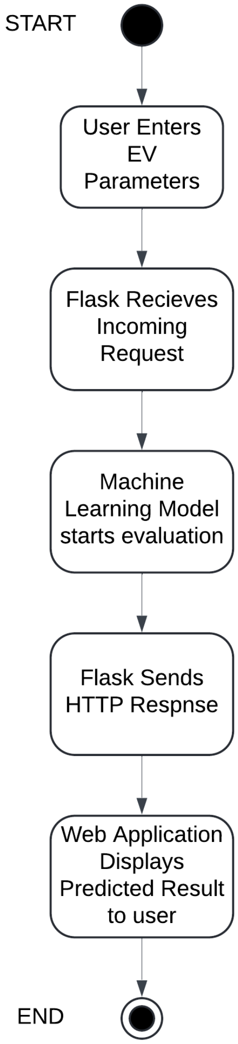

START
User Enters EV Parameters
Flask Recieves Incoming Request
Machine Learning Model starts evaluation
Flask Sends HTTP Respnse
Web Application Displays Predicted Result to user
END

Fig 4.2.4 (State Chart Diagram for EV charging optimization)

EXPLANATION:

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. UML, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. UML activity diagrams could potentially model the internal logic of a complex operation. In many ways UML activity diagrams are the object-oriented equivalent of flow charts and data flow diagrams (DFDs) from structural development.

4.2.5 ACTIVITY DIAGRAM

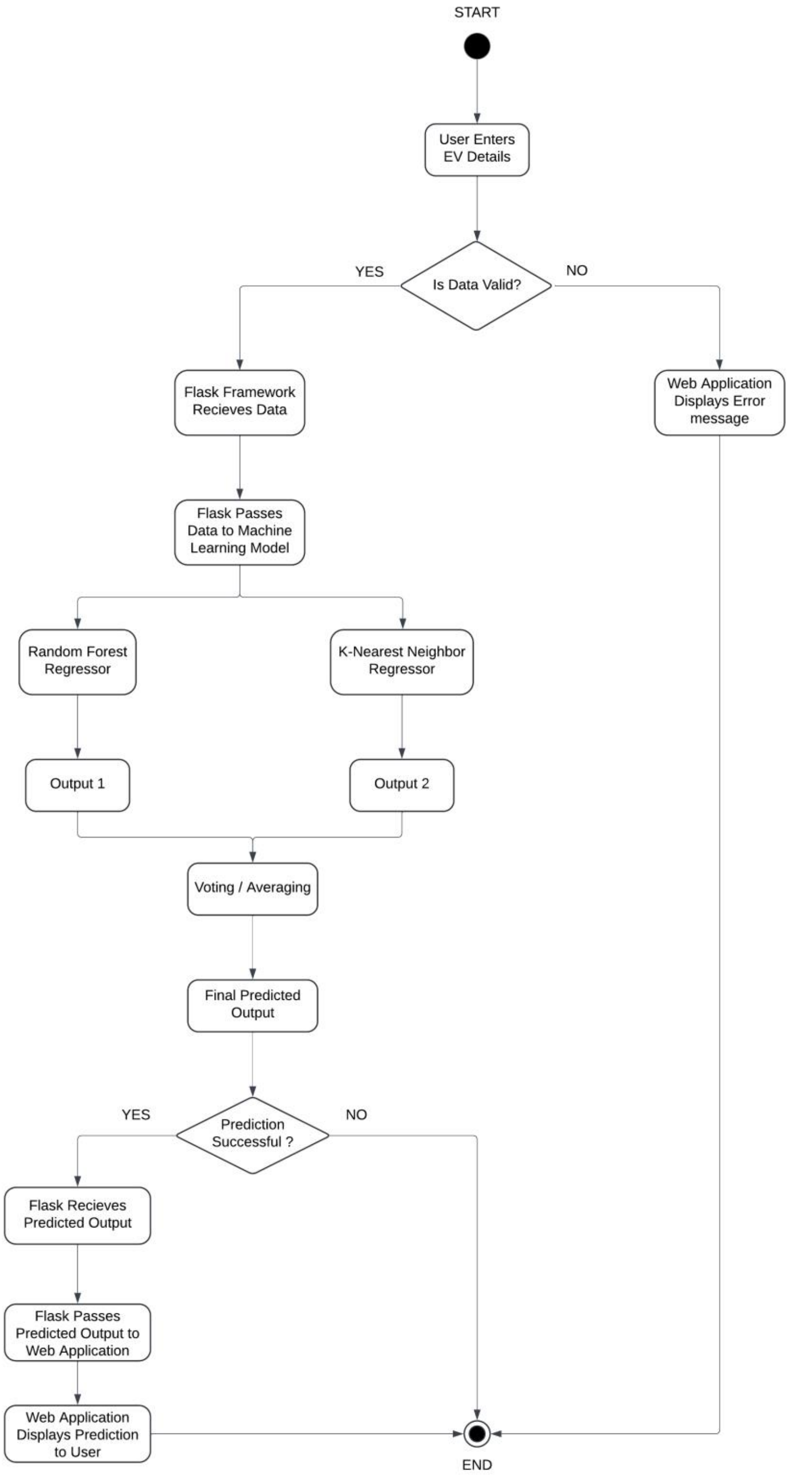

START
User Enters EV Details
Is Data Valid?
YES
NO
Flask Framework Recieves Data
Web Application Displays Error message
Flask Passes Data to Machine Learning Model
Random Forest Regressor
K-Nearest Neighbor Regressor
Output 1
Output 2
Voting / Averaging
Final Predicted Output
Prediction Successful ?
YES
NO
Flask Recieves Predicted Output
Flask Passes Predicted Output to Web Application
Web Application Displays Prediction to User
END

Fig 4.2.5 (Activity Diagram for EV charging optimization)

EXPLANATION:

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.

4.2.6 SEQUENCE DIAGRAM

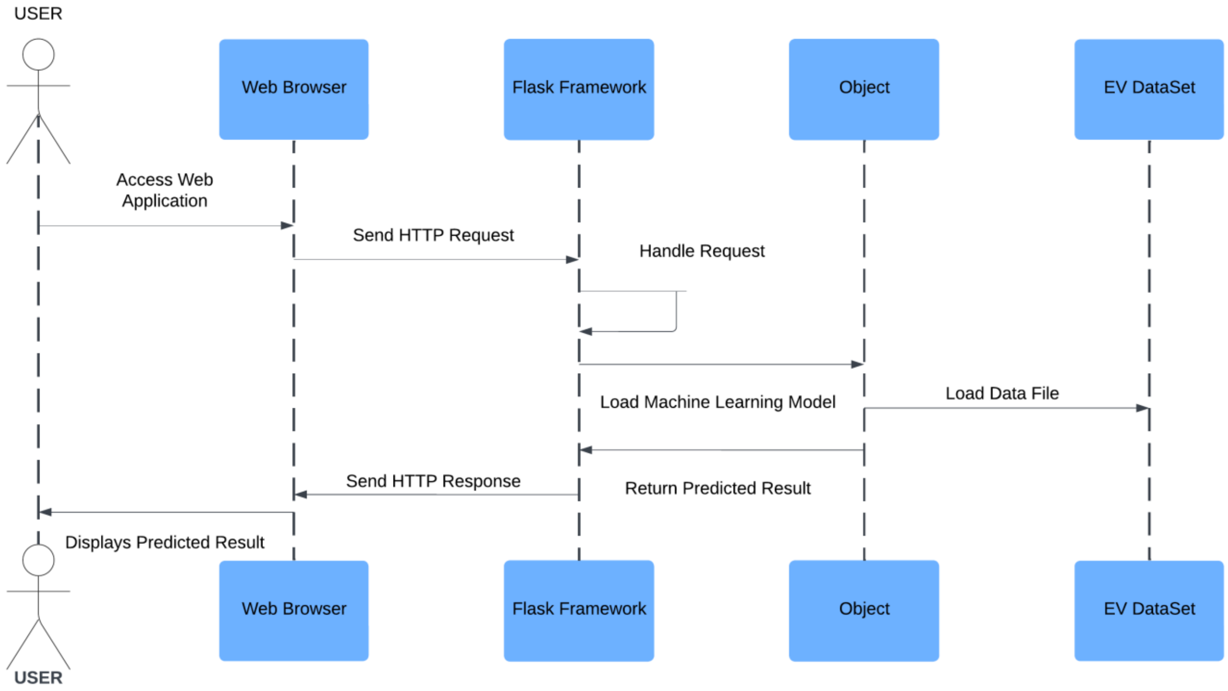


Fig 4.2.6 (Sequence Diagram for EV charging optimization)

EXPLANATION:

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 object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

