Real Time Scheduling of an Automotive Maintenance

P Shiva Naga Raju[1]; P Nikhitha[2]

[1,2]Department of Mechanical Engineering, University College of Engineering, Osmania University, Hyderabad, Telangana, India.

Publication Date: 2025/07/19

Abstract: This electronic document represents a smart, proactive vehicle maintenance predictor system designed to transform the traditional ownership experience by optimizing performance, reducing downtime, and enhancing safety. By integrating advanced data analytics, machine learning, and IoT technology, the system continuously monitors critical vehicle parameters such as engine oil and filter, air cleaner filter, fuel filter, coolant, and more. The goal is to bring a data-driven, user-centric approach to vehicle maintenance and performance monitoring. Real-time data is collected via sensors and visualized through an intuitive mobile or interactive web application, which also issues alerts for issues like overheating or low oil levels. The system applies predictive maintenance techniques using historical data to forecast potential problems and schedule service tasks based on usage patterns and manufacturer guidelines. It maintains a log of previous services and sends automated reminders for upcoming maintenance. Additionally, the integration of telematics enables tracking of driving behavior to promote eco-friendly habits and record fuel efficiency and trip history. A simulation model was built using Python libraries and the Twilio API to demonstrate the concept. It tracks parameters like speed, fuel level, and gear status, triggering maintenance alerts—such as engine oil changes every 2000 km and gear oil changes every 6000 km—along with real-time notifications. The system effectively showcases the use of predictive analytics and real-time communication to ensure timely maintenance, improve reliability, and lower long-term vehicle repair costs. With automated reminders, comprehensive maintenance logs, and intelligent analysis of driving behavior, it supports better decision-making for vehicle owners. The integration with telematics not only enhances maintenance precision but also encourages eco-friendly driving by analyzing acceleration, braking, and speed patterns. Overall, the project demonstrates a scalable and impactful solution for smart vehicle management.

Keywords: *Predictive Maintenance, IoT, Telematics, Machine Learning, RealTime Monitoring, Streamlit, Maintenance Scheduling.*

How to Cite: P Shiva Naga Raju; P Nikhitha (2025), Real Time Scheduling of an Automotive Maintenance. *International Journal of Innovative Science and Research Technology*, 10(7), 1295-1308. https://doi.org/10.38124/ijisrt/25jul735

I. INTRODUCTION

A. Paradigm Shift in Automotive Maintenance:

In today's fast-paced world, vehicle downtime translates directly to economic losses for businesses and inconvenience for individuals. Traditional maintenance approaches, often based on fixed intervals, fail to account for the unique operating conditions and wear patterns of each vehicle. This leads to either over-maintenance, incurring unnecessary costs, or under-maintenance, risking unexpected breakdowns and costly repairs.

The "Real time Scheduling of an Automotive Maintenance" project seeks to address these challenges by implementing a proactive and data-driven approach to maintenance scheduling. By harnessing the power of IoT, machine learning, and real-time data analytics, this system aims to optimize maintenance intervals, minimize downtime, and maximize vehicle lifespan.

The Vehicle Maintenance Predictor is an interactive web application built with Streamlit that helps users predict when their SUV will need maintenance based on their driving patterns and conditions.

The application uses machine learning to provide personalized maintenance recommendations by analysing factors such as:

- Average speed
- Daily driving duration
- Number of driving days per month
- Gear usage patterns
- Road type (City, Highway, Rural, etc.)
- Weather conditions
- Traffic conditions
- Vehicle load

The result will be a dynamic and intelligent maintenance system that empowers vehicle owners and fleet managers to optimize their maintenance budgets, enhance vehicle reliability, and improve overall operational efficiency. This system will not only reduce maintenance costs but also enhance safety and minimize environmental impact by optimizing vehicle performance and reducing emissions.

B. Key Improvements:

➢ *Stronger Emphasis on Problem:*

Highlighted the economic and operational impacts of traditional maintenance methods.

➢ *Clearer Articulation of the Solution:*

Emphasized the use of advanced technologies like IoT, machine learning, and real-time data analytics.

➢ *Enhanced Focus on Benefits:*

Emphasized the benefits beyond cost reduction, such as improved safety, environmental impact, and operational efficiency.

This study includes hardware capable of interfacing with the vehicle's ECU to gather real-time data, IoT solutions to establish seamless connectivity with the cloud, and leveraging cloud computing platforms such as AWS or Microsoft Azure for advanced data analysis. The data analysis pipeline would incorporate machine learning and AI modules to ensure high accuracy and real-time processing. Additionally, user-friendly iOS and Android applications would serve as the interface for users, complemented by robust notification services for timely updates and alerts.

This can be successful through the development of a holistic, end-to-end ecosystem. Achieving this requires a strategically structured, multi-layered approach:

- *Data Acquisition*

✓ *Hardware Integration:*

Efficient integration of hardware components with the vehicle's Electronic Control Unit (ECU) is essential. These components serve as a robust data gateway, facilitating the seamless collection of real-time data streams from a wide range of vehicle sensors.

The hardware must support diverse sensor types to ensure comprehensive monitoring of parameters such as engine performance, tire pressure, fuel consumption, and more.

✓ *Optimal Sensor Selection:*

Strategic selection of sensors is critical to guarantee precise and reliable data collection. The choice of sensors should align with the specific monitoring objectives, encompassing parameters like temperature, humidity, vibration, pressure, and other key metrics.

Advanced sensors with high accuracy and durability should be prioritized to ensure the integrity and reliability of the data, particularly under varying environmental and operational conditions.

- *Data Connectivity:*

✓ *Hardware Integration:*

Harnessing advanced IoT solutions is pivotal for enabling seamless and secure connectivity between the vehicle's hardware and the cloud platform.

This involves selecting the most suitable communication protocols, such as cellular networks for wide-area coverage, Wi-Fi for high-speed local connectivity, or Bluetooth for short-range, low-power communication.

Ensuring consistent and reliable data transmission is key, necessitating redundancy mechanisms and fail-safe designs to handle potential disruptions.

Integration with IoT platforms should also prioritize scalability, allowing the system to accommodate increasing data volumes and additional connected devices as the ecosystem grows.

- *Data Processing and Analysis:*

✓ *Cloud Computing:*

Leveraging powerful cloud computing platforms such as AWS, Azure, or Google Cloud ensures the scalability, flexibility, and computational capacity required to manage and analyze the immense data streams generated by modern vehicles.

These platforms provide robust infrastructure for storing, processing, and visualizing data, enabling seamless integration with other components of the monitoring and scheduling ecosystem

✓ *AI/ML Integration:*

Embedding cutting-edge machine learning (ML) and artificial intelligence (AI) algorithms into the cloud-based data pipeline unlocks transformative capabilities, including:

▪ *Predictive Maintenance:*

Utilizing historical data and real-time sensor readings to predict and pre-empt potential failures, reducing downtime and maintenance costs.

