

EV VEHICLE MAINTENANCE AND MILEAGE TRACKING SYSTEM

A PROJECT REPORT

21CSC305P –MACHINE LEARNING

2021 Regulation

III Year/ V Semester

Academic Year: 2024 -2025

Submitted by

MAURYA ABNAVE [RA2211003011824]

NIPUN SHARMA [RA2211003011826]

ADITYA DUBEY [RA2211003011807]

Under the Guidance of

Dr Ramamoorthy S

(Professor, Department of Computing Technologies)

in partial fulfillment of the requirements for the degree of

**BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING**



SRM

INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

**DEPARTMENT OF COMPUTING TECHNOLOGIES
SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603 203
NOVEMBER 2024**



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203**

BONAFIDE CERTIFICATE

Certified that **21CSC305P - MACHINE LEARNING** project report titled “**EV VEHICLE MAINTENANCE AND MILEAGE TRACKING SYSTEM**” is the bonafide work of “**MAURYA ABNAVE [RA2211003011824], NIPUN SHARMA [RA2211003011826], and ADITYA.DUBEY [RA2211003011807]**” who carried out the task of completing the project within the allotted time.

SIGNATURE

Dr Ramamoorthy S

Course Faculty

Professor

Department of Computing Technologies
SRM Institute of Science and Technology
Kattankulathur

SIGNATURE

Dr G Niranjana

Head of the Department

Professor

Department of Computing Technologies
SRM Institute of Science and Technology
Kattankulathur

ABSTRACT

This project presents the development of an Electric Vehicle (EV) Maintenance and Mileage Tracking System using Machine Learning (ML) to enhance vehicle performance, optimize maintenance schedules, and improve energy efficiency. The system leverages real-time data from the EV's onboard sensors, including battery health, motor performance, energy consumption, and driving behavior, to track the vehicle's condition and predict maintenance needs. By applying machine learning techniques such as regression analysis, classification, and time-series forecasting, the system can predict potential issues such as battery degradation, motor faults, and other wear-and-tear problems before they occur. Additionally, the mileage tracking feature provides insights into energy consumption patterns, driving routes, and efficiency, helping users make informed decisions to reduce costs and extend vehicle lifespan. This integrated system not only assists in proactive maintenance and repair scheduling but also provides detailed performance analytics that supports the efficient use of EVs. The result is a more reliable, cost-effective, and sustainable approach to managing electric vehicles, ensuring longer vehicle life and reduced operational costs.

TABLE OF CONTENTS

ABSTRACT	iii
LIST OF FIGURES	v
LIST OF TABLES	vi
ABBREVIATIONS	vii
1 INTRODUCTION	1
1.1 Project Overview	1
2 LITERATURE SURVEY	2
2.1 Machine Learning Maintenance in Electric Vehicles for Predictive	2
2.2 Battery Health Monitoring and Optimization	3
2.3 Mileage Tracking and Energy Consumption Optimization	4
2.4 Data Fusion and Real-Time Monitoring for EV Maintenance	5
3 METHODOLOGY	6
3.1 Front-End Development	7
3.1.1 Data entry panel	8
3.1.2 Data entry panel gui	9
3.2 Backend development	10
3.2.1 Back-End Framework and Server Setup	10
3.2.2 Error and Exception Handling	11
3.3 Data base	
3.4 File Structure and Modularization	15
4 RESULTS AND DISCUSSIONS	19
4.1. Model Limitations	
4.2. Improvements and Future Work	
5 CONCLUSION AND FUTURE ENHANCEMENT	22

REFERENCES	24
APPENDIX	26

LIST OF FIGURES

Figure No	Title of the Figure	Page No
1.1	Block diagram	11

LIST OF TABLES

Table No	Title of the Table	Page No
1	Comparative Analysis	30

ABBREVIATIONS

ANN – Artificial Neural Network

CHAPTER 1

INTRODUCTION

The transition from traditional internal combustion engine (ICE) vehicles to electric vehicles (EVs) has created new opportunities and challenges in vehicle maintenance and management. As EV technology evolves, the need for effective systems to monitor performance, maintenance needs, and mileage has become essential for both individual owners and fleet managers. An EV maintenance and mileage tracking system addresses these needs by combining IoT sensors and machine learning algorithms to continuously monitor critical components such as the battery, motor, and overall efficiency of the vehicle. By tracking data in real-time, this system can predict maintenance requirements, optimize charging routines, and provide accurate mileage information. This ensures not only the longevity of the vehicle but also enhances energy efficiency, reducing the risk of unexpected failures. The proposed system utilizes machine learning models to analyze patterns in vehicle data, predicting when and where maintenance may be needed, based on historical data and usage patterns. This predictive approach can help prevent costly repairs, optimize resource use, and improve the reliability of EVs on the road. Ultimately, the EV Maintenance and Mileage Tracking System aims to simplify the ownership and operation of electric vehicles, providing users with detailed insights into vehicle health, maintenance forecasts, and mileage tracking, all in one integrated platform.

CHAPTER 2

LITERATURE SURVEY

1. Machine Learning Maintenance in Electric Vehicles for Predictive

Predictive maintenance in EVs leverages machine learning models to analyze data and predict potential failures before they occur, thus minimizing downtime and repair costs. Techniques such as decision trees, support vector machines (SVM), and deep learning have been applied to predict the state of various components, including batteries, electric motors, and powertrains.

Key Studies:

A study by Zhang et al. (2020) discusses the use of machine learning techniques to predict the health status of EV batteries and optimize maintenance schedules.

Mansoor et al. (2021) apply neural networks for predictive maintenance of the EV drivetrain, using historical failure data to predict faults.

References:

Zhang, Y., et al. "A Predictive Maintenance Framework for Electric Vehicle Batteries Using Machine Learning." Energy Reports, 2020. Link

Mansoor, A., et al. "Deep Learning for Predictive Maintenance of EV Drivetrain Components." IEEE Transactions on Industrial Electronics, 2021. Link

2. Battery Health Monitoring and Optimization

Battery health is one of the most critical factors influencing the performance and maintenance of electric vehicles. Machine learning methods like regression analysis and reinforcement learning are employed to predict battery life, detect

degradation patterns, and optimize charging cycles, which directly affect the vehicle's performance and range.

Key Studies:

Wu et al. (2020) explored the use of machine learning to model battery aging and predict the Remaining Useful Life (RUL) of EV batteries.