4.2.7 COLLABORATION DIAGRAM

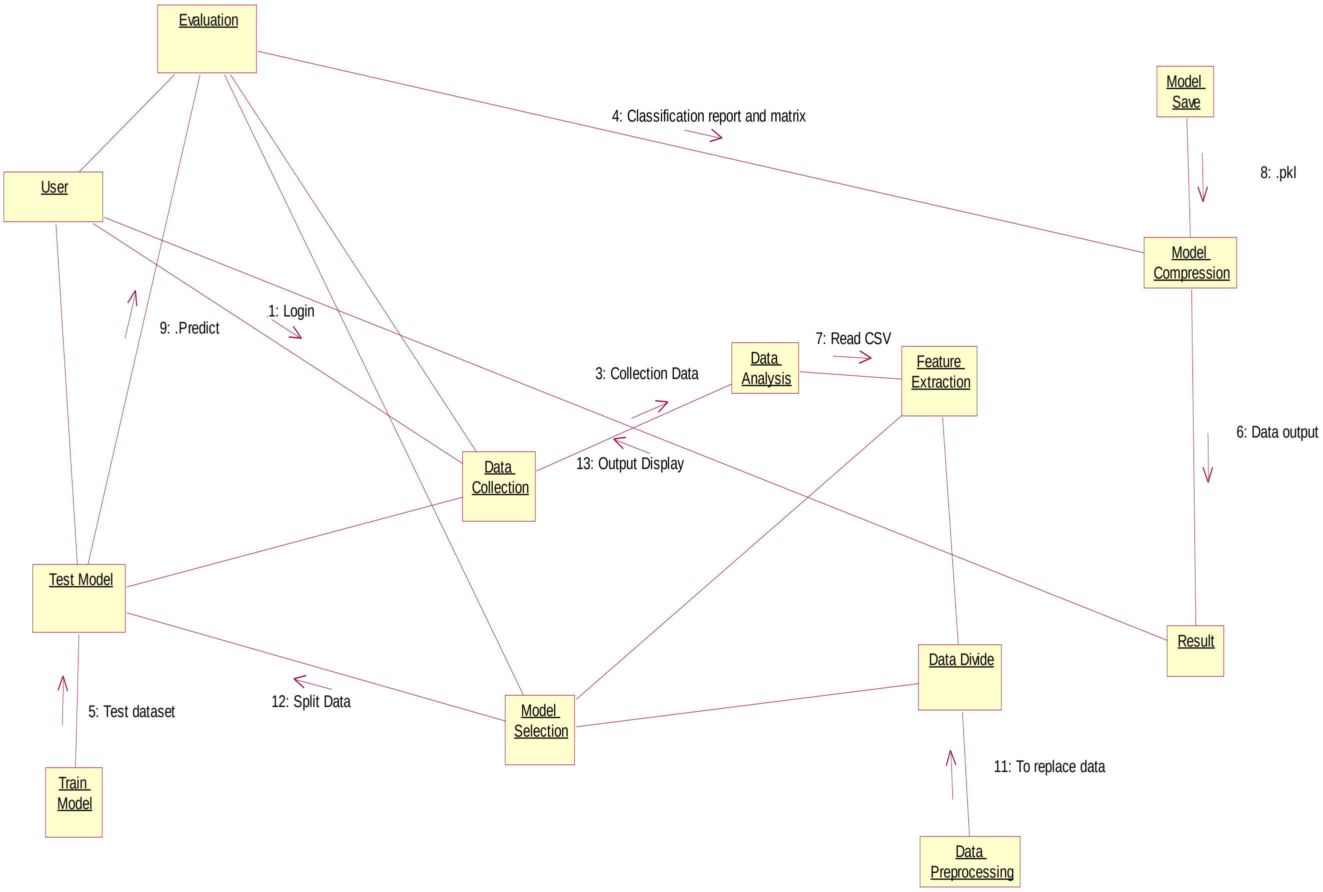


Fig 4.2.7 (Collaboration Diagram for EV chaging optimization)

EXPLANATION:

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

4.2.8 COMPONENT DIAGRAM

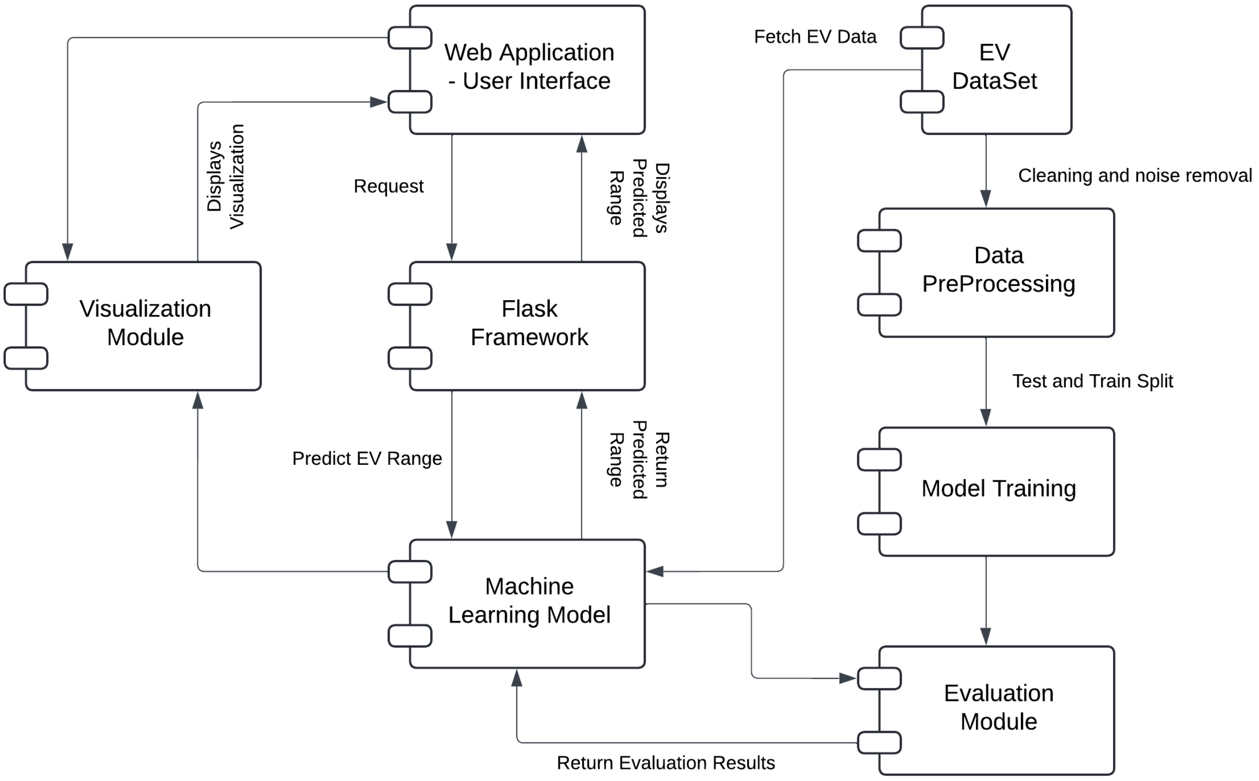


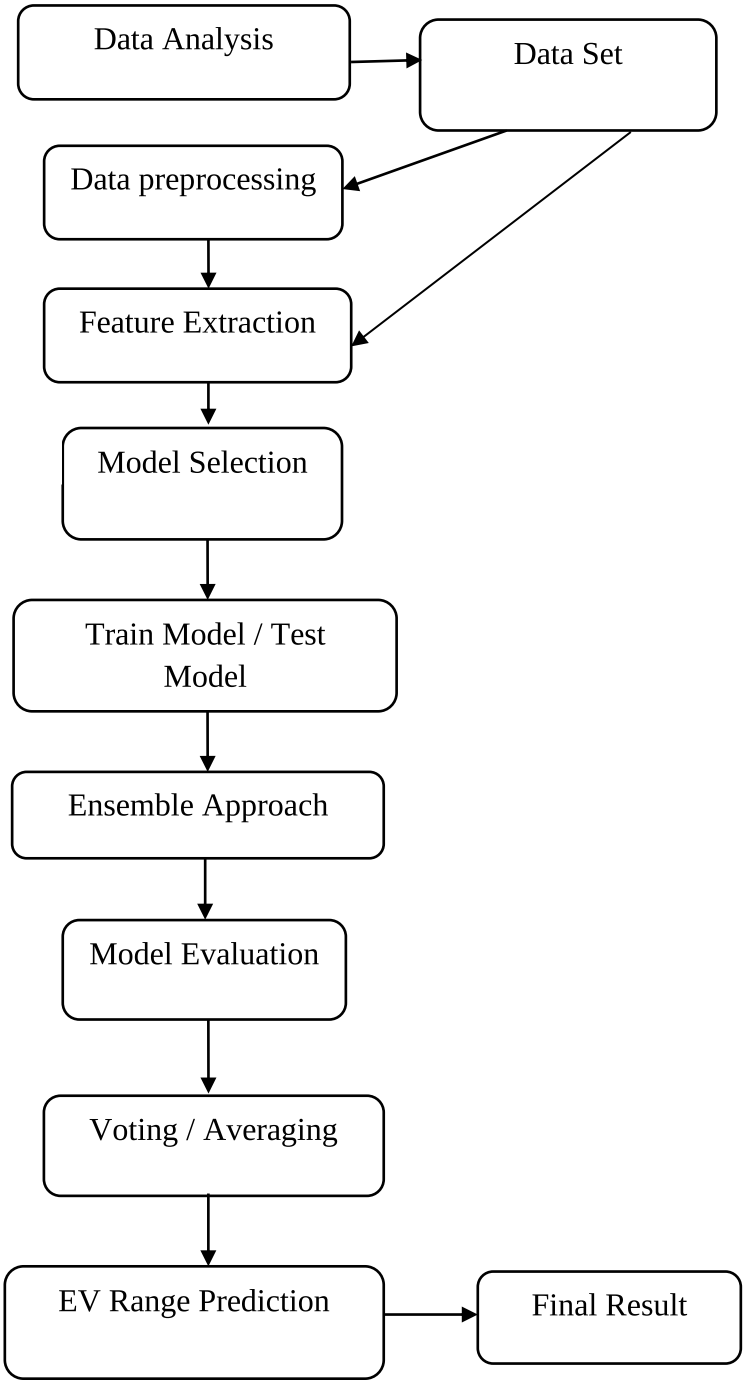
Fig 4.2.8 (Component Diagram for EV charging optmization)

EXPLANATION:

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

4.2.9 DATA FLOW DIAGRAM

Level - 0 :



Level – 1 :

Fig 4.2.9 (Data Flow Diagram for EV charging optimization)

EXPLANATION:

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

4.2.10 DEPLOYMENT DIAGRAM

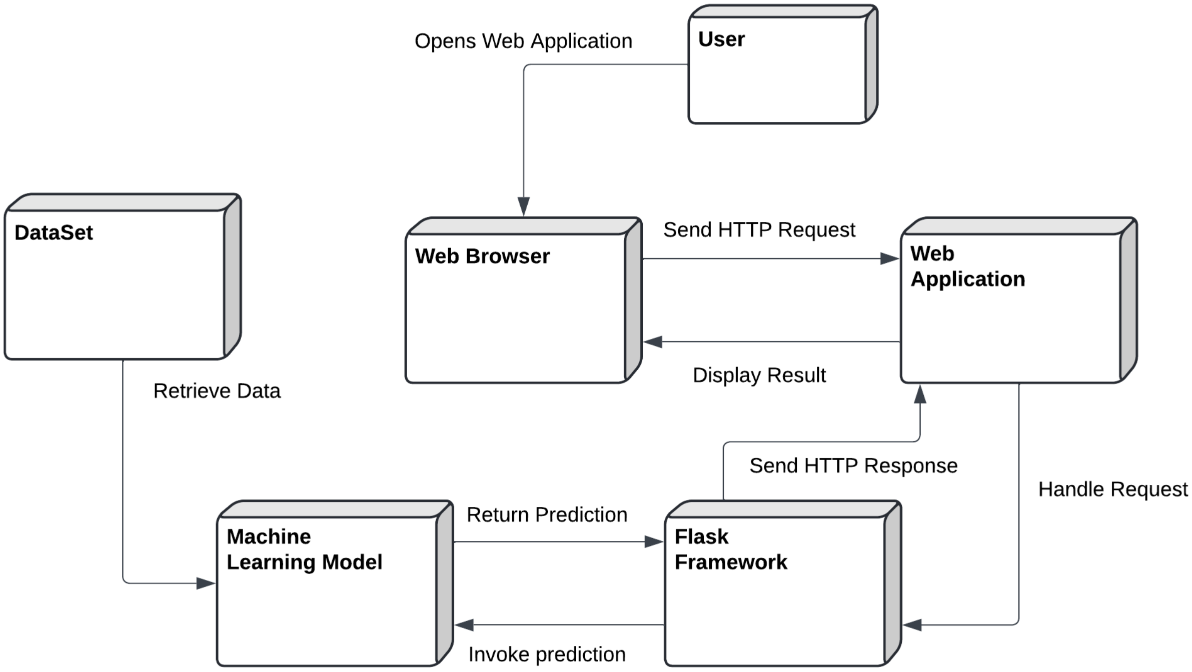


Fig 4.2.10 (Deployment Diagram for EV charging optimization)