▪ *Anomaly Detection:*

Detecting irregularities or deviations in vehicle performance, allowing for early identification of potential issues before they escalate.

▪ *Driving Behavior Analysis:*

Examining driving patterns to optimize fuel efficiency, enhance safety, and encourage better driving habits. AI/ML models continuously improve through feedback loops, ensuring greater accuracy and adaptability over time.

- *User Interface and Interaction*

✓ *Intuitive Applications:*

Designing highly responsive and user-friendly mobile applications for iOS and Android platforms ensures seamless access to vehicle data, maintenance schedules, and real-time alerts.

These applications should prioritize an intuitive layout, customizable dashboards, and easy navigation to enhance user engagement and satisfaction.

✓ *Dynamic Notification System:*

Deploying a robust and adaptable notification framework guarantees timely updates on vehicle health, maintenance requirements, and potential issues.

Notifications can be delivered through multiple channels, including push notifications, SMS, and email, to suit user preferences.

- *Data Security and Privacy:*

✓ *Comprehensive Security Framework:*

Establishing a multi-layered security framework is essential to safeguard sensitive vehicle and user data from unauthorized access and potential cyber threats. The Key measures include:

- Data Encryption
- Access Controls
- Regular Security Audits

✓ *Privacy-Centric Design:*

Adopting privacy-by-design principles ensures user data is collected, processed, and stored in compliance with regulatory standards like GDPR or CCPA.

Features such as data anonymization, user consent mechanisms, and transparent data usage policies build trust and enhance user confidence in the system.

This approach involves integrating various systems through seamless communication between hardware components. Real-time data streams, including engine RPM, speed (in km/h), temperature, and past maintenance/service records, are collected from the vehicle. This data is then analyzed in conjunction with pre-determined or imposed standard data sets.

The analysis, potentially utilizing machine learning algorithms, identifies potential anomalies or deviations from expected vehicle behavior. Based on these insights, the system generates real-time alerts and notifications through a user-friendly interface (e.g., mobile app, dashboard), informing the driver or vehicle owner about potential issues or the need for upcoming maintenance.

Developing a complete ecosystem for [specific purpose, e.g., real-time vehicle monitoring] can be resource-intensive, requiring significant time, energy, and financial investment. To efficiently validate the core concepts and feasibility, a proof-of-concept (POC) can be developed using software tools deployed on PCs or laptops.

This approach allows for rapid prototyping and experimentation while minimizing initial costs. The POC will serve as a foundational step to demonstrate the core functionalities, test key assumptions, and gather valuable insights before investing heavily in a full-scale implementation.

II. METHODOLOGY

This research focuses on designing and implementing an advanced real-time vehicle data monitoring and Maintenance Predictor, an interactive web application built with Streamlit that helps users predict when their SUV will need maintenance based on their driving patterns and conditions. The application uses machine learning to provide personalized maintenance recommendations by analyzing factors such as: Average speed, Daily driving duration, Number of driving days per month, Gear usage patterns etc.

The outlined objectives can be effectively realized by developing a comprehensive end-to-end ecosystem. This includes hardware capable of interfacing with the vehicle's ECU to gather real-time data, IoT solutions to establish seamless connectivity with the cloud, and leveraging cloud computing platforms such as AWS or Microsoft Azure for advanced data analysis. The data analysis pipeline would incorporate machine learning and AI modules to ensure high accuracy and real-time processing. Additionally, user-friendly iOS and Android applications would serve as the interface for users, complemented by robust notification services for timely updates and alerts. This necessitates a multi-layered approach.

This approach consists of the following subsystems.

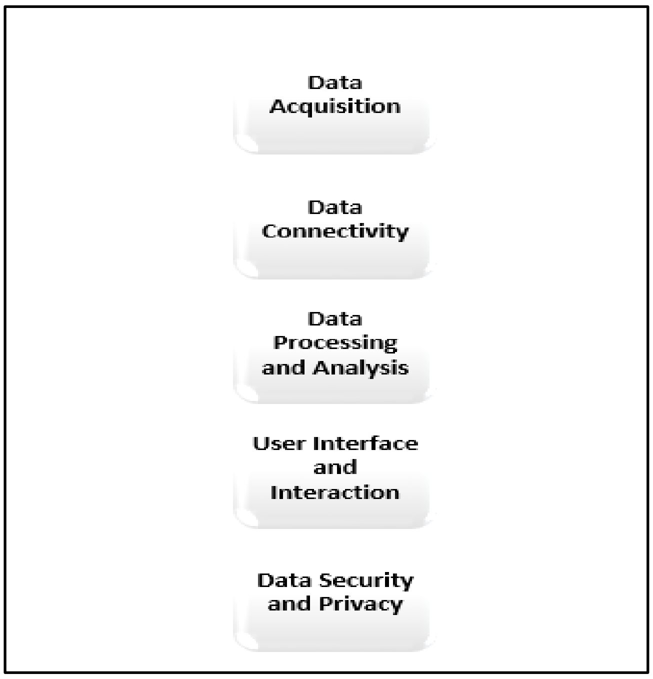

Data Acquisition
Data Connectivity
Data Processing and Analysis
User Interface and Interaction
Data Security and Privacy

Fig 1 Methodology

III. BACKEND SYSTEM TO MONITOR AND SCHEDULE AN AUTOMOTIVE MAINTANANCE-APPLICATION

This method focuses on the integration of multiple systems through efficient communication among hardware components. Real-time data, such as engine RPM, vehicle speed (km/h), temperature, and historical maintenance or service records, is gathered directly from the vehicle. The collected data is then analyzed alongside predefined or standardized datasets to derive meaningful insights.

Building a comprehensive ecosystem for [specific purpose, e.g., real-time vehicle monitoring] is a demanding endeavor, necessitating substantial commitments of time, effort, and financial resources.

To effectively validate core concepts and assess feasibility, a proof-of-concept (POC) can be developed utilizing software tools deployed on PCs or laptops. This strategy enables rapid prototyping and iterative experimentation while keeping initial costs low. The POC acts as a foundational step, showcasing essential functionalities, testing critical assumptions, and providing valuable insights that inform further development before committing to a full-scale implementation. The POC involves the below mentioned sub system in order to achieve outlined objectives, testing critical assumptions, and providing valuable insights that inform further development before committing to a full-scale implementation. The POC involves the below mentioned sub system in order to achieve outlined objectives.

This approach allows for rapid prototyping and experimentation while minimizing initial costs. The POC will serve as a foundational step to demonstrate the core functionalities, test key assumptions, and gather valuable insights before investing heavily in a full-scale implementation.

A Python-based application server, hosted on a PC or laptop, will serve as the core data processing hub. It will continuously receive simulated vehicle parameters, including speed (in km/h), temperature, historical maintenance records, etc. This data will then be analyzed and processed to identify potential issues and generate relevant insights.