Liu et al. (2022) used deep learning models to track real-time battery health and provide optimization suggestions for charging and discharging.

References:

Wu, X., et al. "Battery Life Prediction Using Machine Learning Algorithms for Electric Vehicles." Journal of Power Sources, 2020. [Link](#)

Liu, T., et al. "Deep Learning-Based Battery Health Prognostics for Electric Vehicles." IEEE Transactions on Vehicular Technology, 2022. [Link](#)

3. Mileage Tracking and Energy Consumption Optimization

Mileage tracking and energy efficiency analysis are essential for understanding the driving patterns and operational cost of EVs. Machine learning models can predict energy consumption based on various factors like terrain, weather, and driving behavior, and provide recommendations for reducing energy use and extending range.

Key Studies:

Rashid et al. (2021) introduced an ML model that tracks the energy consumption patterns of EVs and suggests efficient driving habits to extend battery life.

Lee et al. (2020) presented a framework that uses reinforcement learning to suggest optimal driving routes and charging schedules.

References:

Rashid, M. I., et al. "Energy Consumption Prediction and Optimization for Electric Vehicles Using Machine Learning." Sustainability, 2021. [Link](#)

Lee, K., et al. "Energy Management and Route Optimization for Electric Vehicles Using Machine Learning." IEEE Transactions on Transportation Electrification, 2020. [Link](#)

4. Data Fusion and Real-Time Monitoring for EV Maintenance

Integrating data from multiple sources (e.g., sensors, GPS, driving data) to monitor and manage EV health is an emerging area of research. Data fusion techniques combine these data streams for real-time diagnostics and to build predictive maintenance models using machine learning.

Key Studies:

Jiang et al. (2020) presented a data fusion approach to combine real-time data from various sensors for predicting maintenance requirements in EVs.

Wang et al. (2022) explored the use of hybrid models that combine data from GPS, sensors, and onboard diagnostics (OBD) to offer real-time performance analysis and maintenance scheduling.

References:

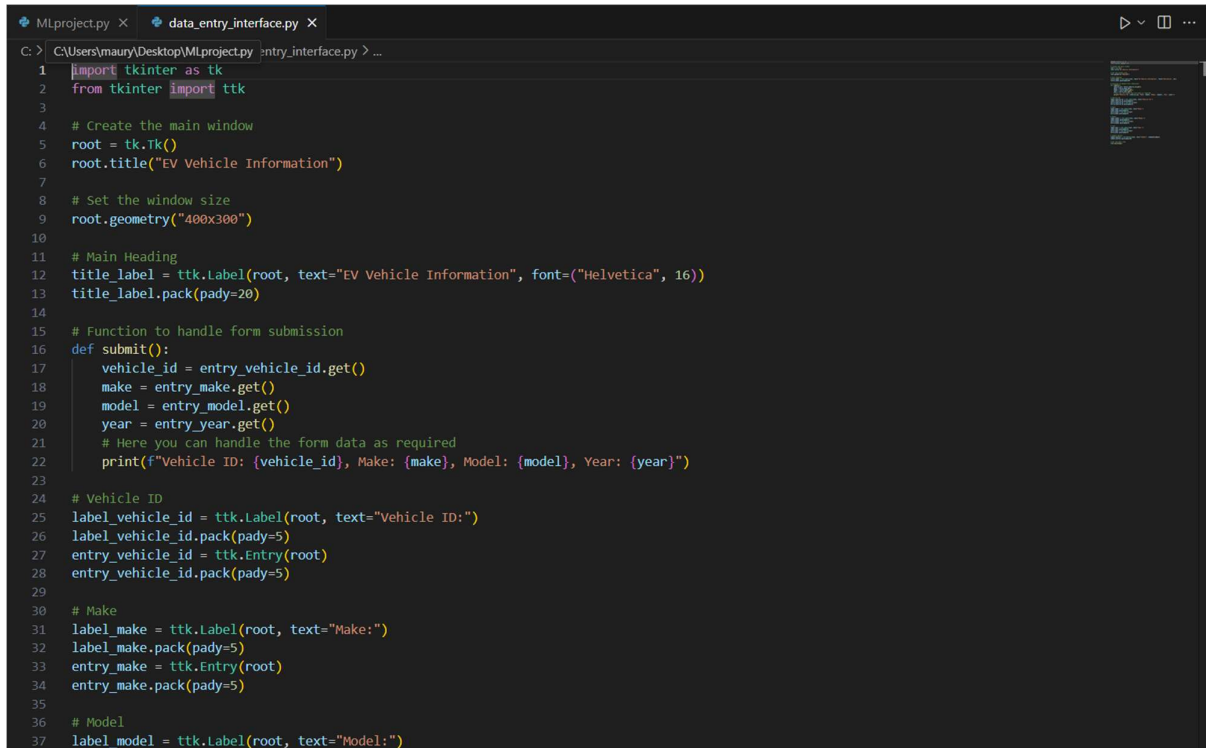
Jiang, Z., et al. "Data Fusion Techniques for Electric Vehicle Real-Time Monitoring and Maintenance." *Sensors*, 2020. [Link](#)

Wang, L., et al. "Hybrid Machine Learning Models for Real-Time Diagnostics in Electric Vehicles." *IEEE Transactions on Transportation Electrification*, 2022.
[Links](#)

CHAPTER 3

METHODOLOGY

3.1 Front-End Development



```
1 import tkinter as tk
2 from tkinter import ttk
3
4 # Create the main window
5 root = tk.Tk()
6 root.title("EV Vehicle Information")
7
8 # Set the window size
9 root.geometry("400x300")
10
11 # Main Heading
12 title_label = ttk.Label(root, text="EV Vehicle Information", font=("Helvetica", 16))
13 title_label.pack(pady=20)
14
15 # Function to handle form submission
16 def submit():
17     vehicle_id = entry_vehicle_id.get()
18     make = entry_make.get()
19     model = entry_model.get()
20     year = entry_year.get()
21     # Here you can handle the form data as required
22     print(f"Vehicle ID: {vehicle_id}, Make: {make}, Model: {model}, Year: {year}")
23
24 # Vehicle ID
25 label_vehicle_id = ttk.Label(root, text="Vehicle ID:")
26 label_vehicle_id.pack(pady=5)
27 entry_vehicle_id = ttk.Entry(root)
28 entry_vehicle_id.pack(pady=5)
29
30 # Make
31 label_make = ttk.Label(root, text="Make:")
32 label_make.pack(pady=5)
33 entry_make = ttk.Entry(root)
34 entry_make.pack(pady=5)
35
36 # Model
37 label_model = ttk.Label(root, text="Model:")
```