EXPLANATION:

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

4.2.11 SYSTEM ARCHITECTURE (FRONT END & BACK END)

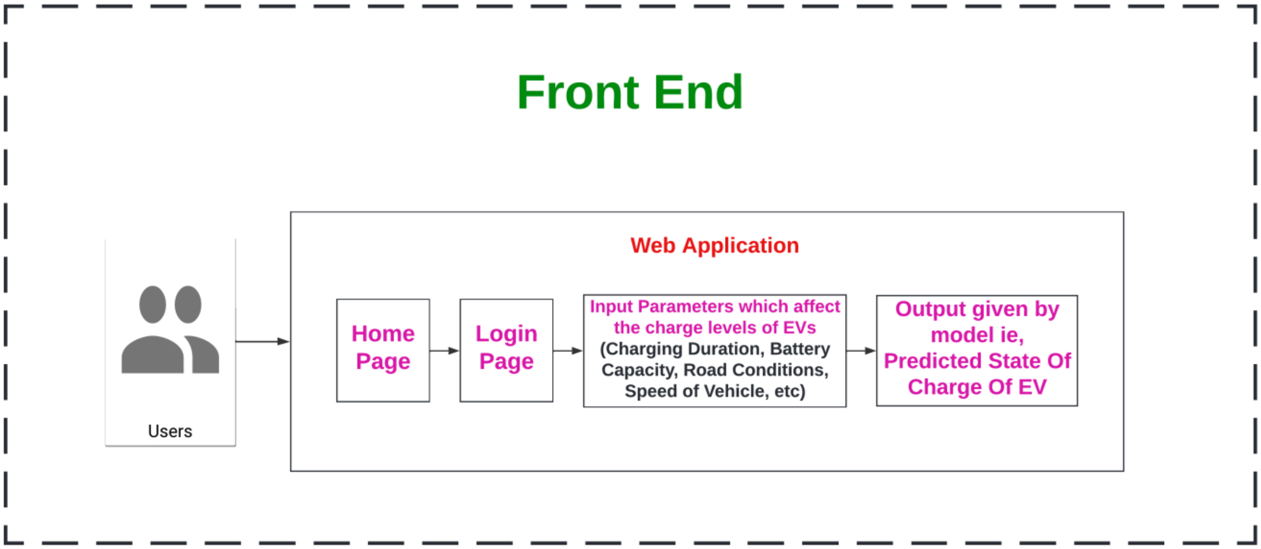

Front End
Users
Web Application
Home Page
Login Page
Input Parameters which affect the charge levels of EVs
(Charging Duration, Battery Capacity, Road Conditions, Speed of Vehicle, etc)
Output given by model ie, Predicted State Of Charge Of EV

Fig 4.2.11 (System Architecture Front End Diagram for EV Charging Optimization)

EXPLANATION :

The Electra Synergy project follows a two-tier architecture consisting of a front-end and a back-end. The front-end, developed using HTML, CSS, and JavaScript, provides a user-friendly interface for users to input EV details and view predictions.

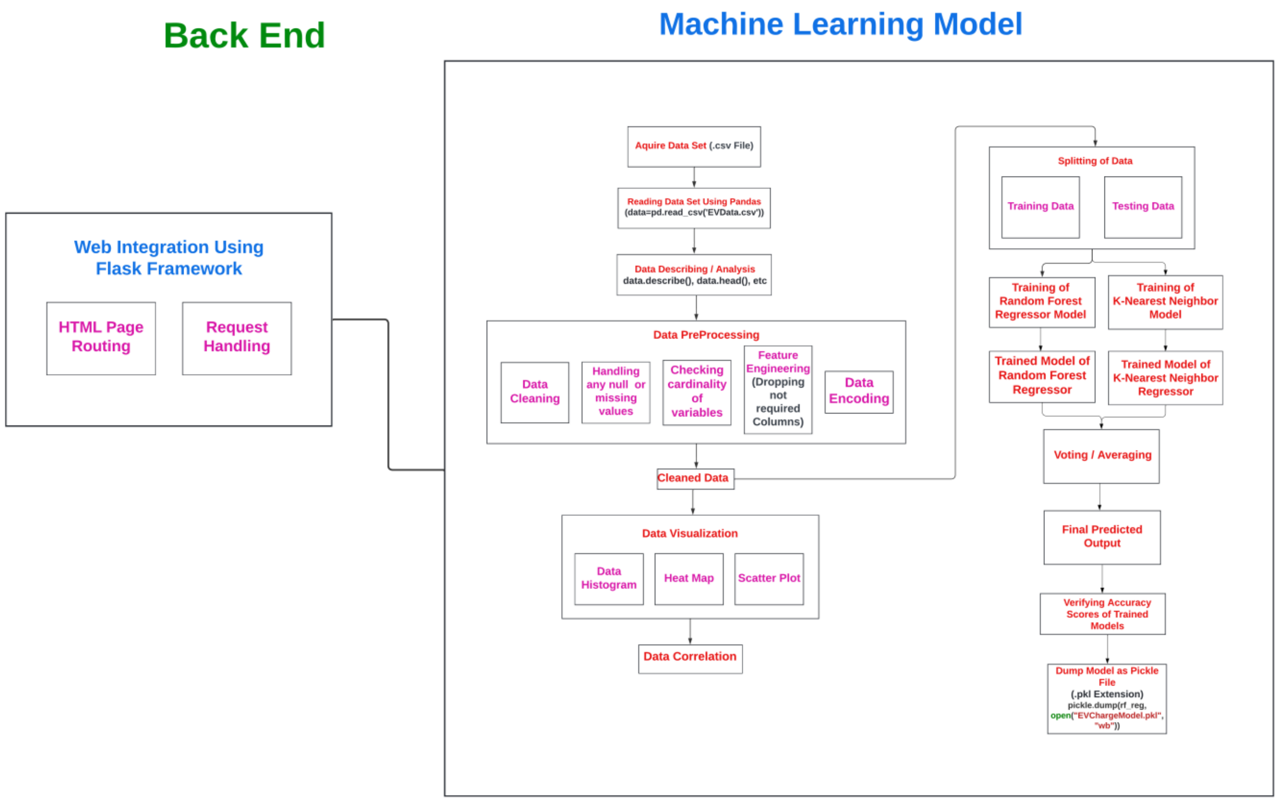

Back End
Machine Learning Model
Web Integration Using Flask Framework
HTML Page Routing
Request Handling
Aquire Data Set (.csv File)
Reading Data Set Using Pandas (data=pd.read_csv('EVData.csv'))
Data Describing / Analysis data.describe(), data.head(), etc
Data PreProcessing
Data Cleaning
Handling any null or missing values
Checking cardinality of variables
Feature Engineering (Dropping not required Columns)
Data Encoding
Cleaned Data
Data Visualization
Data Histogram
Heat Map
Scatter Plot
Data Correlation
Splitting of Data
Training Data
Testing Data
Training of Random Forest Regressor Model
Training of K-Nearest Neighbor Model
Trained Model of Random Forest Regressor
Trained Model of K-Nearest Neighbor Regressor
Voting / Averaging
Final Predicted Output
Verifying Accuracy Scores of Trained Models
Dump Model as Pickle File (.pkl Extension) pickle.dump(rf_reg, open("EVChargeModel.pkl", "wb"))

Fig 4.2.11 (System Architecture Back End Diagram for EV Charging Optimization)

EXPLANATION :

The back-end, powered by Python and Flask, handles data processing, model prediction, and serves as a bridge between the front-end interface and the machine learning model, ensuring seamless communication and efficient data flow.

CHAPTER 5

DEVELOPMENT TOOLS

5.1 PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

5.2 HISTORY OF PYTHON

- Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.
- Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).
- Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

5.3 IMPORTANCE OF PYTHON

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

5.4 FEATURES OF PYTHON

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

5.5 MACHINE LEARNING

Machine learning is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and models that enable computers to learn patterns and make predictions or decisions without being explicitly programmed. It is based on the idea that systems can learn from data, identify patterns, and improve their performance over time.

There are several types of machine learning approaches:

Supervised Learning : The algorithm is trained on a labeled dataset, where it learns the relationship between input features and corresponding output labels. The goal is to make accurate predictions on new, unseen data.

UnSupervised Learning : The algorithm is given data without explicit instructions on what to do with it. It must discover patterns or relationships within the data, often through techniques like clustering or dimensionality reduction.

ReInforcement Learning : The algorithm learns by interacting with an environment and receiving feedback in the form of rewards or penalties. The goal is to learn the optimal actions to take in different situations to maximize cumulative reward.Machine learning is applied across various domains, including image and speech recognition, natural language processing, recommendation systems, autonomous vehicles, and predictive analytics. It has the potential to automate complex tasks, improve decision-making processes, and uncover insights from vast amounts of data. As technology continues to advance, machine learning plays a central role in driving innovation and transforming the way we approach problem-solving in diverse fields.

5.6 PYTHON LIBRARIES FOR MACHINE LEARNING

Python has become a dominant programming language in the field of machine learning (ML) due to its simplicity, versatility, and the availability of robust libraries specifically designed for ML tasks.

Here are some key libraries in Python for machine learning:

Numpy : NumPy is a fundamental library for numerical operations in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy is the foundation for many other machine learning libraries.

Pandas : Pandas is widely used for data manipulation and analysis. It offers data structures like DataFrame, which is especially useful for handling structured data. Pandas simplifies tasks such as cleaning, transforming, and exploring datasets before feeding them into machine learning models.

Scikit-Learn : Scikit-learn is a comprehensive machine learning library that provides simple and efficient tools for data analysis and modeling. It includes various algorithms for classification, regression, clustering, and dimensionality reduction. Scikit-learn is designed to work seamlessly with other scientific libraries like NumPy and SciPy.

Matplotlib And Seaborn : Matplotlib and Seaborn are essential for data visualization in Python. Matplotlib offers a wide range of static, animated, and interactive plots, while Seaborn provides a high-level interface for creating attractive and informative statistical graphics.

TensorFlow And PyTorch : TensorFlow and PyTorch are deep learning frameworks that facilitate the creation of neural networks and deep learning models. They offer flexibility and scalability for building and training complex models, making them popular choices for deep learning tasks.

Keras : Keras is a high-level neural networks API that runs on top of TensorFlow and Theano. It simplifies the process of building and training neural networks by providing a user-friendly interface, making it an excellent choice for beginners in deep learning.

NLTK And SpaCy : Natural Language Toolkit (NLTK) and SpaCy are libraries for natural language processing (NLP). NLTK is widely used for tasks such as tokenization, stemming, and part-of-speech tagging, while SpaCy focuses on providing efficient and production-ready tools for NLP.