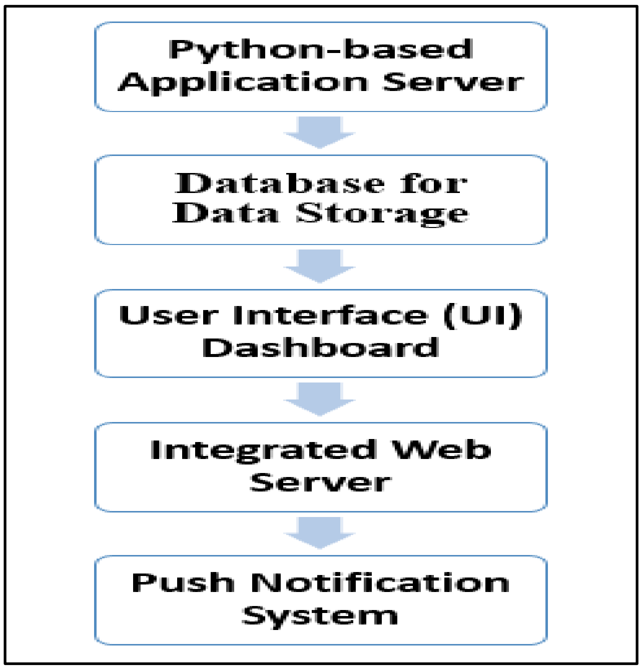

Python-based Application Server
Database for Data Storage
User Interface (UI) Dashboard
Integrated Web Server
Push Notification System

Fig 2 Backend System to Monitor and Schedule an Automotive Maintanance-Application

This application consists of the following components:

A. Main Application (app.py) – Streamlit Web Interface

➢ *Purpose:*

Essentially, app.py serves as the central hub through which users like vehicle owners or fleet managers directly interact with your predictive maintenance system. Think of it as the cockpit of the project, providing a clear and intuitive view into the health and future needs of their vehicles. Its primary purpose is to translate the complex data and predictions generated in the back-end into an understandable and actionable format for the user. Instead of raw sensor readings or cryptic algorithm outputs, app.py presents this information in a visually appealing and easily digestible way through its Streamlit web interface.

Furthermore, app.py isn't just about displaying information; it also empowers users with a degree of control and personalization. By allowing them to input vehicle details, service history, and driving preferences, the application can tailor its predictions and recommendations more accurately to their specific context.

Finally, app.py acts as a communication bridge, delivering timely alerts and notifications about potential issues or upcoming maintenance needs. It also facilitates the management of service records and even offers guidance on eco-friendly driving, enhancing the overall user experience and the value of your predictive maintenance system. In short, it's the face and voice of your project, making its powerful capabilities accessible and useful to the end-user.

B. Maintenance Predictor (maintenance_predictor.py) Predictive Logic:

At its core, maintenance_predictor.py is the brain of your predictive maintenance system. Its fundamental purpose is to analyze data and determine when a vehicle component is likely to require maintenance in the future. It moves beyond simple reactive maintenance by proactively identifying potential issues before they lead to breakdowns or costly repairs.

➢ *Predictive Intelligence Engine:*

This script houses the crucial logic for forecasting maintenance needs. It takes in various forms of data – both historical (past maintenance records, failure data) and real-time (sensor readings, driving behavior) – and processes it using the loaded machine learning model to generate predictions. This allows the system to anticipate problems rather than just reacting to them.

➢ *Decision-Making Hub:*

maintenance_predictor.py doesn't solely rely on the ML model. It intelligently combines the model's probabilistic predictions with established, rule-based checks. For example, even if the ML model doesn't flag an immediate oil change, the system will still recommend one based on a fixed mileage interval (like every 2000 km). This hybrid approach adds a layer of reliability and incorporates industry best practices.

➢ *Personalization and Adaptation Layer:*

A key purpose is to tailor maintenance recommendations to specific circumstances. By considering the SUV brand/model (which influences expected wear and tear patterns) and individual user driving habits (e.g., frequent hard braking, long idling times), the script ensures that the predictions and alerts are more relevant and accurate for each user. This moves away from a one-size-fits-all approach to maintenance.

➢ *Insight Generation:*

Raw sensor data can be meaningless to the average user. Therefore, maintenance_predictor.py plays a vital role in interpreting these raw values and translating them into clear, actionable insights. Instead of just reporting a coolant temperature of 110°C, it converts this into a user-friendly message like "Coolant temperature is above the optimal range," along with potential implications or recommended actions.

In essence, maintenance_predictor.py is the engine that drives the predictive capabilities of your system. It takes in data, applies sophisticated logic (both learned and rule-based), personalizes the analysis, and generates meaningful insights that empower users to proactively manage their vehicle's maintenance needs.

C. Data Generator (data_generator.py) – Simulated Telematics Feed

The fundamental purpose of data_generator.py is to provide a controlled and flexible source of simulated real-time vehicle data. In situations connecting to actual vehicle telematics or live sensor feeds is impractical (like during the initial development, testing, or demonstration phases of your project), this script steps in to mimic the continuous stream of information that a real vehicle would produce. It acts as a virtual vehicle, allowing you to build, test, and showcase your predictive maintenance system without needing a physical fleet of connected cars.

➢ *Working:*

• *Creating Realistic Synthetic Data:*

The script employs methods to generate data points for various vehicle parameters that are statistically plausible. This can range from simple random number generation within realistic ranges (e.g., speed between 0 and 180 km/h) to more sophisticated algorithms that introduce correlations between parameters (e.g., higher RPM generally corresponds to higher fuel consumption and potentially increasing engine temperature). This ensures that the simulated data behaves in a somewhat lifelike manner.

• *Simulating Temporal Dynamics:*

data_generator.py can go beyond just generating single data points. It's designed to simulate vehicle operation *over time*. This means it can produce a sequence of data points that represent a driving session, including periods of acceleration, cruising at a steady speed, idling at traffic lights, and eventually shutting down the engine. This temporal aspect is crucial for testing how your predictive models react to trends and changes in data over time.

• *Enabling Custom Test Scenarios:*

A significant advantage of a data generator is its ability to reproduce specific, often rare, events or edge cases on demand. Instead of waiting for a real vehicle to experience a sudden oil leak or a rapid overheating event, you can program data_generator.py to simulate these scenarios directly. This allows you to rigorously test the robustness and responsiveness of your predictive maintenance system to unusual or critical situations and ensure it triggers the correct alerts and recommendations.

In essence, data_generator.py is a vital tool for decoupling the development and testing of your system from the availability of real-world vehicle data. It provides a reliable, controllable, and customizable data source that accelerates the development process, facilitates thorough testing of various scenarios (including edge cases), and enables effective demonstrations of your project's capabilities. It's like having a virtual test fleet at your fingertips.