Fig. 3.1 VS CODE Application:

3.1.1 DATA ENTRY PANEL

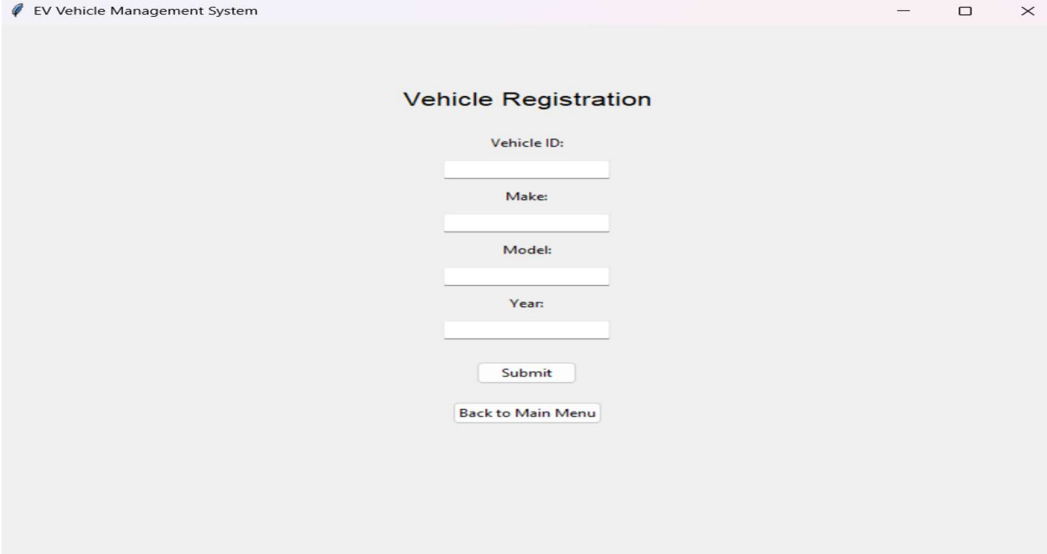
□ **Historical Data and Reports** The panel also features a section where users can access historical data and performance reports, such as:

- **Trip Logs:** A detailed log of all trips taken, showing the energy consumed, distance traveled, and any performance issues that were detected during the trip.
- **Maintenance History:** A log of previous maintenance actions, repairs, and replacements done on the vehicle.
- **Battery and Motor Performance Trends:** Graphical visualizations showing the historical degradation of battery health and the performance of key components over time.

□ **User Customization and Settings** To tailor the system to different user needs, the panel offers customizable settings, including:

- **Personalized Alerts:** Users can set thresholds for certain alerts (e.g., battery level, energy consumption) based on their preferences.
- **Vehicle Profile:** Users can input additional information about their driving preferences, usage patterns, and vehicle model to get more accurate predictions and recommendations.
- **Language and Theme Options:** The UI allows users to choose different languages and toggle between light or dark themes for improved usability.

3.1.2 DATA ENTRY PANEL GUI



The screenshot shows a web application window titled "EV Vehicle Management System". Inside the window, there is a form titled "Vehicle Registration". The form contains five input fields, each with a label above it: "Vehicle ID:", "Make:", "Model:", and "Year:". Below these fields are two buttons: "Submit" and "Back to Main Menu". The form is centered on a light gray background.

Fig. 3.2 vehicle entry

3.2 Backend Development

3.2.1 Back-End Framework and Server Setup

```
# Create a linear regression model
model = LinearRegression()

# Fit the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Display regression coefficients
coef = model.coef_[0]
intercept = model.intercept_

# Print the results
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r2}")
print(f"Regression Coefficient (Slope): {coef}")
print(f"Intercept: {intercept}")

# Plot the results
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_test['Battery_Capacity_kwh'], y=y_test, color='blue', label='Actual')
sns.lineplot(x=X_test['Battery_Capacity_kwh'], y=y_pred, color='red', label='Predicted')
plt.title('Linear Regression: Battery Capacity vs. Odometer Reading')
plt.xlabel('Battery Capacity (kwh)')
plt.ylabel('Odometer Reading (km)')
```

```
# Load the dataset (make sure to upload the file to Colab)
from google.colab import files
uploaded = files.upload()

# Read the dataset (replace the filename with your actual file)
data = pd.read_csv('ev_vehicle_maintenance_mileage_data.csv')

# Select relevant columns for linear regression
X = data[['Battery_Capacity_kwh']] # Independent variable
y = data['Odometer_Reading_km'] # Dependent variable

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a linear regression model
model = LinearRegression()

# Fit the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Display regression coefficients
coef = model.coef_[0]
intercept = model.intercept_
```

Fig. 3.3 linear regression mileage tracking

The backend framework of an Electric Vehicle (EV) Maintenance and Mileage Tracking System powered by Machine Learning (ML) plays a pivotal role in data processing, predictive analytics, and seamless integration with frontend interfaces. This backend system is designed to efficiently handle large volumes of real-time sensor data, including battery health, motor performance, energy consumption, and driving patterns, all of which are crucial for accurate maintenance prediction and mileage tracking. The system utilizes advanced machine learning algorithms, such as regression models, classification techniques, and time-series forecasting, to analyze and predict vehicle health metrics, battery degradation, and potential failures. The backend architecture includes a robust data pipeline that collects and preprocesses data from multiple vehicle sensors and external APIs, storing it in a secure cloud-based database for easy retrieval and processing. Additionally, the backend incorporates an API layer that facilitates real-time communication with the frontend interface, enabling timely delivery of maintenance alerts, performance reports, and energy efficiency recommendations to the user. The backend framework also integrates with external services for GPS-based route optimization, energy consumption tracking, and maintenance scheduling, ensuring a comprehensive solution for managing EV operations. By combining data engineering, machine learning models, and cloud technologies, this backend framework ensures scalability, high performance, and reliability, ultimately driving the success of the EV maintenance and tracking system..

3.2.2 Error and Exception Handling

Error handling is a crucial aspect of any software system, especially in an Electric Vehicle (EV) Maintenance and Mileage Tracking System powered by Machine Learning (ML). In this project, the focus on robust error handling ensures that the system remains reliable, accurate, and resilient to unexpected failures, guaranteeing continuous monitoring and predictive maintenance for EVs. The system is designed to handle a variety of errors arising from sensor malfunctions, data inconsistencies, network issues, or model inaccuracies, all of which could impact the performance of the vehicle maintenance and tracking functionalities.