StatsModels : Statsmodels is a library for estimating and testing statistical models. It is particularly useful for traditional statistical analyses, hypothesis testing, and regression modeling.

XGBoost And LightBGM : XGBoost and LightGBM are gradient boosting frameworks that excel in handling tabular data and structured datasets. They are widely used for classification and regression tasks, offering high performance and efficiency.

OpenCV : OpenCV is a computer vision library that provides a wide range of tools for image and video processing. It is essential for tasks such as object detection, image recognition, and feature extraction.

These libraries collectively form a powerful ecosystem in Python for machine learning, covering a broad spectrum of tasks from data preprocessing and analysis to building and deploying complex machine learning models.

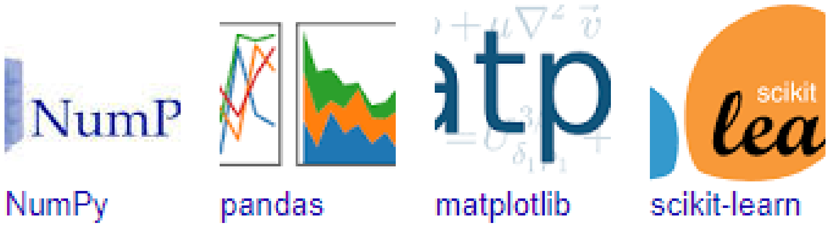


Figure : NumPy, Pandas, Matplotlib, Scikit-learn

5.7 FEATURES OFFERED BY PYTHON FOR MACHINE LEARNING

Python and machine learning (ML) have become closely intertwined, and Python is now the de facto language for many aspects of ML development. Several factors contribute to this strong relationship:

Ease Of Learning And Use : Python's syntax is clear and readable, making it accessible for beginners. This ease of learning and use has contributed to Python's popularity in ML, allowing practitioners to focus on solving problems rather than wrestling with the complexities of the language.

Extensive Libraries : Python boasts a rich ecosystem of libraries and frameworks that cater specifically to machine learning and data science. Libraries like NumPy, Pandas, Scikit-learn, TensorFlow, and PyTorch provide efficient tools for various ML tasks, streamlining development and research processes.

Community Support : Python has a large and active community of developers. This community support ensures that there is a wealth of resources, tutorials, and documentation available for anyone working on ML projects in Python. Collaboration and knowledge sharing are facilitated through platforms like GitHub and forums.

Versatility : Python is a versatile language, capable of handling diverse tasks. It seamlessly integrates with other languages and technologies, making it a preferred choice for end-to-end ML development—from data preprocessing and model training to deployment.

Data Science And Visualization : Python excels in data science and visualization, critical components of the ML workflow. Libraries like Pandas and Matplotlib provide tools for handling and exploring datasets, and Jupyter Notebooks allow for interactive and iterative development, making it easy to visualize results and share findings.

Deep Learning Frameworks : Major deep learning frameworks, such as TensorFlow and PyTorch, have official Python APIs, making Python the primary language for developing and implementing complex neural network architectures. Keras, a high-level neural networks API, is also seamlessly integrated with Python.

Rapid Prototyping : Python's dynamic nature and interpreted paradigm facilitate rapid prototyping. ML researchers and developers can quickly experiment with different algorithms, models, and parameters, accelerating the iterative development process.

Big Data Integration : Python integrates well with big data processing frameworks like Apache Spark. This enables ML practitioners to work with large-scale datasets and distributed computing environments seamlessly.

Support For Multiple Paradigms : Python supports various programming paradigms, including procedural, object-oriented, and functional programming. This flexibility allows developers to choose the paradigm that best suits their ML project requirements.

Deployment And Integration : Python's flexibility extends to deployment and integration. Models developed in Python can be integrated into various production environments, and tools like Flask and Django facilitate the development of web-based applications with ML capabilities.

In summary, Python's simplicity, extensive libraries, community support, and adaptability make it an ideal language for machine learning development. Its role spans from initial data exploration to the implementation of complex models, and its integration with deep learning frameworks has solidified its position as the language of choice for the ML community.

CHAPTER 6

IMPLEMENTATION

6.1 GENERAL

6.2 IMPLEMENTATION & SOURCE CODE

FILE NAME : Electric_Vehicle.ipynb

# Importing Required Libraries

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.ensemble import VotingClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score
```

# Analyzing Data Set

```
df = pd.read_csv('C:/Users/PythonTeam/Desktop/ELECTRICAL/Dataset-1_EV-charging-reports.csv')

df.columns
```

```
Index(['session_ID', 'Garage_ID', 'User_ID', 'User_type', 'Shared_ID',
       'Start_plugin', 'Start_plugin_hour', 'End_plugout', 'End_plugout_hour',
       'El_kWh', 'Duration_hours', 'month_plugin', 'weekdays_plugin',
       'Plugin_category', 'Duration_category'], dtype='object')
```

# Data PreProcessing

```
# Preprocessing - replace commas with periods in 'Duration_hours' and convert to float
```

```
df['Duration_hours'] = df['Duration_hours'].str.replace(',', '.').astype(float)
# Replace NaN values with zeros in the entire DataFrame
df.fillna(0, inplace=True)
# Convert 'Duration_category' to strings for classification
df['Duration_category'] = df['Duration_category'].astype(str)
# Selecting Features values & target values
# Define features and target
features = ['Start_plugin_hour', 'End_plugout_hour', 'Duration_hours']  # Add more features as needed
target = 'Duration_category'
X = df[features]
y = df[target]
# Split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Applying Ensemble Model
# Using KNN & Random Forest Algorithm with Hard Voting
knn = KNeighborsClassifier(n_neighbors=5)
rf = RandomForestClassifier(n_estimators=100, random_state=42)
ensemble_model = VotingClassifier(estimators=[('knn', knn), ('rf', rf)], voting='hard')
ensemble_model.fit(X_train, y_train)
ensemble_pred = ensemble_model.predict(X_test)
ensemble_accuracy = accuracy_score(y_test, ensemble_pred)
print("Ensemble Accuracy:", ensemble_accuracy)
```

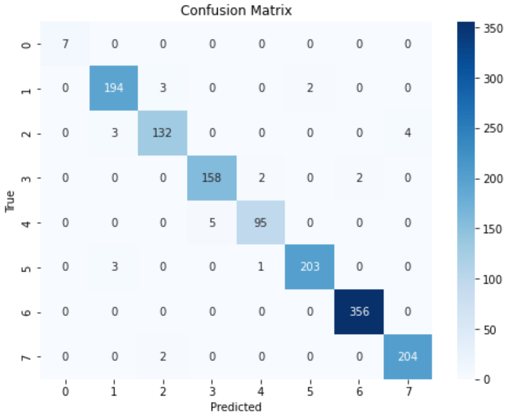
```
Ensemble Accuracy: 0.9883720930232558
```

```
# Classification report
report = classification_report(y_test, predictions)
```

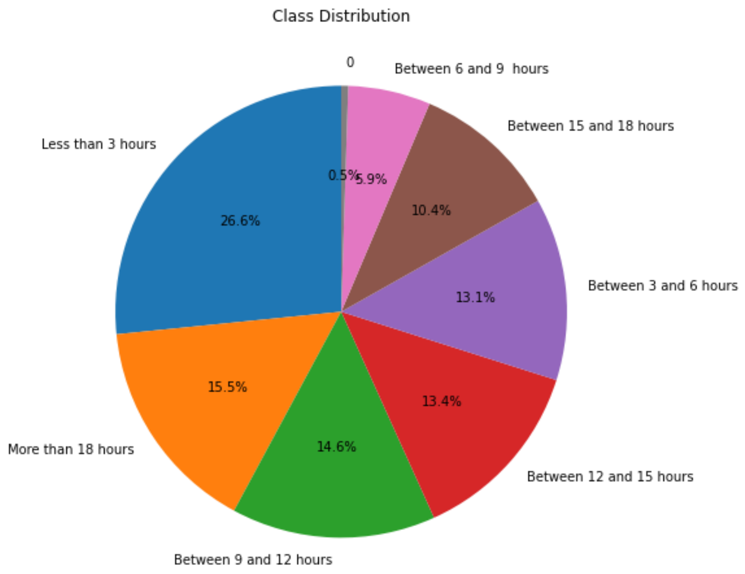
```
print("Classification Report:")

print(report)
```

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

                       0       1.00      1.00      1.00         7
Between 12 and 15 hours       0.97      0.97      0.97       199
Between 15 and 18 hours       0.96      0.95      0.96       139
  Between 3 and 6 hours       0.97      0.98      0.97       162
 Between 6 and 9  hours       0.97      0.95      0.96       100
 Between 9 and 12 hours       0.99      0.98      0.99       207
      Less than 3 hours       0.99      1.00      1.00       356
     More than 18 hours       0.98      0.99      0.99       206

               accuracy                           0.98      1376
              macro avg       0.98      0.98      0.98      1376

           weighted avg       0.98      0.98      0.98      1376
```

```
# Confusion matrix

conf_matrix = confusion_matrix(y_test, predictions)

plt.figure(figsize=(8, 6))

sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()
```



```
# Pie chart for class distribution

class_distribution = df['Duration_category'].value_counts()

plt.figure(figsize=(8, 8))

class_distribution.plot(kind='pie', autopct='%1.1f%%', startangle=90)

plt.title('Class Distribution')

plt.ylabel('')

plt.show()
```



```
#Save Model

import pickle

with open('Ensemble_Approach_Model.pkl', 'wb') as file:

pickle.dump(model, file)
```

FILE NAME : App.py

```
from flask import Flask, render_template, request, redirect, url_for

import pandas as pd

import pickle
```

```
app = Flask(__name__)
# Load the saved model
with open('naive_bayes_model.pkl', 'rb') as file:
    knn_model = pickle.load(file)
################################################
@app.route('/', methods=['POST','GET'] )
def first():
    return render_template('home.html')
################################################
users = {
    "electric": "pass123",
}
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if username in users and users[username] == password:
            return render_template('main.html')
        else:
            return render_template('login.html', error="Invalid username or password")
    return render_template('login.html', error="")
#################################################
# Define a route to render the home page
@app.route('/home', methods=['POST','GET'] )
def home():
    return render_template('main.html')
########################################################
# Define a route to handle form submission and make predictions
@app.route('/predict', methods=['POST','GET'])
def predict():
    if request.method == 'POST':
        # Get user input from the form
        start_hour = float(request.form['start_hour'])
        end_hour = float(request.form['end_hour'])
        duration = float(request.form['duration'])
        # Create a DataFrame from user input
        new_input_data = pd.DataFrame({
            'Start_plugin_hour': [start_hour],
            'End_plugout_hour': [end_hour],
            'Duration_hours': [duration]
        })
        # Make predictions using the loaded model
        prediction = knn_model.predict(new_input_data)
        return render_template('main.html', prediction=prediction[0])
#########################################################
@app.route('/performance')
def performance():
    return render_template('performance.html')
########################################################
@app.route('/logout')
def logout():
    return render_template('home.html')
#################################################
@app.route('/about')
def about():
    return render_template('about.html')
if __name__ == '__main__':
    app.run()
```