D. Utilities (utils.py) – Helper Functions:

The primary purpose of utils.py is to serve as a central repository for reusable code and backend functionalities that are needed across different parts of your predictive maintenance system (app.py, maintenance_predictor.py, and potentially others). It promotes code organization, reduces redundancy, and makes your codebase more maintainable and efficient by housing common tasks in one place. Think of it as the toolkit that provides essential tools and services to all the other modules.

This utils.py acts as the glue that binds the different parts of your system together by providing essential shared functionalities. It promotes code reuse, ensures data consistency, handles external integrations, and manages crucial temporal information, ultimately contributing to a more organized, efficient, and robust predictive maintenance system.

E. Maintenance Data (maintenance_data.py) – Brand-Specific Schedules:

The primary purpose of maintenance_data.py is to serve as a structured repository for manufacturer-recommended maintenance schedules that are specific to different vehicle brands and models. Think of it as a digital library of official servicing guidelines. This module provides a baseline or a set of default expectations for when certain maintenance tasks should typically be performed based on factors like mileage or time intervals, as advised by the vehicle manufacturers themselves. Here's a more detailed breakdown of its purpose:

➢ *Providing Foundational Maintenance Information:*

This script acts as a source of reliable, industry-standard maintenance guidelines. Instead of relying solely on the machine learning model's predictions (which are based on usage patterns), it incorporates the expert knowledge and recommendations of the vehicle manufacturers. This ensures that essential maintenance tasks, regardless of driving style, are considered.

➢ *Enabling Brand and Model Customization:*

By storing schedules based on specific brands and models, maintenance_data.py allows the system to tailor maintenance recommendations right from the start. Different manufacturers have varying service intervals for components like oil changes, filter replacements, and fluid flushes. This module ensures that the system accounts for these inherent differences.

➢ *Supporting Rule-Based Checks:*

As mentioned earlier in the context of maintenance_predictor.py, the system likely combines ML predictions with rule-based checks. maintenance_data.py provides the data for these rule-based checks, such as "change oil every 12 months or 15,000 kilometres, whichever comes first" for a specific vehicle model.

➢ *Serving as a Reference Point*:

This module can act as a reference point for users to compare the system's predictions against the manufacturer's recommendations. It can also be used to initialize or inform the machine learning models about typical maintenance patterns.

➢ *Facilitating System Updates:*

When manufacturers release updated maintenance schedules for their vehicles, maintenance_data.py can be updated accordingly, ensuring that the system's baseline recommendations remain current and accurate.

In essence, maintenance_data.py grounds your predictive maintenance system in the established best practices of vehicle manufacturers. It provides a crucial layer of brand and model-specific information that complements the dynamic predictions generated by the machine learning models, leading to more comprehensive and reliable maintenance recommendations. It ensures that essential maintenance tasks are not overlooked and that recommendations align with industry standards.

F. Machine Learning Model – Predictive Engine:

The fundamental purpose of this component is to act as the intelligent core of your predictive maintenance system. It moves beyond static rules and leverages the power of data to learn patterns and relationships within historical vehicle data. This learning enables it to forecast future maintenance needs in a dynamic and data-driven way, offering significant advantages over traditional, schedule-based maintenance. A deeper dive into its purpose based on the workflow you've described:

➢ *Data-Driven Forecasting:*

The primary aim is to predict when specific vehicle components are likely to require maintenance or fail. Instead of simply adhering to fixed intervals, the ML model analyses a multitude of factors – such as accumulated mileage, patterns in how the vehicle is driven (e.g., aggressive acceleration, frequent hard braking), fluctuations in environmental conditions (like extreme temperatures), and the history of past servicing – to identify subtle indicators of impending issues. This allows for more accurate and timely maintenance recommendations.

➢ *Proactive Maintenance Planning:*

By predicting future failures or degradation, the model enables proactive maintenance planning. This means users can be alerted to potential problems *before* they lead to breakdowns or safety concerns, allowing them to schedule maintenance at a convenient time and potentially prevent more costly repairs down the line. This minimizes vehicle downtime and enhances overall vehicle reliability.

➢ *Personalized Maintenance Schedules:*

Because the model learns from the specific data of vehicles (or similar vehicles), it can contribute to more personalized maintenance schedules. Instead of a generic recommendation, the model can suggest an oil change sooner for a vehicle that consistently experiences high engine temperatures or predict brake wear based on the driver's braking habits.

➢ *Continuous Improvement:*

The "Adaptability" aspect highlights a key purpose: ongoing improvement of prediction accuracy. As the system collects more data from the vehicles it monitors, this new information can be used to retrain or fine-tune the machine learning model. This continuous learning loop allows the model to adapt to evolving usage patterns, environmental conditions, and even vehicle-specific characteristics, leading to increasingly precise and reliable predictions over time.

In essence, the Machine Learning Model – Predictive Engine is the brainpower behind your system's ability to anticipate maintenance needs. It transforms raw data into actionable insights, enabling a shift from reactive repairs to proactive prevention. By learning from the past, it empowers users to make informed decisions about their vehicle's upkeep, ultimately leading to cost savings, reduced downtime, and improved safety. It's the engine that drives the "predictive" aspect of your predictive maintenance system.

In General

- *Main Application (app.py):*

✓ Interactive Stream lit dashboard for real-time vehicle monitoring.
✓ Displays predictions, alerts, trip history, and driving behaviour.
✓ Allows manual input of service logs and vehicle selection.
✓ Integrates all backend modules into a user-friendly interface.

- *Maintenance Predictor (maintenance_predictor.py):*

✓ Contains logic for predicting maintenance needs using ML models.
✓ Analyses sensor data and driving patterns to forecast service tasks.
✓ Combines fixed intervals with dynamic, data-driven alerts.
✓ Customizes predictions based on vehicle brand and usage.

- *Data Generator (data_generator.py):*

✓ Simulates live vehicle sensor data for development and testing.
✓ Generates values like speed, fuel level, and engine temperature.
✓ Enables testing of system responses to different scenarios.
✓ Supports time-based simulations and stress testing.

- *Utilities (utils.py):*

✓ Provides helper functions for model loading, notifications, etc.
✓ Manages WhatsApp alerts via Twilio API.
✓ Handles data formatting, unit conversion, and time stamps.
✓ Keeps codebase modular and clean with reusable functions.

- *Maintenance Data (maintenance_data.py):*

✓ Stores brand-specific service intervals for SUV models.
✓ Provides a reference for comparing predicted vs recommended tasks.
✓ Easily extendable for more vehicles or updated guidelines.
✓ Used by the predictor module for customized alerting.

- *Machine Learning Model:*

✓ Trained on historical and simulated vehicle maintenance data.
✓ Predicts component wear and service needs based on usage.
✓ Supports personalized maintenance over fixed schedules.
✓ Enhances reliability and reduces unnecessary servicing.

A. Prerequisites:

To run this application, we need:

➢ *Python 3.7 or Higher:*

Python runs all the code — the user interface, the data processing, and the machine learning predictions.