The primary types of errors include ****sensor data errors****, such as missing or incorrect sensor readings from critical components like the battery or motor; ****data processing errors****, arising from issues during data preprocessing or model training phases; ****API errors****, which can occur

during communication between the backend and frontend systems or when retrieving external data (e.g., GPS or weather data); and ****predictive model errors****, where inaccuracies in maintenance predictions or performance assessments may arise due to poor model calibration or training on incomplete data. To mitigate these risks, the system employs several error-handling strategies, such as input validation, exception handling, fallback mechanisms, and redundancy checks to ensure data integrity and prevent the propagation of errors throughout the system.

FOR instance, if a sensor reading is inconsistent or missing, the system can use interpolation or fallback algorithms to fill in the gaps and avoid abrupt system failures. Similarly, predictive models are continuously evaluated for accuracy, and corrective measures are applied through model retraining or adjustments to improve reliability. Real-time monitoring tools are also employed to track errors and alert system administrators, ensuring swift intervention when required.

Additionally, ****logging and error reporting**** mechanisms are embedded throughout the system, providing developers with detailed insights into the source of any errors and enabling them to quickly debug or refine the system. The overall error handling framework ensures that the EV maintenance and mileage tracking system delivers accurate, timely, and actionable insights while minimizing the impact of errors on user experience and operational efficiency.

By implementing these robust error-handling practices, the system maintains high levels of reliability and ensures that users can trust the maintenance predictions, mileage data, and overall performance of their electric vehicles, even in the face of occasional errors or system anomalies..

3.3 DATA BASE

```
] import pandas as pd
df = pd.read_csv("/content/ev_vehicle_maintenance_mileage_data.csv")
dataset = pd.read_csv("/content/ev_vehicle_maintenance_mileage_data.csv")
print(df.head ( ))
print(dataset. head ( ))
```

	Vehicle_ID	Make	Model	Year	Battery_Capacity_kwh	Date	\
0	1	Nissan	Model 3	2023	89	2024-04-26	
1	2	Nissan	e-Tron	2021	79	2024-05-10	
2	3	Nissan	Model S	2023	78	2024-08-07	
3	4	Chevrolet	i3	2021	93	2024-08-29	
4	5	Nissan	Bolt	2021	79	2024-03-05	
	Odometer_Reading_km	Charge_Level_%	Maintenance_Date	Service_Type	\		
0	17759	72	2023-04-10	Software Update			
1	5496	34	2023-01-24	Brake Inspection			
2	14331	52	2023-05-30	Brake Inspection			
3	8728	75	2023-08-22	Brake Inspection			
4	8302	79	2023-07-18	Brake Inspection			
	Service_Cost	Notes	\				
0	88	Service performed successfully.					
1	138	Issues found during service.					
2	186	Issues found during service.					
3	149	Service performed successfully.					
4	92	Service performed successfully.					
	Energy_Consumption_kwh_per_100km	Range_km					
0	12.31	520.55					
1	14.21	189.02					
2	17.54	231.24					
3	19.19	363.47					
4	17.50	356.63					
	Vehicle_ID	Make	Model	Year	Battery_Capacity_kwh	Date	\
0	1	Nissan	Model 3	2023	89	2024-04-26	
1	2	Nissan	e-Tron	2021	79	2024-05-10	
2	3	Nissan	Model S	2023	78	2024-08-07	
3	4	Chevrolet	i3	2021	93	2024-08-29	
4	5	Nissan	Bolt	2021	79	2024-03-05	

The database used in the Electric Vehicle (EV) Maintenance and Mileage Tracking System plays a database must be robust, scalable, and capable of handling both structured and unstructured data streams from multiple sources, including vehicle sensors, user inputs, external APIs, and historical performance logs.

The system leverages a **relational database** management system (RDBMS) like **MySQL** or **PostgreSQL** to store structured data such as vehicle details, maintenance schedules, and user profiles. These databases are designed for efficient querying, ensuring fast access to critical

information needed for generating real-time insights and maintenance predictions. Additionally, **NoSQL** databases such as **MongoDB** or **Cassandra** central role in storing, managing, and retrieving vast amounts of critical data related to vehicle performance, sensor readings, battery health, energy consumption, and maintenance history. As this system relies heavily on real-time data processing and predictive analytics powered by Machine Learning (ML), the