FILE NAME : Home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Home Page</title>
    <style>
        body {
                font-family: Arial, sans-serif;
                margin: 0;
                padding: 0;
                background-image: url('/static/back.jpg');
                background-size: auto 100vh; /* Adjust background size */
                background-position: center center;
                background-attachment: fixed; }
        header {
            background-color: #041322;
            color: #6cc4c1;
            padding: 10px 20px;
            text-align: center; }
        nav ul {
            list-style-type: none;
            padding: 0; }
        nav ul li {
            display: inline;
            margin-right: 20px;
            border: 1px solid #fff;
            padding: 5px 10px;
            border-radius: 5px;
            transition: background-color 0.3s;  }
        nav ul li a {
            color: #fff;
            text-decoration: none;
        }
        nav ul li:hover {
            background-color: #6cc4c1; }
        nav ul li:hover a {
            color: #333;
        }
        h3 {
            padding: 5px 10px;
            text-align: center;}
        footer {
```

```
            background-color: #041322;
            color: #fff;
            text-align: center;
            padding: 10px 0;
            position: fixed;
            bottom: 0;
            width: 100%; }
    </style>
</head>
<body>
    <header>
        <h1>ELECTRA SYNERGY</h1>
        <h3>A MACHINE LEARNING APPROACH FOR EV CHARGING OPTIMIZATION</h3>
        <nav>
            <ul>
                <li><a href="/">Home</a></li>
                <li><a href="/about">About</a></li>
                <li><a href="/login">Login</a></li>
            </ul>
        </nav>
    </header>
    <footer>
        <p> Project By Karanveer Singh &copy; &trade;</p>
    </footer>
</body>
</html>\
```

FILE NAME : Login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Home Page</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;}
        header {
            background-color: #041322;
            color: #6cc4c1;
            padding: 20px;
            text-align: center; }
        nav ul {
            list-style-type: none;
            padding: 0; }
        nav ul li {
            display: inline;
            margin-right: 20px;
            border: 1px solid #fff;
            padding: 5px 10px;
            border-radius: 5px;
            transition: background-color 0.3s; }
        nav ul li a {
            color: #fff;
            text-decoration: none;}
        nav ul li:hover {
            background-color: #6cc4c1; }
        nav ul li:hover a {
            color: #333; }
        main {
            padding: 20px; }
        footer {
            background-color: #041322;
            color: #fff;
            text-align: center;
            padding: 10px 0;
            position: fixed;
            bottom: 0;
            width: 100%; }
         .login-box {
            width: 300px;
            margin: 200px auto;
            background-color: #fff;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); }
        h2 {
            text-align: center;
            margin-bottom: 20px; }
        label {
            display: block;
            margin-bottom: 8px; }
        input[type="text"],
        input[type="password"] {
            width: 93%;
            padding: 8px;
            margin-bottom: 20px;
            border: 1px solid #ccc;
            border-radius: 3px; }
        input[type="submit"] {
            width: 100%;
            padding: 10px;
            border: none;
            background-color: #333;
            color: white;
            cursor: pointer;
            border-radius: 3px; }
        input[type="submit"]:hover {
            background-color: #333; }
        .error-message {
            color: red;
            text-align: center;
            margin-bottom: 10px; }
            </style>
        </head>
        <body>
            <header>
                <h1>ELECTRA SYNERGY</h1>
                <h3> A MACHINE LEARNING APPROACH FOR EV CHARGING OOPTIMIZATION</h2>
                <nav>
                    <ul>
                        <li><a href="/">Home</a></li>
                        <li><a href="/about">About</a></li>
                        <li><a href="/login">Login</a></li>
                    </ul>
                </nav>
            </header>
            <main style="background-image: url('/static/3back.jpg'); background-size: cover; background-position: center; padding: 20px;">
                <div class="login-box">
                <h2>Login</h2>
                {% if error %}
                    <p class="error-message">{{ error }}</p>
                {% endif %}
                <form method="post" action="/login">
                    <label for="username">Username:</label>
                    <input type="text" id="username" name="username">
                    <label for="password">Password:</label>
                    <input type="password" id="password" name="password">
                    <input type="submit" value="Login">
                </form>
            </div>
            </main>
            <footer>
                <p>Project By Karanveer Singh &copy; &trade;</p>
            </footer>
        </body>
        </html>
```

FILE NAME : Main.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Home Page</title>
</head>
<body>
    <header>
        <h1>ELECTRA SYNERGY</h1>
        <h3> A MACHINE LEARNING APPROACH FOR EV CHARGING OPTIMIZATION</h3>
        <nav>
            <ul>
                <li><a href="/">Home</a></li>
                <li><a href="#">Prediction</a></li>
                <li><a href="/performance">Model Performance</a></li>
            </ul>
        </nav>
    </header>
    <main style="background-image: url('/static/4back.jpg'); background-size: cover; background-position: center; padding: 20px; height: 100%;">
        <div style="width: 600px; margin: 100px auto; background-color: white; padding: 20px; border-radius: 10px; box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); text-align: center;">
            <h2>Enter EV Details</h2>
            <form method="post" action="/predict">
                <div class="form-row">
                    <div class="form-group">
                        <label for="start_hour">Plug-In Start Time:</label>
                        <input type="text" id="start_hour" name="start_hour" required>
                    </div>
                    <div class="form-group">
                        <label for="end_hour">Plug-Out End Time:</label>
                        <input type="text" id="end_hour" name="end_hour" required>
                    </div>
                </div>
                <div class="form-row">
                    <div class="form-group">
                        <label for="duration">Charging Duration (Hours):</label>
                        <input type="text" id="duration" name="duration" required>
                    </div>
                    <div class="form-group">
                        <label for="BatteryCapacity">Battery Capacity (kWh):</label>
        <input type="text" id="BatteryCapacity" name="BatteryCapacity">
    </div>
</div>
<div class="form-row">
    <div class="form-group">
        <label for="ChargerOutput">Charger Output (kW):</label>
        <input type="text" id="ChargerOutpur" name="ChargerOutput">
    </div>
    <div class="form-group">
        <label for="DrivingSpeed">Driving Speed (Km/Hr):</label>
        <input type="text" id="DrivingSpeed" name="DrivingSpeed">
    </div>
</div>
<div class="form-row">
    <div class="form-group">
        <label for="VehicleWeight">Vehicle Weight (Kg):</label>
        <input type="text" id="VehicleWeight" name="VehicleWeight">
    </div>
    <div class="form-group">
        <label for="BatteryTemperature">Battery Temperature (C°):</label>
        <input type="text" id="BatteryTemperature" name="BatteryTemperature">
    </div>
</div>
<div class="form-row">
    <div class="form-group">
        <label for="HVAC_Usage">HVAC Usage:</label>
        <select id="HVAC_Usage" name="HVAC_Usage">
            <option value="ON">ON</option>
            <option value="OFF">OFF</option>
        </select>
    </div>
    <div class="form-group">
        <label for="TerrainType">Terrain Type:</label>
        <select id="TerrainType" name="TerrainType">
            <option value="Flat">Flat</option>
            <option value="Hills_Or_Mountains">Hills / Mountains</option>
            <option value="Urban_Or_City">Urban / City</option>
            <option value="Highway">Highway</option>
            <option value="OffRoad">Off Road</option>
            <option value="Wet_Or_Slippery">Wet / Slippery</option>
            <option value="Snow_Or_Ice">Snow / Ice</option>
        </select>
    </div>
</div>
<div class="form-row">
    <div class="form-group">
        <label for="BatteryAge">Battery Age (Years):</label>
        <input type="text" id="BatteryAge" name="BatteryAge">
    </div>
    <div class="form-group">
        <label for="RegenerativeBraking">Regenerative Braking:</label>
        <select id="RegenerativeBraking" name="RegenerativeBraking">
```