➢ *Pip (Python Package Installer):*

pip is a tool that lets you install all the Python libraries the app needs to work. This project uses external Python libraries like: streamlit (for the web interface), scikit-learn (for machine learning), twilio (for sending WhatsApp alerts).

➢ *Internet Connection:*

An internet connection is required for two main reasons:

- *Lottie Animations:*

The app uses animations (like loading spinners or alert icons) from the internet to make the interface more engaging. These are fetched live from the web.

- *Twilio WhatsApp Alerts:*

The app sends maintenance reminders or alerts via WhatsApp using Twilio's online API. This only works when you're connected to the internet.

B. Dependencies (Python Libraries):

A Python library is a collection of pre-written code (functions, classes, and tools) that you can use in your own programs to save time and effort. These libraries must be installed for our application to work properly.

- *Stream Lit:*

A fast way to build interactive web apps using Python. It powers the entire front-end of your application. It creates dashboards, forms, charts, and real-time data displays without needing HTML/CSS/JS.

- *Pandas:*

A powerful data manipulation and analysis library. To load, filter, and manipulate vehicle maintenance data and sensor logs. It Helps manage structured data like service history, vehicle schedules, and prediction results.

- *Numpy:*

A numerical computing library for arrays and mathematical operations. It used for calculations related to mileage, temperature analysis, and other numeric operations. It works behind the scenes with pandas and machine learning algorithms.

- *Matplotlib:*

A plotting library for creating static graphs and charts. To visualize trends like engine temperature over time or mileage history. It is Useful for building simple, readable graphs in the Stream lit app.

- *Plotly:*

An advanced interactive charting library. For sleek, interactive graphs (like animated line charts or dials). It Enhances the UI by allowing users to zoom, hover, and explore vehicle performance data.

- *Pickle:*

A built-in Python module for saving and loading Python objects (like models). Used to load the pre-trained machine learning model used for predicting maintenance. It Helps store and retrieve data structures like dictionaries or arrays easily.

- *Requests*:

A library for sending HTTP requests. Used to fetch Lottie animations or data from external APIs. It also used if you integrate any cloud services, like a database or weather API.

- *Streamlit-Lottie:*

A Streamlit plugin that allows embedding Lottie animations. It adds animated visuals to make the app more engaging (e.g., alerts, loading animations) and supports animations hosted online via JSON URLs.

- *Streamlit-Option-Menu:*

A plugin to create stylish sidebar and horizontal navigation menus. It allows easy navigation between different sections of the app (like Dashboard, History, Eco Driving, etc.) and improves user experience and app layout.

- *Streamlit-Extras:*

A collection of community-contributed extras for Streamlit. It adds useful UI features like custom buttons, sticky headers, popups, and more and helps enhance the design without writing complex code.

C. Running the Application:

To run the application:

- ➢ Open a terminal/command prompt
- ➢ Navigate to the project directory
- ➢ Run the following command: streamlit run app.py
- ➢ The application will start and automatically open in your default web browser at 127.0.0.1:5000.

D. Experimental Procedure:

Using the Application:

➢ *Select Your SUV Brand:*

Choose your SUV brand from the available options:

- KIA
- TATA
- Mahindra
- Maruti
- Toyota

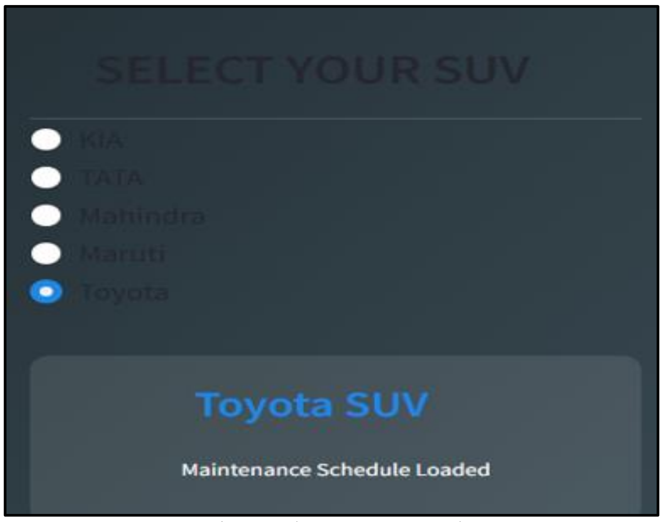


Fig 3 Select SUV Brand

➢ *Configure Driving Parameters:*

The application provides three tabs for configuring your driving parameters:

- *Speed & Time:*
- ✓ Adjust the average speed using the interactive speedometer (10-120 km/h)
- ✓ Set your daily driving duration (0.5-12 hours)
- ✓ Specify the number of driving days per month (1-31 days)