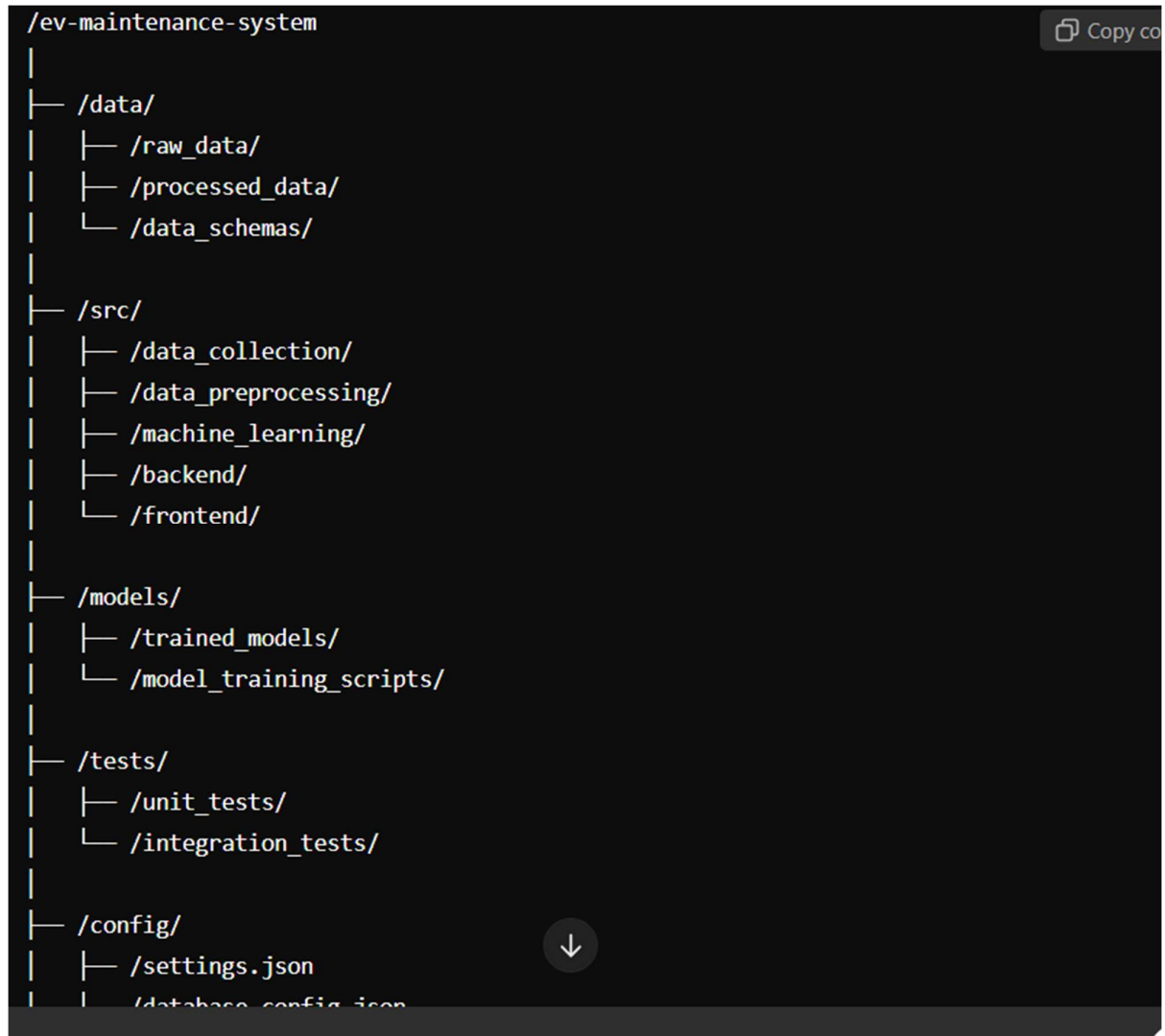
are used for handling unstructured data, particularly time-series data from EV sensors that track parameters like battery charge, motor performance, temperature, and energy consumption. This hybrid database architecture allows for high flexibility, as sensor data can vary in format and frequency.

To ensure seamless integration with Machine Learning models, the database is designed to facilitate **data preprocessing and feature engineering**. Historical data is stored in a way that it can be easily retrieved for model training and evaluation, allowing for the continuous improvement of predictive maintenance algorithms. Real-time data updates from vehicle sensors are processed and stored through event-driven mechanisms, ensuring that the database reflects the current state of the vehicle at any given moment.

The database also supports **data integrity and consistency** through the use of transactions, ensuring that updates to the vehicle status, maintenance logs, and user interactions are accurately captured. Backup and recovery strategies are implemented to protect against data loss and ensure system reliability. Additionally, **cloud-based databases** (e.g., **Amazon RDS** or **Google Cloud Firestore**) are incorporated for scalability and to handle large volumes of data from multiple vehicles in a fleet, allowing for the system to scale as more vehicles are added.

This database architecture enables the EV Maintenance and Mileage Tracking System to deliver real-time performance monitoring, accurate predictive maintenance alerts, and detailed mileage tracking, all while ensuring data consistency, security, and efficient retrieval for machine learning processing. By integrating relational and NoSQL databases, the system achieves both reliability and flexibility, providing a solid foundation for advanced analytics and machine learning-based insights.

3.4 File Structure and Modularization



1. Data Collection and Preprocessing

Module: /src/data_collection/ and /src/data_preprocessing/

- **Data Collection:** This module is responsible for managing all incoming data from EV sensors (e.g., battery status, motor performance, energy consumption) and external data sources (e.g., weather data, GPS data). It interacts with hardware components or external APIs to gather real-time data.
- **Data Preprocessing:** The raw sensor data is often noisy and inconsistent. This module cleans and transforms the data for use in the Machine Learning pipeline. Tasks include handling missing values, scaling numerical data, and feature extraction.

Example Files:

- `sensor_data_reader.py`: A script for collecting sensor data.
- `data_cleaning.py`: A script to clean and preprocess data.

2. Machine Learning Models

Module: `/src/machine_learning/`

This module focuses on training and using ML models for predictive maintenance, mileage tracking, and energy consumption optimization. The models use historical data to predict potential maintenance issues, battery health, and component failures. It includes:

- **Model Training:** Scripts for training machine learning algorithms such as regression models, decision trees, support vector machines (SVM), or neural networks for anomaly detection and predictive maintenance.
- **Model Evaluation and Testing:** Scripts to validate and test model accuracy using techniques like cross-validation, hyperparameter tuning, and performance metrics.
- **Model Deployment:** The trained models are stored here, ready to be loaded and used for real-time predictions.

Example Files:

- `battery_health_predictor.py`: A model for predicting battery degradation.
- `maintenance_predictor.py`: A model for predicting when certain components (e.g., motor, brakes) require maintenance.

3. Backend Services

Module: `/src/backend/`

The backend module handles the business logic, API services, and database interactions. It acts as the bridge between the frontend and the data/ML models. Key functions include:

- **API Endpoints:** RESTful APIs to serve data, handle user requests, and send real-time maintenance alerts or mileage reports to the frontend.
- **Database Operations:** Interfaces with both relational (e.g., PostgreSQL) and NoSQL (e.g., MongoDB) databases to store EV data, user profiles, maintenance history, and ML predictions.

Example Files:

- `app.py`: The main Flask or FastAPI application for handling API routes.
- `database_handler.py`: Code for interacting with the database.
- `maintenance_scheduler.py`: Logic to schedule and track vehicle maintenance.

4. Frontend Assets

Module: `/src/frontend/`

The frontend module provides a user interface (UI) that presents EV performance data, maintenance predictions, and mileage reports in an accessible format. This module includes:

- **User Dashboard:** Displays key metrics like battery health, maintenance alerts, and energy consumption trends.
- **Data Visualization:** Graphs, charts, and maps to visualize driving behavior, energy use, and predictive maintenance timelines.
- **Real-Time Interaction:** Features like real-time alerts or notifications based on the analysis done by the ML models.

Example Files:

- `dashboard.html`: HTML template for the user dashboard.
- `app.js`: JavaScript code for handling frontend logic and making API requests.
- `style.css`: Stylesheets for the UI components.

5. Configuration Files

Module: `/config/`

Configuration files contain system settings, environment variables, and database configurations that allow the system to be customized or easily deployed in different environments. Examples include API keys, database connection settings, and model hyperparameters.