```
                            <option value="ON">ON</option>
                            <option value="OFF">OFF</option>
                        </select>
                    </div>
                </div>
                <input type="submit" value="Predict">
            </form>
            {% if prediction %}
                <h3>Prediction:</h3>
                <p>{{ prediction }}</p>
            {% endif %}
        </div>
    </main>
    <footer>
        <p>Project By Karanveer Singh &copy; &trade;</p>
    </footer>
</body>
</html>
```

CHAPTER 7

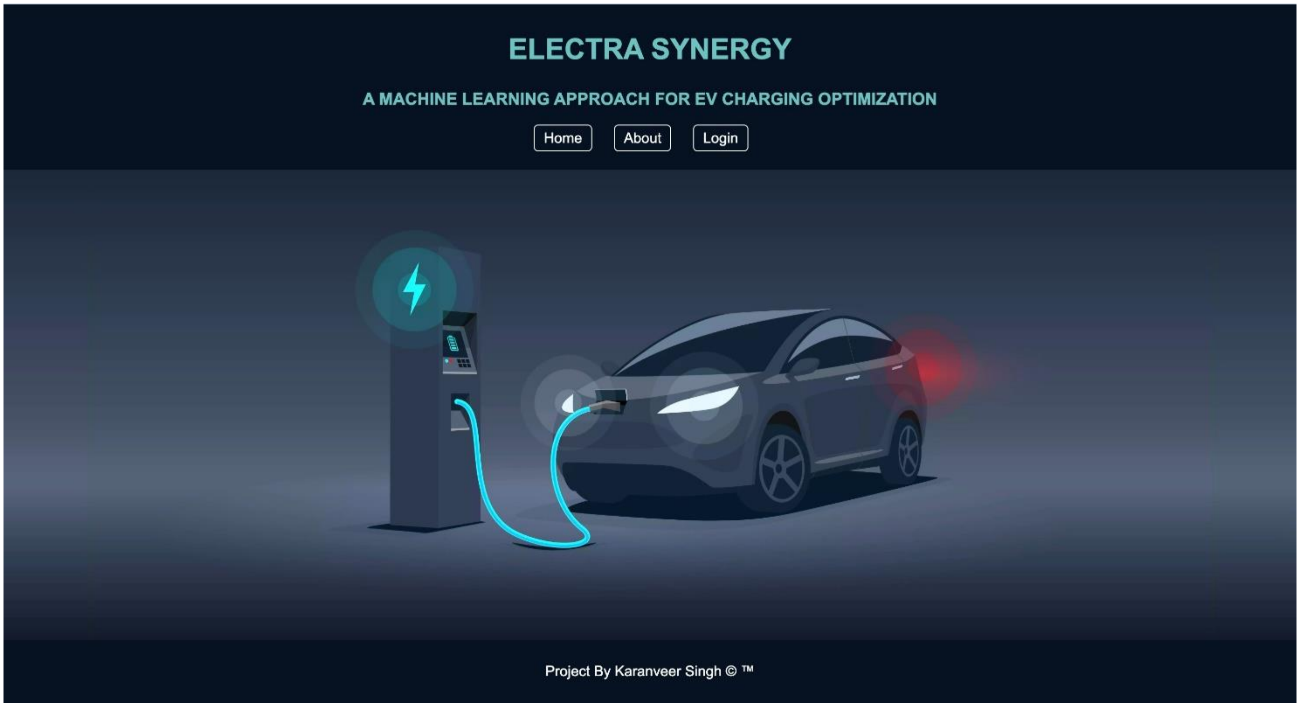
SNAPSHOTS

7.1 GENERAL

This project is implemented like an application using python and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

7.2 VARIOUS SNAPSHOTS

SNAPSHOT 1 : HOME PAGE



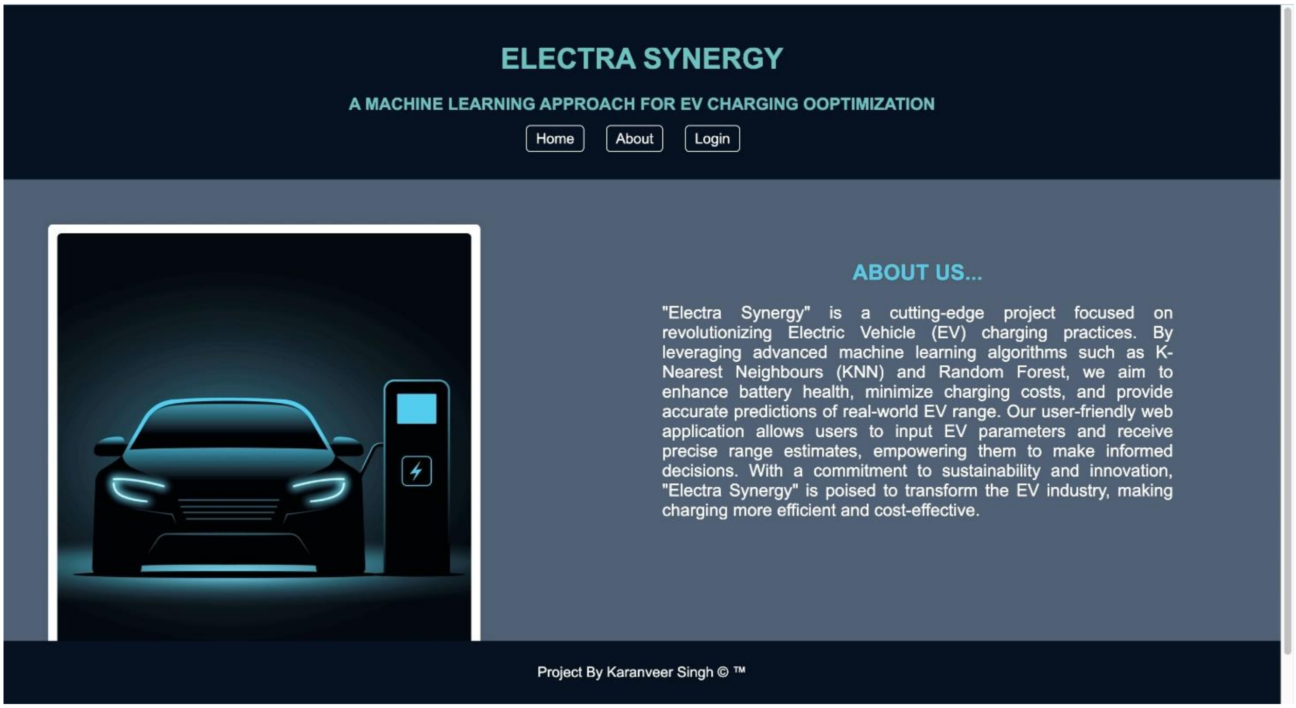
ELECTRA SYNERGY

A MACHINE LEARNING APPROACH FOR EV CHARGING OPTIMIZATION

Project By Karanveer Singh © ™

Description : The home page of Electra Synergy features a clean and intuitive design with three prominent buttons: "Home," "About Us," and "Login." These buttons provide easy navigation for users to access different sections of the application. The "Home" button allows users to return to the main page at any time, while the "About Us" button provides information about the project and its objectives. The "Login" button directs users to a secure login page.

SNAPSHOT 2 : ABOUT US PAGE


ELECTRA SYNERGY

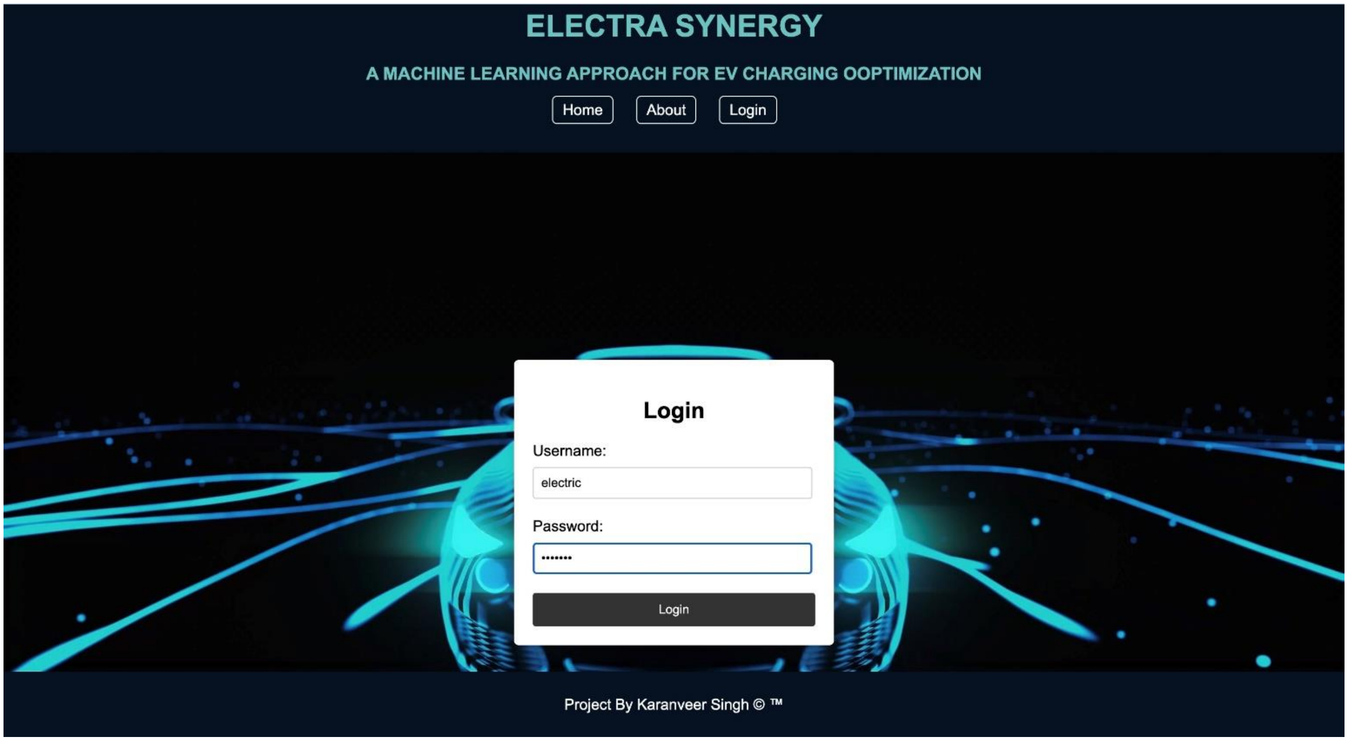
A MACHINE LEARNING APPROACH FOR EV CHARGING OOPTIMIZATION

ABOUT US...

"Electra Synergy" is a cutting-edge project focused on revolutionizing Electric Vehicle (EV) charging practices. By leveraging advanced machine learning algorithms such as K-Nearest Neighbours (KNN) and Random Forest, we aim to enhance battery health, minimize charging costs, and provide accurate predictions of real-world EV range. Our user-friendly web application allows users to input EV parameters and receive precise range estimates, empowering them to make informed decisions. With a commitment to sustainability and innovation, "Electra Synergy" is poised to transform the EV industry, making charging more efficient and cost-effective.

Project By Karanveer Singh © ™

SNAPSHOT 3 : LOGIN PAGE


ELECTRA SYNERGY

A MACHINE LEARNING APPROACH FOR EV CHARGING OOPTIMIZATION

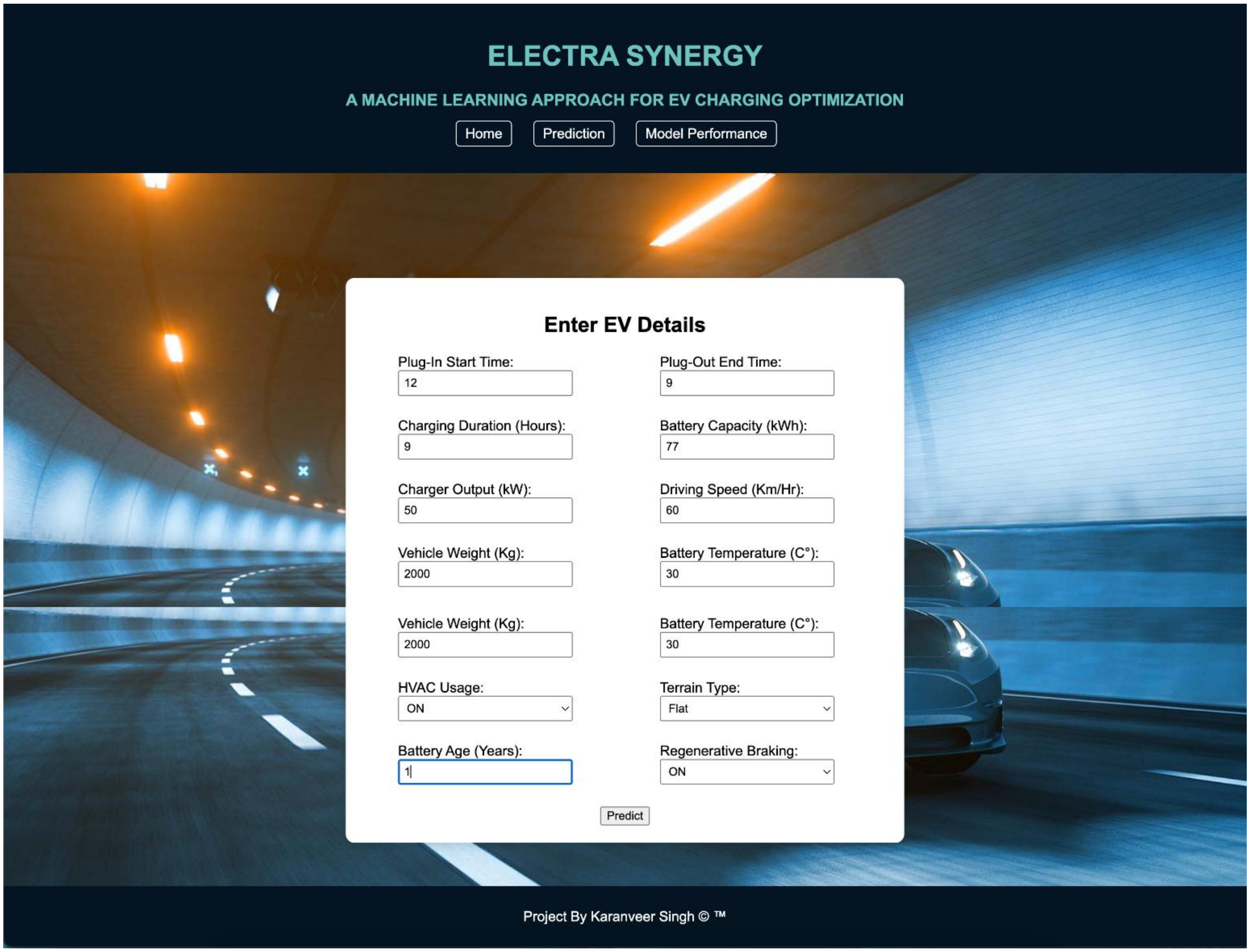