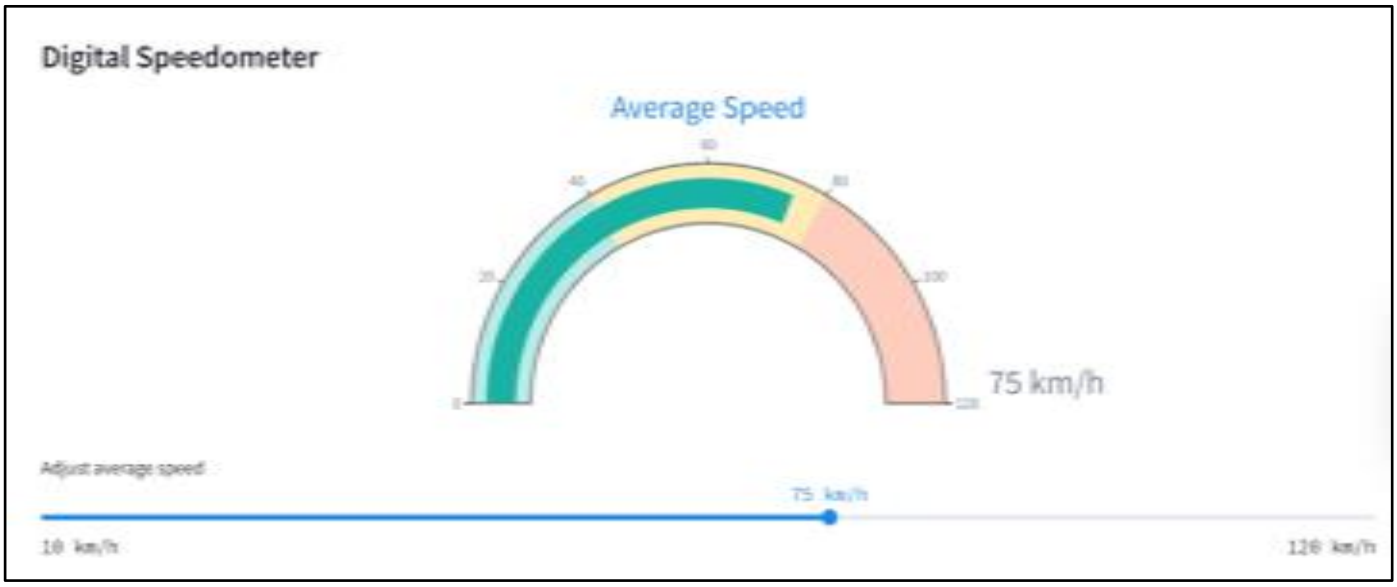


Fig 4 Digital Speedometer

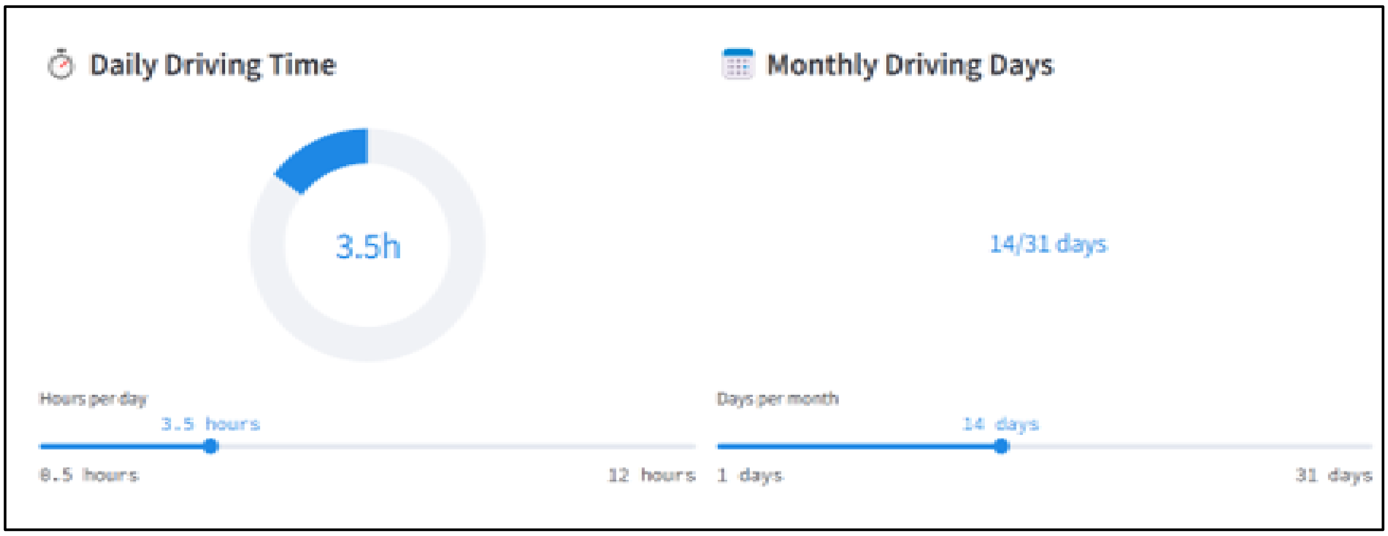


Fig 5 Daily driving time and Monthly Driving Days

- *Vehicle Operation:*

✓ Select your gear usage pattern (Low 1-2, Medium 3-4, High 5-6).
✓ Choose your vehicle load (Light, Medium, Heavy).