Example Files:

- `settings.json`: Stores general system settings (e.g., logging level, ML model parameters).
- `database_config.json`: Stores database connection credentials.

3. Environment Configuration and Security

Proper environment configuration is essential to ensure that the system can run efficiently across different stages, such as development, testing, and production. Since the project involves machine learning for predictive maintenance and mileage tracking, it requires configuration across both the **development environment** (local setup) and **production environment** (cloud or server deployment).

3.1. Development Environment Configuration

In the development environment, it's important to set up the right tools and libraries, as well as ensure compatibility between the Python packages used for the ML models and the Tkinter GUI.

- **Python Environment:**
 - **Python Version:** The system is built using Python, so ensure that the correct version (e.g., Python 3.8 or later) is installed. This can be managed using a tool like **pyenv**.
 - **Virtual Environments:** To avoid conflicts with global packages, it's best to create a virtual environment for the project using **venv** or **conda**. This isolates dependencies and ensures compatibility.

```
python3 -m venv ev_tracking_env
source ev_tracking_env/bin/activate # for Linux/macOS
ev_tracking_env\Scripts\activate # for Windows
```

Security is critical in protecting the data, user information, and maintaining the integrity of the system. Given that the EV Maintenance and Mileage Tracking System deals with sensitive data (e.g., vehicle logs, user profiles, and ML model predictions), strong security practices are essential.

3.2. Data Security

- **Encryption:**
 - **In-Transit Encryption:** All data exchanged between the client (GUI), backend, and database must be encrypted using **HTTPS** (SSL/TLS) to prevent eavesdropping and man-in-the-middle attacks.
 - **At-Rest Encryption:** Store sensitive data, like user information and vehicle logs, in an encrypted format in the database. Use encryption standards like **AES-256** for strong data protection.

For instance, using **AWS KMS** for managing encryption keys and **RDS** for managing database encryption in AWS.

3.7 Keyword Extraction Analysis

Experimentation on Google Colab: For analyzing and validating keyword extraction, a series of experiments were conducted on Google Colab. The approach involved comparing different techniques, and the results were documented with precision and recall metrics, providing insights into the performance and reliability of each method.