Login

Username: electric

Password: •••••••

Project By Karanveer Singh © ™

SNAPSHOT 4 : ENTERING EV DATA



ELECTRA SYNERGY

A MACHINE LEARNING APPROACH FOR EV CHARGING OPTIMIZATION

Enter EV Details

Plug-In Start Time: 12

Plug-Out End Time: 9

Charging Duration (Hours): 9

Battery Capacity (kWh): 77

Charger Output (kW): 50

Driving Speed (Km/Hr): 60

Vehicle Weight (Kg): 2000

Battery Temperature (C°): 30

Vehicle Weight (Kg): 2000

Battery Temperature (C°): 30

HVAC Usage: ON

Terrain Type: Flat

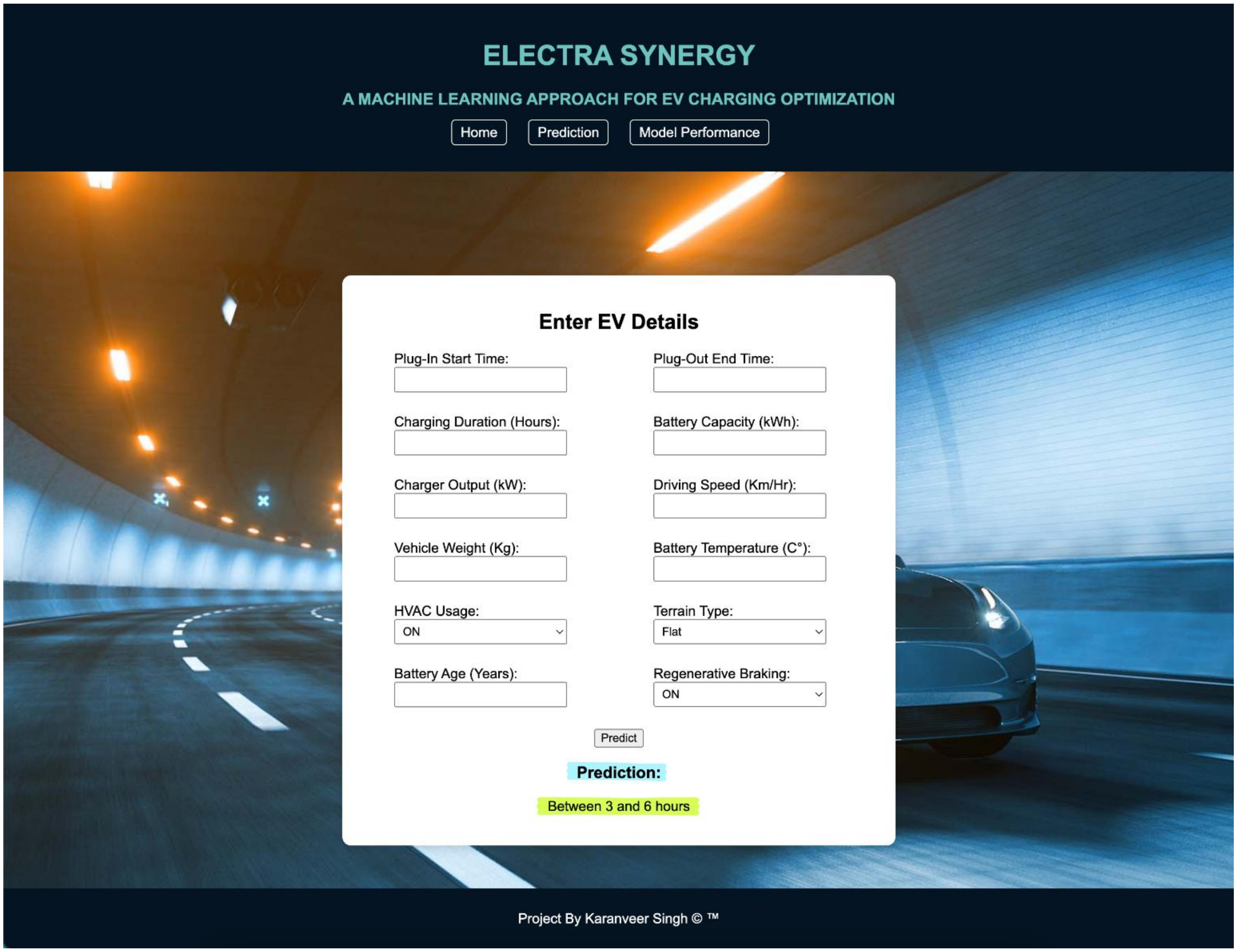
Battery Age (Years): 1

Regenerative Braking: ON

Project By Karanveer Singh © ™

Description : In this page the usersare presented with a user-friendly interface to input specific information about their electric vehicles (EVs) for charging optimization predictions. This interface includes fields for entering details such as the EV model, battery capacity, charging infrastructure availability, and typical driving patterns. Users are prompted to provide key information that influences charging strategies, such as battery type, capacity, and expected driving range. The design ensures a seamless user experience, with clear prompts and easy-to-navigate input fields.

SNAPSHOT 5 : PREDICTED EV RANGE (OUTPUT)



ELECTRA SYNERGY

A MACHINE LEARNING APPROACH FOR EV CHARGING OPTIMIZATION

Enter EV Details

Plug-In Start Time:

Plug-Out End Time:

Charging Duration (Hours):

Battery Capacity (kWh):

Charger Output (kW):

Driving Speed (Km/Hr):

Vehicle Weight (Kg):

Battery Temperature (C°):

HVAC Usage: ON

Terrain Type: Flat

Battery Age (Years):

Regenerative Braking: ON

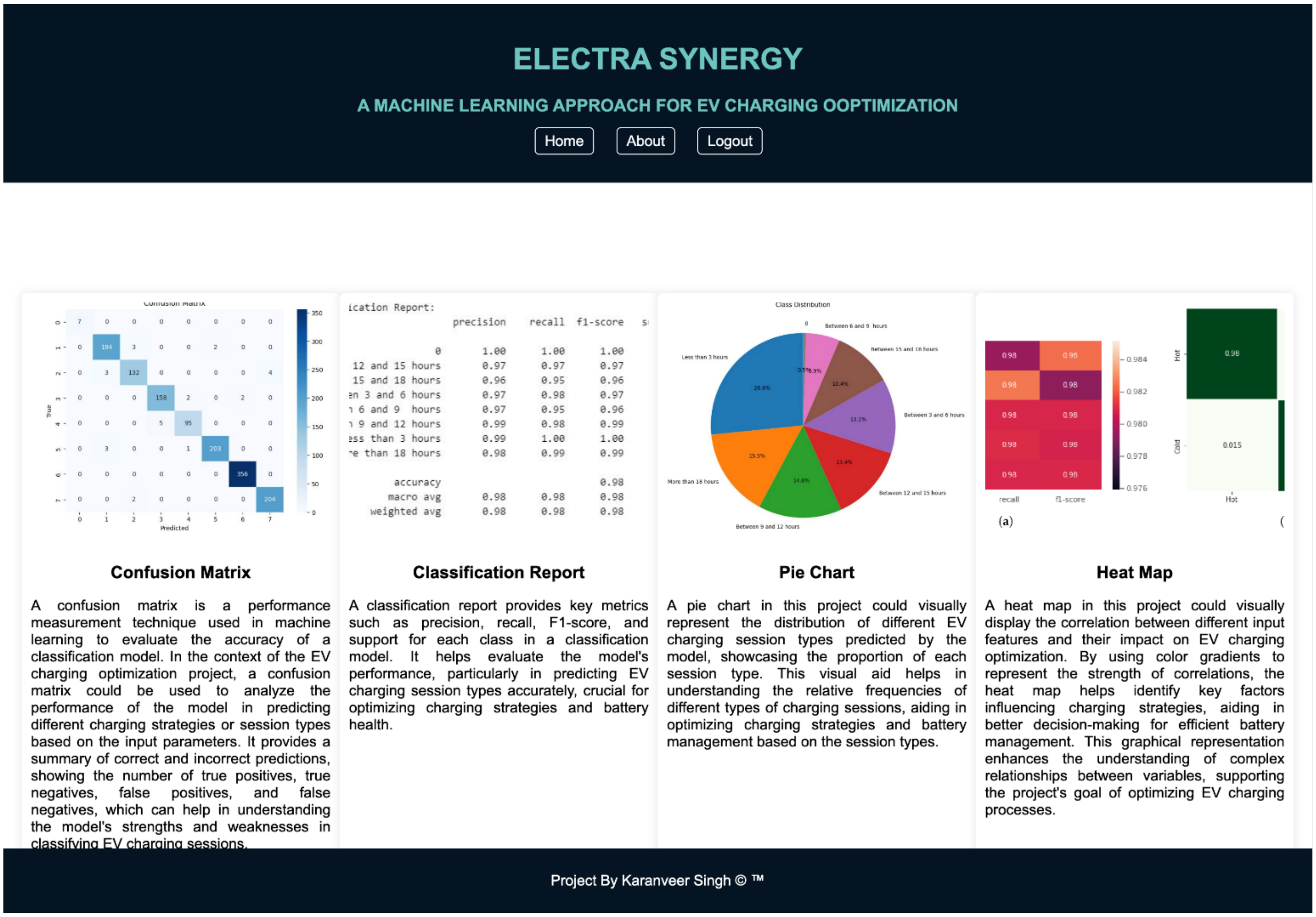
Prediction:

Between 3 and 6 hours

Project By Karanveer Singh © ™

Description : After clicking the "Predict" button, users are directed to a page displaying the predicted output for EV charging optimization. This page provides users with a comprehensive summary of the recommended charging strategy based on the input parameters provided earlier.

SNAPSHOT 6 : PERFORMANCE ANALYSIS OF THE MODEL



ELECTRA SYNERGY

A MACHINE LEARNING APPROACH FOR EV CHARGING OOPTIMIZATION

Confusion Matrix

True \ Predicted	0	1	2	3	4	5	6	7
0	7	0	0	0	0	0	0	0
1	0	194	3	0	0	2	0	0
2	0	3	132	0	0	0	0	4
3	0	0	0	158	2	0	2	0
4	0	0	0	5	95	0	0	0
5	0	3	0	0	1	203	0	0
6	0	0	0	0	0	0	356	0
7	0	0	2	0	0	0	0	204