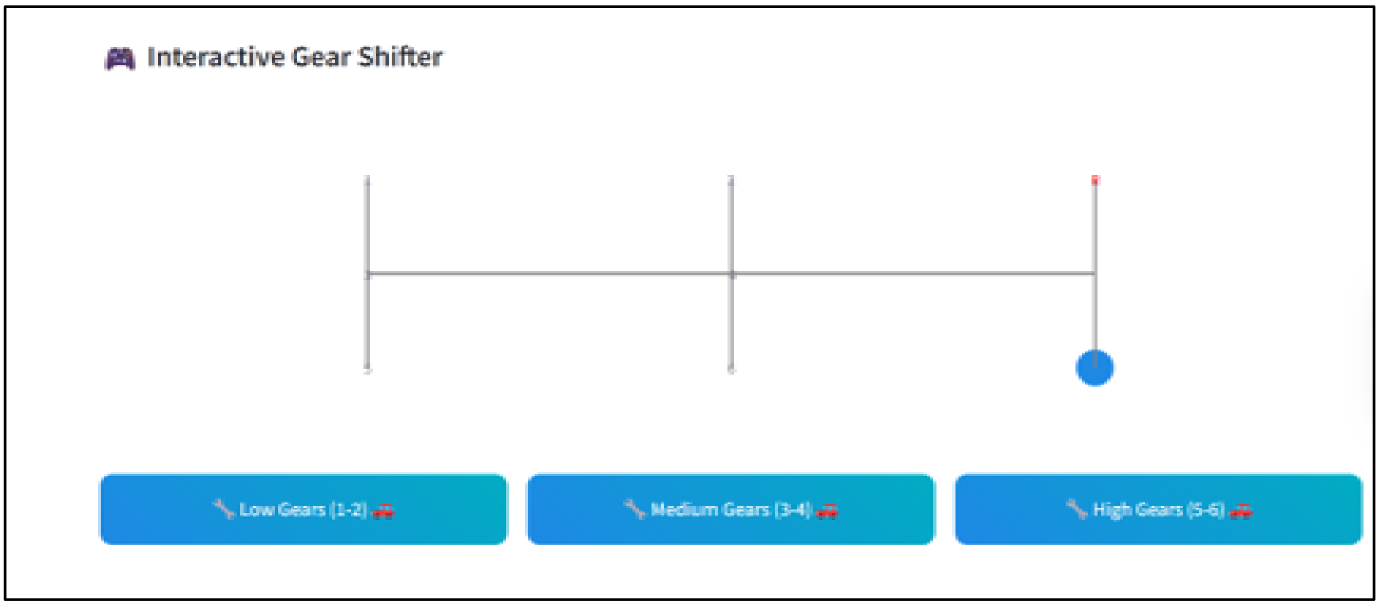


Fig 6 Interactive Gear Shifter

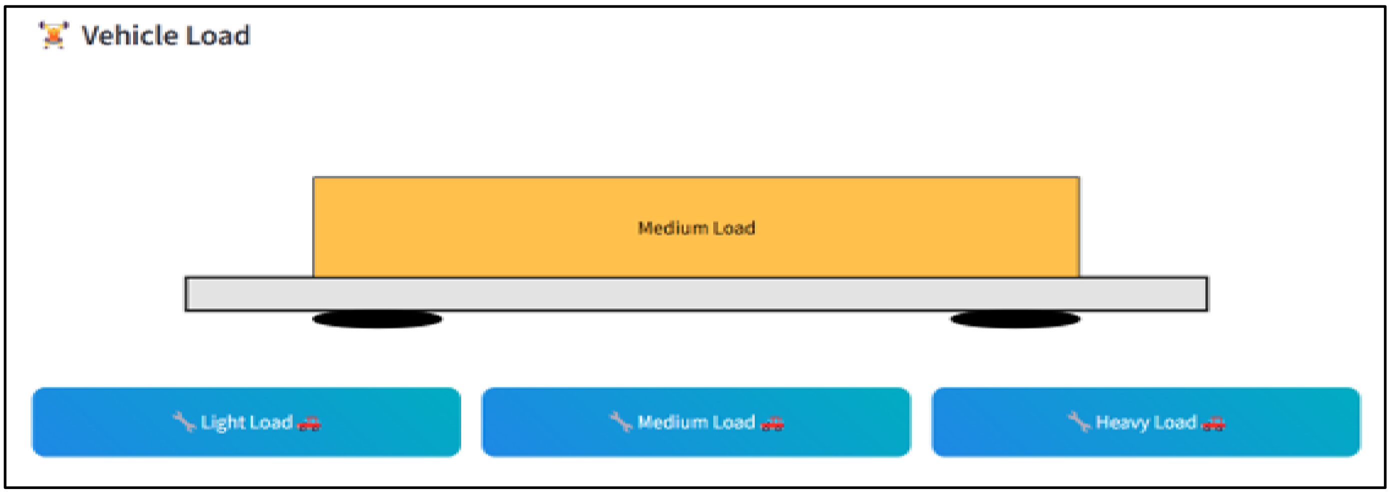


Fig 7 Vehicle Load

- *Environment:*

✓ Select the road type (City, Highway, Rural, Mixed, Off-road)
✓ Choose the weather conditions (Normal, Hot, Cold, Rainy, Snowy)
✓ Specify the traffic conditions (Light, Moderate, Heavy)

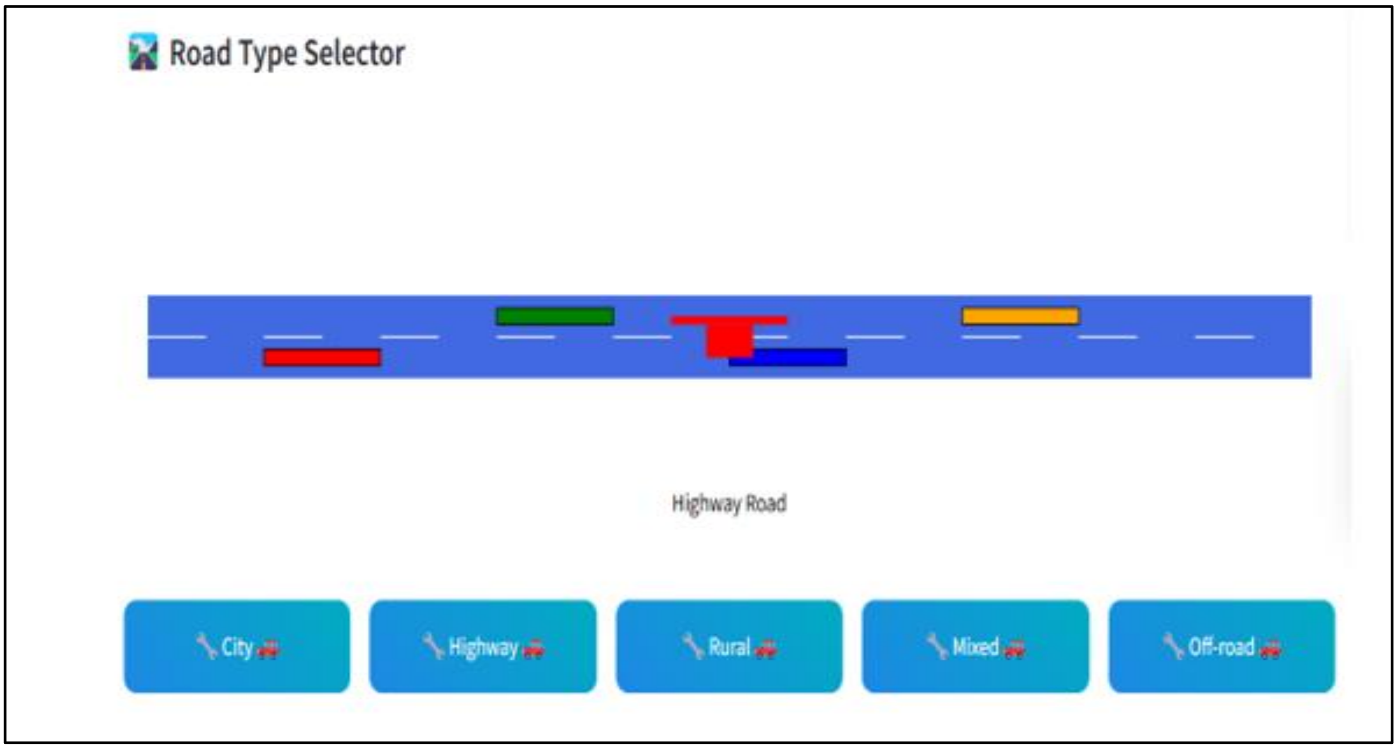


Fig 8 Road Type Selector

➢ *Generate Predictions:*
Click the "ANALYZE VEHICLE & PREDICT MAINTENANCE NEEDS" button to generate personalized maintenance predictions based on your inputs.

➢ *View Results*
The application will display:

- Estimated monthly distance
- Visual representation of your driving profile
- Maintenance schedule with gauges showing remaining kilometres until service
- Maintenance timeline showing when different components will need service
- Complete maintenance schedule for your selected SUV brand

IV. RESULTS

Once the application is launched, we are greeted with an interactive and data-driven dashboard that provides a holistic overview of their vehicle's current status, driving habits, and upcoming maintenance needs. Designed with clarity and user experience in mind, the dashboard consolidates real-time data, predictive insights, and manufacturer guidelines into a centralized view, enabling vehicle owners to make informed decisions and stay ahead of potential issues.

We can establish the various number of results by varying the following parameters:

- Average speed
- Daily driving duration
- Number of driving days per month
- Gear usage patterns
- Road type (City, Highway, Rural, etc.)
- Weather conditions
- Traffic conditions
- Vehicle load

Here, we are providing the maintenance prediction for different brands of Sports Utility Vehicle [SUV], like KIA, Toyota etc. The standard data is being developed from the different brand manuals and manufacturing.