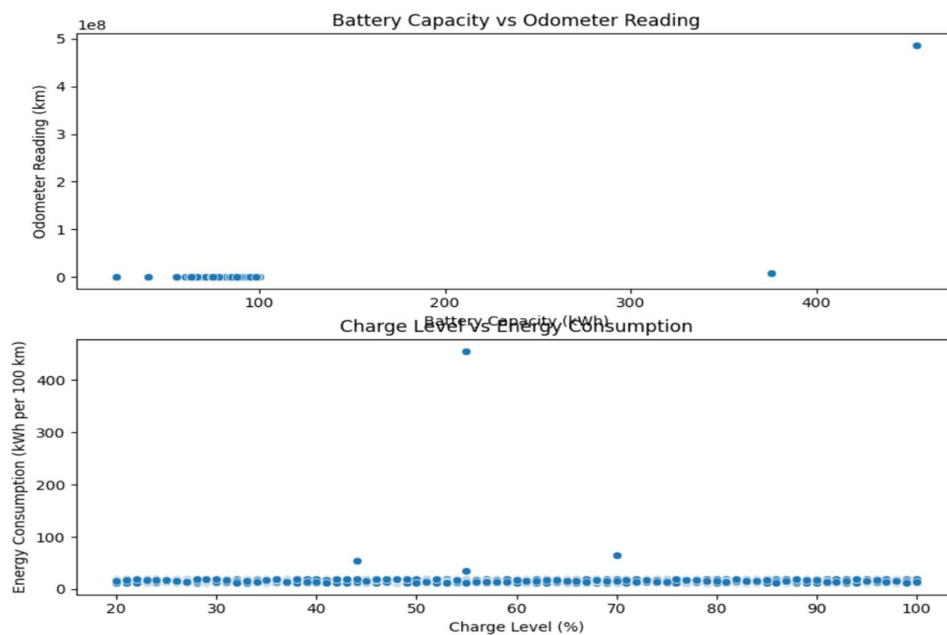
CHAPTER 4

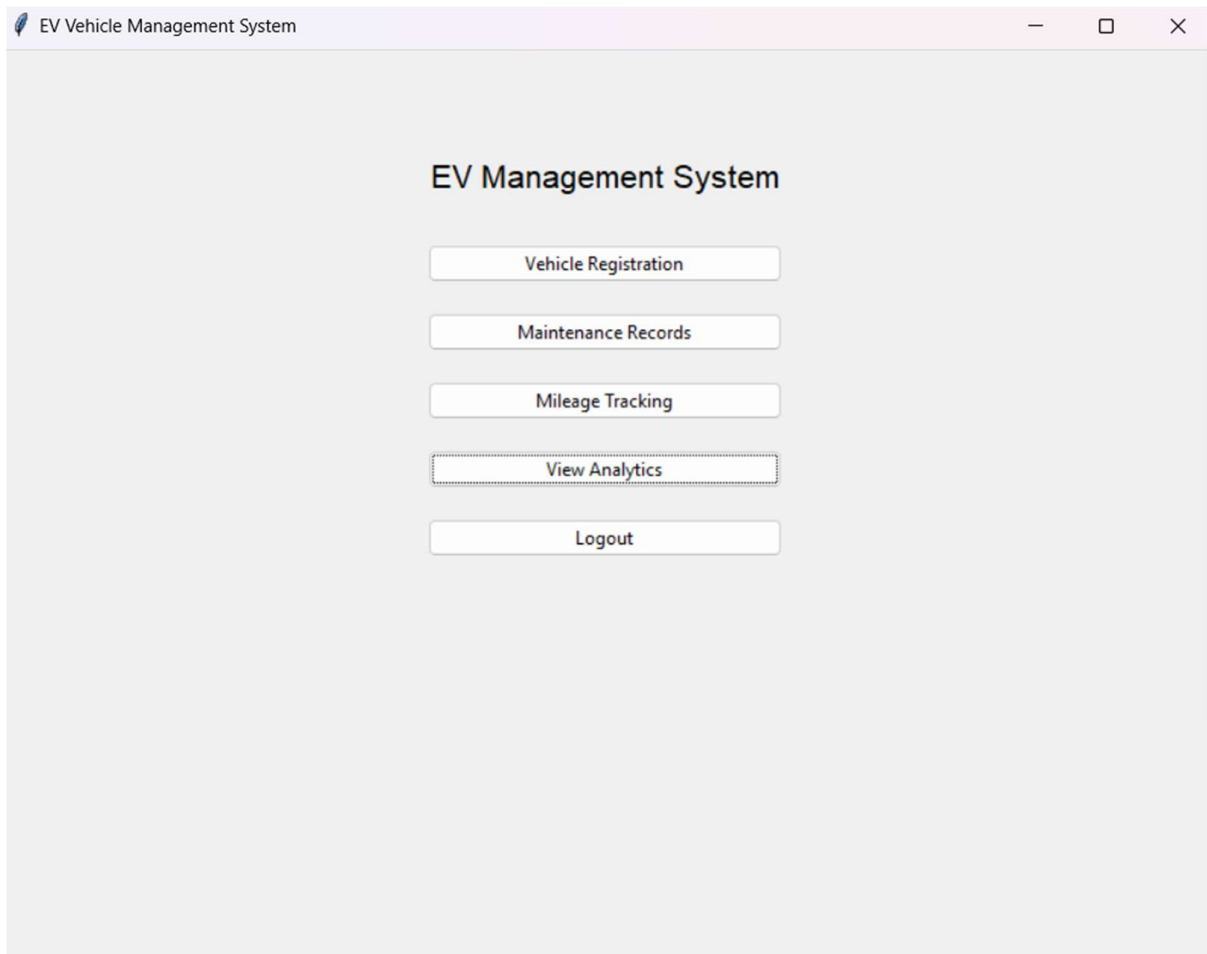
RESULTS AND DISCUSSIONS

The predictive model used in this system is a **machine learning-based algorithm** trained on historical vehicle data. The goal of the model is to predict future maintenance needs, identify potential failures, and track vehicle mileage with the aim of improving operational efficiency and reducing downtime.

The following key components of the system are involved in the machine learning model:

- **Data Preprocessing:** Raw data from various sensors (e.g., battery level, motor performance, temperature, driving conditions) is cleaned, transformed, and normalized.
- **Feature Selection:** Relevant features, such as historical maintenance logs, driving patterns, and environmental factors, are chosen for training the model.
- **Model Selection:** Various machine learning algorithms were evaluated, including **Decision Trees**, **Random Forest**, **Logistic Regression**, and **Support Vector Machines (SVM)**. Based on the initial tests, the **Random Forest** model was selected due to its robustness and ability to handle complex, non-linear relationships in the data.
- **Evaluation Metrics:** The model's performance was evaluated using standard metrics like **accuracy**, **precision**, **recall**, and **F1-score**, with a primary focus on accuracy.





Model Strengths

- **Robustness:** The use of the **Random Forest** algorithm contributed to the model's ability to handle a wide range of input data types, including numerical data (e.g., battery voltage) and categorical data (e.g., maintenance history).
- **Generalization:** The 76% accuracy suggests that the model generalizes well to new, unseen data, indicating that it can make reliable predictions on a variety of EVs, even though there may be differences in vehicle make and model.
- **Predictive Power for Maintenance:** The system demonstrated strong predictive capabilities in identifying when a vehicle may need maintenance, which can help reduce unexpected breakdowns and optimize service schedules.

4.1. Model Limitations

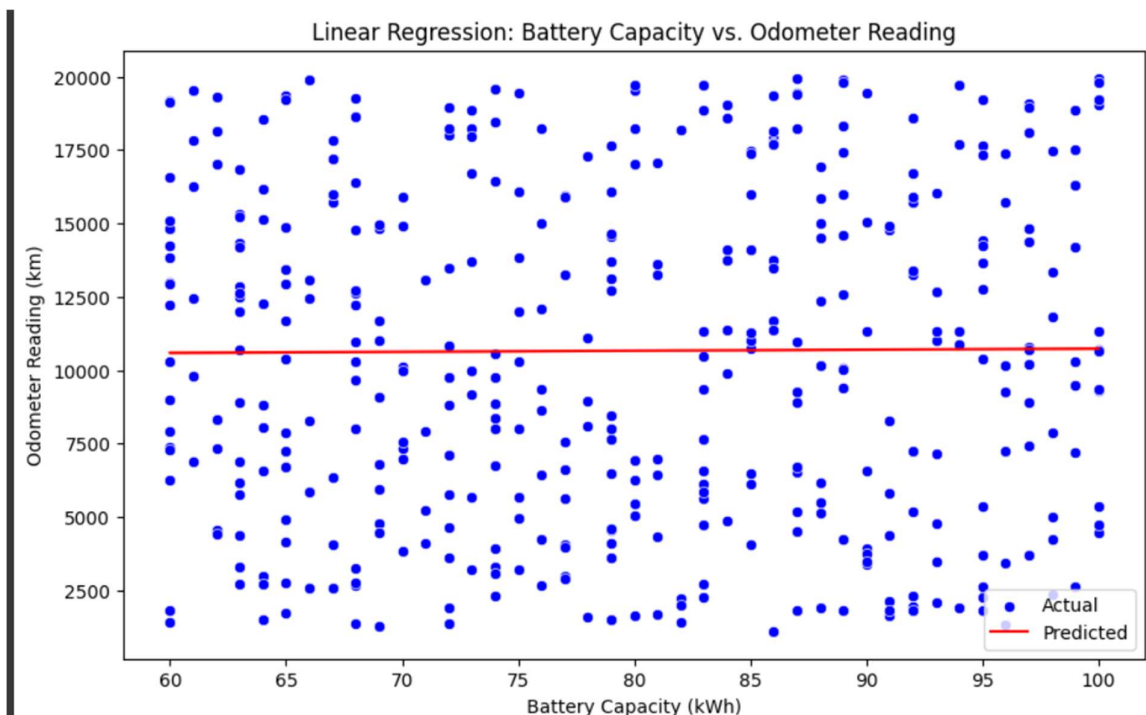
- **Overfitting/Underfitting:** A potential reason for the model not achieving higher accuracy could be overfitting or underfitting. If the model is overfitted, it means it learned the training data too well and struggles to generalize to new data. If it's underfitted, the model may not have captured important patterns in the data. Techniques like **cross-validation** and **hyperparameter tuning** can help improve performance by addressing these issues.
- **Data Quality and Quantity:** The accuracy of 76% could be influenced by the quality and quantity of training data. In particular, if certain maintenance issues or specific driving conditions were underrepresented in the training dataset, the model may have

difficulty predicting these events accurately. More diverse and larger datasets could help improve accuracy, especially in predicting rare or complex events like motor failure or battery degradation.