```
ication Report:
                    precision    recall  f1-score   s

                 0       1.00      1.00      1.00
 12 and 15 hours       0.97      0.97      0.97
 15 and 18 hours       0.96      0.95      0.96
en 3 and 6 hours       0.97      0.98      0.97
 6 and 9  hours       0.97      0.95      0.96
 9 and 12 hours       0.99      0.98      0.99
ess than 3 hours       0.99      1.00      1.00
e than 18 hours       0.98      0.99      0.99

        accuracy                           0.98
       macro avg       0.98      0.98      0.98
    weighted avg       0.98      0.98      0.98
```

Confusion Matrix

A confusion matrix is a performance measurement technique used in machine learning to evaluate the accuracy of a classification model. In the context of the EV charging optimization project, a confusion matrix could be used to analyze the performance of the model in predicting different charging strategies or session types based on the input parameters. It provides a summary of correct and incorrect predictions, showing the number of true positives, true negatives, false positives, and false negatives, which can help in understanding the model's strengths and weaknesses in classifying EV charging sessions.

Classification Report

A classification report provides key metrics such as precision, recall, F1-score, and support for each class in a classification model. It helps evaluate the model's performance, particularly in predicting EV charging session types accurately, crucial for optimizing charging strategies and battery health.

Pie Chart

A pie chart in this project could visually represent the distribution of different EV charging session types predicted by the model, showcasing the proportion of each session type. This visual aid helps in understanding the relative frequencies of different types of charging sessions, aiding in optimizing charging strategies and battery management based on the session types.

Heat Map

A heat map in this project could visually display the correlation between different input features and their impact on EV charging optimization. By using color gradients to represent the strength of correlations, the heat map helps identify key factors influencing charging strategies, aiding in better decision-making for efficient battery management. This graphical representation enhances the understanding of complex relationships between variables, supporting the project's goal of optimizing EV charging processes.

Project By Karanveer Singh © ™

Description : After the prediction process, users are directed to a comprehensive performance analysis page showcasing the model's performance metrics and visualizations. This page includes a detailed overview of the model's accuracy, precision, recall, and F1-score, presented in a classification report format. Additionally, users can explore a confusion matrix, providing insights into the model's prediction errors and strengths. A pie chart illustrates the distribution of different EV charging session types, offering a visual representation of the model's predictions. Furthermore, a histogram and a heat map showcase the correlations between different input features and their impact on EV charging optimization. These visualizations aim to enhance user understanding of the model's performance and provide actionable insights for optimizing charging strategies.

CHAPTER 8

SOFTWARE TESTING

8.1 GENERAL

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

8.2 DEVELOPING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

8.3 TYPES OF TESTS

8.3.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.3.2 FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures: interfacing systems or procedures must be invoked.

8.3.3 SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.3.4 PERFORMANCE TEST

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

8.3.5 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

8.3.6 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

ACCEPTANCE TESTING FOR DATA SYNCHRONIZATION :

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updation process

8.4 BUILD THE TEST PLAN

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

CHAPTER 9

CONCLUSION

9.1 CONCLUSION

Electric vehicle (EV) charging optimization is a complex task due to the numerous factors involved. Data collection and preprocessing are crucial steps in this process. In this research, data normalization, standardization, and cleaning were meticulously performed to reduce noise and enhance the performance of machine learning algorithms. However, the dataset's complexities posed challenges, and employing other machine learning algorithms resulted in disappointing accuracies, falling below 50%.

To address these challenges, this project introduces an ensemble approach combining the strengths of both K-Nearest Neighbors (KNN) and Random Forest Algorithm Regressor Models. This novel approach significantly boosts accuracy, achieving an impressive rate of ~98%. This remarkable improvement underscores the effectiveness of the ensemble technique in tackling the intricacies of EV charging optimization.

Despite its success, it's essential to note that this ensemble approach requires more computational resources compared to other machine learning algorithms. However, the benefits it offers in terms of accuracy and optimization make it a compelling choice for EV charging optimization. Overall, this project represents a significant advancement in the field, paving the way for more efficient and sustainable EV charging practices.

CHAPTER 10

FUTURE ENHANCEMENTS

10.1 FUTURE ENHANCEMNETS AND APPLICATIONS

Although, this system has achieved astonishing performance in the field of EV charging optimization, our aim for the future research is the following :

Dynamic Charging Strategies : Develop algorithms that consider real-time factors like traffic conditions, weather, and grid load to dynamically adjust charging schedules. For example, during off-peak hours or when renewable energy generation is high, the algorithm could increase charging rates.

User Behavior Analysis : Use machine learning to analyze user preferences, driving patterns, and historical charging data. This analysis can help predict future charging needs, recommend optimal charging times, and offer personalized charging plans.

Energy Price Forecasting : Integrate models that predict electricity prices based on historical data, market trends, and demand-supply dynamics. This information can be used to schedule charging during low-cost periods, saving users money.

Grid Integration and Demand Response : Develop protocols for EVs to communicate with the grid and respond to signals for load management. This could involve charging at times when renewable energy is abundant or when the grid needs additional support.

Battery Health Monitoring : Implement algorithms to monitor battery health parameters such as state of charge, temperature, and charge cycles. This data can be used to provide recommendations for charging profiles that prolong battery life.

Enhanced Data Visualization : Use advanced visualization techniques to display charging patterns, energy consumption, and cost savings in an intuitive and interactive manner. This can help users understand their charging behavior and make informed decisions.

Incorporation of AI Assistants : Introduce AI-powered assistants that can answer user queries, provide real-time charging recommendations, and assist in setting up personalized charging profiles based on user preferences.

Integration with Smart Infrastructure : Collaborate with smart city initiatives to integrate EV charging infrastructure with smart grids, smart meters, and other IoT devices. This integration can enable automated, efficient, and optimized charging operations.

Predictive Maintenance : Implement predictive maintenance algorithms that analyze data from charging stations and EV components to anticipate potential failures. By detecting issues early, maintenance can be scheduled proactively, minimizing downtime.

Gamification and Incentive Programs : Introduce gamified elements such as rewards, challenges, and competitions to encourage users to adopt sustainable driving habits and maximize the benefits of electric vehicles.

Fleet Management Optimization : Implement the system in fleets of electric vehicles to optimize charging schedules, reduce operational costs, and improve overall fleet efficiency.

Electric Vehicle Sharing Services : Integrate the system into electric vehicle sharing platforms to manage and optimize charging schedules for shared vehicles, ensuring availability and reliability for users.

Smart Home Integration : Develop interfaces to connect with smart home systems, allowing users to coordinate charging schedules with other home energy consumption activities for better energy management.

Grid Stability and Peak Demand Management : Use the system to manage EV charging to mitigate grid instability and reduce peak demand, contributing to a more stable and efficient electricity grid.

Carbon Footprint Reduction : Utilize the system to calculate and minimize the carbon footprint of EV charging by optimizing charging schedules based on renewable energy availability and grid emissions.

Integration with Renewable Energy Sources : Integrate the system with renewable energy sources such as solar panels or wind turbines to optimize EV charging based on the availability of clean energy.

Emergency Response Vehicle Optimization : Apply the system to optimize charging for emergency response vehicles, ensuring they are always ready for deployment with sufficient charge.

Public Transportation Integration : Integrate the system with public transportation systems to optimize the charging schedules of electric buses and trains, improving the efficiency and reliability of public transportation services.

Vehicle-to-Grid (V2G) Integration : Explore the potential for V2G technology, where EVs can not only charge from the grid but also discharge electricity back to the grid during peak demand periods, providing grid stabilization services.

Healthcare and Elderly Care Services : Use the system to manage EVs used in healthcare and elderly care services, ensuring that vehicles are always charged and ready for transportation needs in these critical sectors.

These applications demonstrate the versatility and potential impact of the Electra Synergy project in various domains, highlighting its ability to revolutionize electric vehicle charging and contribute to a more sustainable future.

REFERENCES

[1] D. Ronanki, A. Hemasundar, and P. Parthiban, “A small 4-wheeler evpropulsion system using dtc controlled induction motor,” in Proceedingsof the World Congress on Engineering, vol. 2, 2013.

[2] Y. Jiang, L. Kang, and Y. Liu, “A unified model to optimize configurationof battery energy storage systems with multiple types of batteries,”Energy, vol. 176, pp. 552–560, 2019.

[3] J. Garche, A. Jossen, and H. Doring, “The influence of different ¨operating conditions, especially over-discharge, on the lifetime andperformance of lead/acid batteries for photovoltaic systems,” Journalof Power Sources, vol. 67, no. 1-2, pp. 201–212, 1997.

[4] S. S. Williamson, A. K. Rathore, and F. Musavi, “Industrial electronics for electric transportation: Current state-of-the-art and futurechallenges,” IEEE Transactions on Industrial Electronics, vol. 62, no. 5,pp. 3021–3032, 2015.

[5] D. Ronanki and S. S. Williamson, “Modular multilevel convertersfor transportation electrification: challenges and opportunities,” IEEETransactions on Transportation Electrification, vol. 4, no. 2, pp. 399–407, 2018.

[6] R. Zhang, B. Xia, B. Li, L. Cao, Y. Lai, W. Zheng, H. Wang, andW. Wang, “State of the art of lithium-ion battery soc estimation forelectrical vehicles,” Energies, vol. 11, no. 7, p. 1820, 2018.

[7] P. Shen, M. Ouyang, L. Lu, J. Li, and X. Feng, “The co-estimationof state of charge, state of health, and state of function for lithiumion batteries in electric vehicles,” IEEE Transactions on VehicularTechnology, vol. 67, no. 1, pp. 92–103, 2018.

[8] Y. Xing, W. He, M. Pecht, and K. L. Tsui, “State of charge estimationof lithium-ion batteries using the open-circuit voltage at various ambienttemperatures,” Applied Energy, vol. 113, pp. 106–115, 2014.

[9] D. Li, J. Ouyang, H. Li, and J. Wan, “State of charge estimation forlimn2o4 power battery based on strong tracking sigma point kalmanfilter,” Journal of Power Sources, vol. 279, pp. 439–449, 2015.

[10] T. Hansen and C.-J. Wang, “Support vector based battery state of chargeestimator,” Journal of Power Sources, vol. 141, no. 2, pp. 351–358, 2005.