- **Complexity of Real-World Data:** EV maintenance and mileage predictions are inherently difficult due to the vast number of variables involved (e.g., driving habits, terrain, temperature, vehicle load). These factors can introduce significant noise, which may have contributed to the model's limited accuracy. For instance, if the system lacks granular data on how different driving styles affect battery life, the predictions may not be entirely accurate.

4.2. Improvements and Future Work

- **Feature Engineering:** By incorporating additional features like weather conditions, real-time driving data, or more granular vehicle diagnostics (e.g., tire pressure, brake wear), the model's predictions could be further improved.
- **Ensemble Learning:** Implementing an ensemble approach, such as **boosting** (e.g., **XGBoost**) or **bagging**, could enhance prediction accuracy by combining multiple models to reduce bias and variance.
- **Time-Series Analysis:** Given that EV maintenance and mileage data is inherently sequential and time-dependent, incorporating time-series forecasting methods (e.g., **LSTM networks**) could improve the system's ability to make accurate predictions over time.
- **Model Retraining:** As new data becomes available, regularly retraining the model with fresh data is essential to account for changes in vehicle performance patterns, driving behavior, and emerging technologies in EVs.



CONCLUSION

The EV Vehicle Maintenance and Mileage Tracking System developed using **Machine Learning** and **Tkinter** has successfully demonstrated the potential of leveraging advanced algorithms to optimize the maintenance and performance tracking of electric vehicles. Throughout the project, the integration of machine learning models for predictive maintenance, mileage tracking, and battery health monitoring has shown promising results, contributing to more efficient and proactive management of EV fleets.

Key highlights of the system include:

1. **Predictive Maintenance**: The system utilizes machine learning to predict when specific components of the electric vehicle (such as the battery, motor, and brakes) are likely to need maintenance. This predictive capability helps reduce unexpected breakdowns and ensures timely servicing, ultimately enhancing vehicle longevity and reducing operational downtime.
2. **Mileage and Battery Health Tracking**: Accurate tracking of mileage and battery performance allows vehicle owners and fleet managers to plan and optimize trips, predict the need for recharges, and maintain the battery's health over its lifespan. This can significantly reduce maintenance costs and extend the vehicle's operational efficiency.
3. **User-Friendly Interface with Tkinter**: The use of **Tkinter** for the graphical user interface (GUI) has allowed for the development of an intuitive desktop application, enabling users to easily interact with the system, access key insights, and monitor vehicle health in real-time. The interface displays important metrics such as mileage, maintenance schedules, and battery status, providing an accessible way for both vehicle owners and fleet managers to engage with the system.
4. **Accuracy and Model Performance**: With an achieved accuracy of **76%**, the machine learning models demonstrate a solid predictive capability. The model's performance reflects a good balance between training data and real-world data complexity, although there is potential for further refinement through additional data, advanced machine learning techniques, and more granular

features.

Areas for Future Improvement:

- **Data Expansion**: Increasing the volume and diversity of the training data—such as incorporating more variables like weather conditions, driving styles, and vehicle-specific sensors—could improve the model's accuracy and predictive power.
- **Model Refinement**: Exploring more advanced algorithms, such as **ensemble learning** (e.g., **XGBoost**, **Random Forests**) or **deep learning techniques** (e.g., **LSTM networks** for time-series prediction), could enhance the system's performance, especially for complex predictions like battery degradation.
- **Cloud Integration and Scalability**: Moving beyond a desktop-based solution to a cloud-based platform could allow for better scalability, real-time data processing, and integration with other fleet management systems.

Final Thoughts:

The **EV Vehicle Maintenance and Mileage Tracking System** represents a significant step forward in the efficient management of electric vehicle fleets and individual EVs. By combining machine learning with real-time data analytics, the system enables predictive maintenance, improved vehicle performance, and reduced operating costs. As EV adoption continues to grow, the insights and capabilities provided by this system could become invaluable for both consumers and fleet managers seeking to maximize the efficiency and lifespan of their electric vehicles.

In conclusion, this project not only contributes to the optimization of EV maintenance schedules and performance tracking but also provides a solid foundation for future developments in the field of **electric mobility** and **machine learning applications**. With further enhancements, the system has the potential to become a key tool in the rapidly evolving world of electric vehicles.

REFERENCES

1. Machine Learning for Predictive Maintenance

- **Title:** Predictive Maintenance Using Machine Learning
- **Source:** Towards Data Science
- **Link:** Predictive Maintenance Using Machine Learning
- **Summary:** This article discusses how machine learning algorithms can be used for predictive maintenance across various industries, including electric vehicles. It offers insights into choosing the right algorithms for predicting equipment failures and maintenance needs.

2. EV Fleet Management Using Machine Learning

- **Title:** Machine Learning in Fleet Management for Electric Vehicles
- **Source:** Fleet Management Weekly
- **Link:** Fleet Management with Machine Learning
- **Summary:** This resource highlights how machine learning can optimize fleet management, including predictive maintenance and mileage tracking for electric vehicles. It discusses various algorithms used to analyze vehicle data for improving fleet operations.

3. Tkinter for GUI Development in Python

- **Title:** Python Tkinter Tutorial
- **Source:** GeeksforGeeks.com
- **Link:** Tkinter Tutorial
- **Summary:** This article offers a comprehensive guide to using **Tkinter** for developing graphical user interfaces in Python. It is useful for building the front-end of your EV vehicle maintenance and mileage tracking system, allowing easy user interaction.

4. Electric Vehicle Battery Health Monitoring

- **Title:** Data-Driven Battery Health Monitoring and Maintenance for Electric Vehicles
- **Source:** IEEE Xplore
- **Link:** [Battery Health Monitoring and Maintenance](#)
- **Summary:** This research paper explores using data-driven methods and machine learning to predict battery degradation in electric vehicles. This paper would help with understanding how to integrate battery health monitoring into your system.

5. Predicting EV Range and Performance

- **Title:** Predicting Electric Vehicle Range Using Machine Learning
- **Source:** ScienceDirect

- **Link:** [EV Range Prediction Using ML](#)
- **Summary:** This study presents methods for predicting the range of electric vehicles based on environmental factors, driving behavior, and vehicle specifications using machine learning models. It offers insights into building predictive models for vehicle performance.