Table 1 KIA SUV Maintenance Schedule with Dynamic Factors

Component	**Standard Interval**	**Dynamic Factors**	**Adjusted Interval**
Engine Oil & Filter	10,000 km / 12 mo	Traffic, short trips, towing, aggressive driving	5,000-7,500 km
Air Cleaner Filter	20,000 km / 2 yr	Dusty/rural roads, off-roading	10,000-15,000 km
Cabin (AC) Filter	20,000 km / 2 yr	Pollution, humidity, frequent AC use	10,000-15,000 km
Fuel Filter (Diesel)	40,000 km / 4 yr	Poor fuel quality, rural use	30,000-35,000 km

Brake Fluid	30,000 km / 3 yr	Hill driving, traffic	20,000-25,000 km
Coolant	1,00,000 km / 5 yr	Hot climate, towing	60,000-80,000 km
Transmission Fluid (AT)	90,000 km	Towing, urban driving	60,000-70,000 km
Transmission Fluid (MT)	1,20,000 km	Heavy traffic, aggressive shifting	90,000-100,000 km
Spark Plugs (Petrol)	70,000- 1,00,000 km	Short trips, poor fuel	60,000-70,000 km
Tire Rotation & Align.	10,000 km	Rough roads, fast driving	5,000-7,500 km

➢ *Case 1:*

Setup;

- *Select of SUV Brand: KIA*
- *Configuration of Driving Parameters*:

✓ Average Speed: 45km/h
✓ Daily Driving Time: 3.5 hours
✓ Monthly Driving Days: 20 days

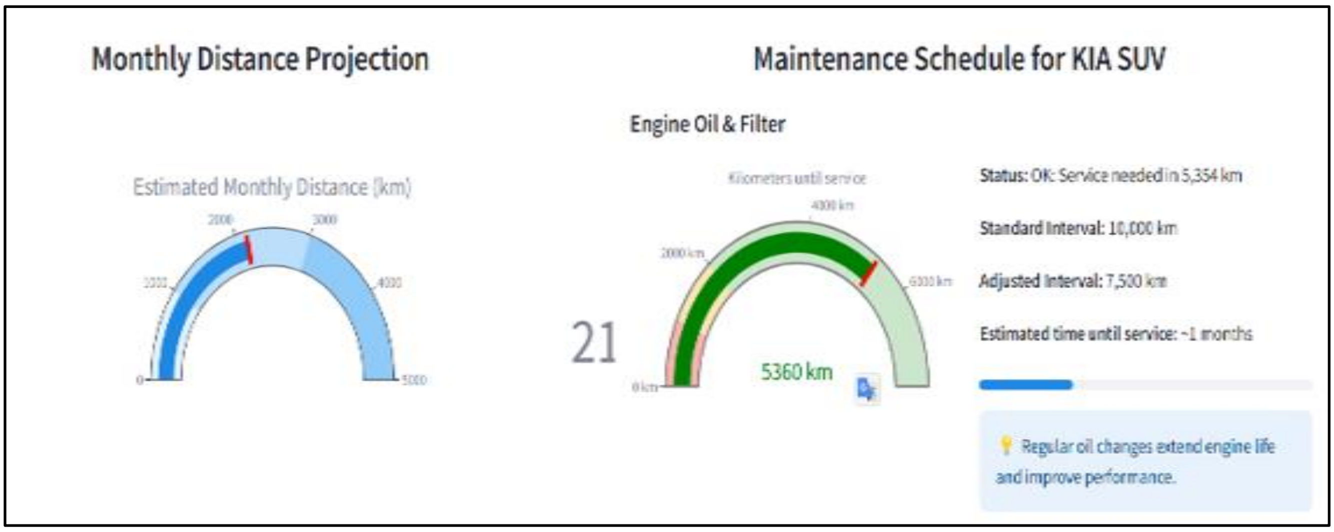
- *Vehicle Operations and Controls:*

✓ Interactive Gear Shifter: Medium Gears (3-4)
✓ Vehicle Load: Medium Load

- *Driving Environment:*

✓ Road Type: City
✓ Weather Condition: Hot
✓ Traffic Condition: Moderate Traffic

V. VEHICLE MAINTENANCE ANALYSIS RESULTS



Monthly Distance Projection

Maintenance Schedule for KIA SUV

Engine Oil & Filter

Status: OK: Service needed in 5,354 km

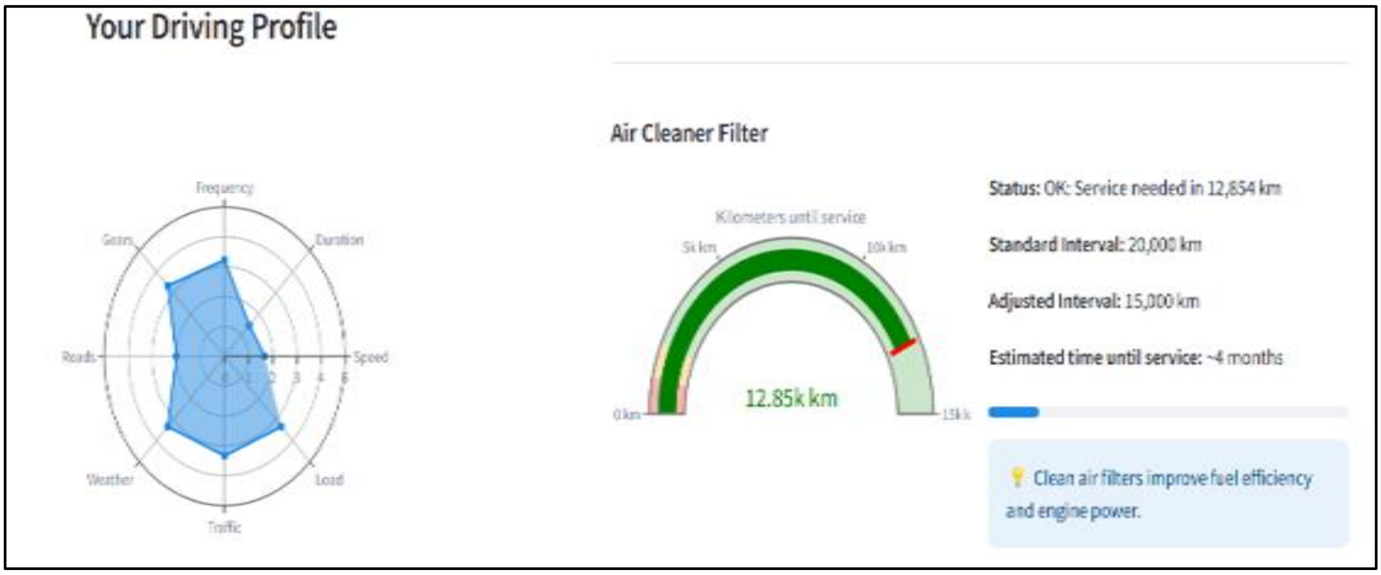
Standard Interval: 10,000 km

Adjusted Interval: 7,500 km

Estimated time until service: ~1 months

💡 Regular oil changes extend engine life and improve performance.

Fig 9 Monthly Distance and Engine Oil and Filter



Your Driving Profile

Air Cleaner Filter

Status: OK: Service needed in 12,854 km

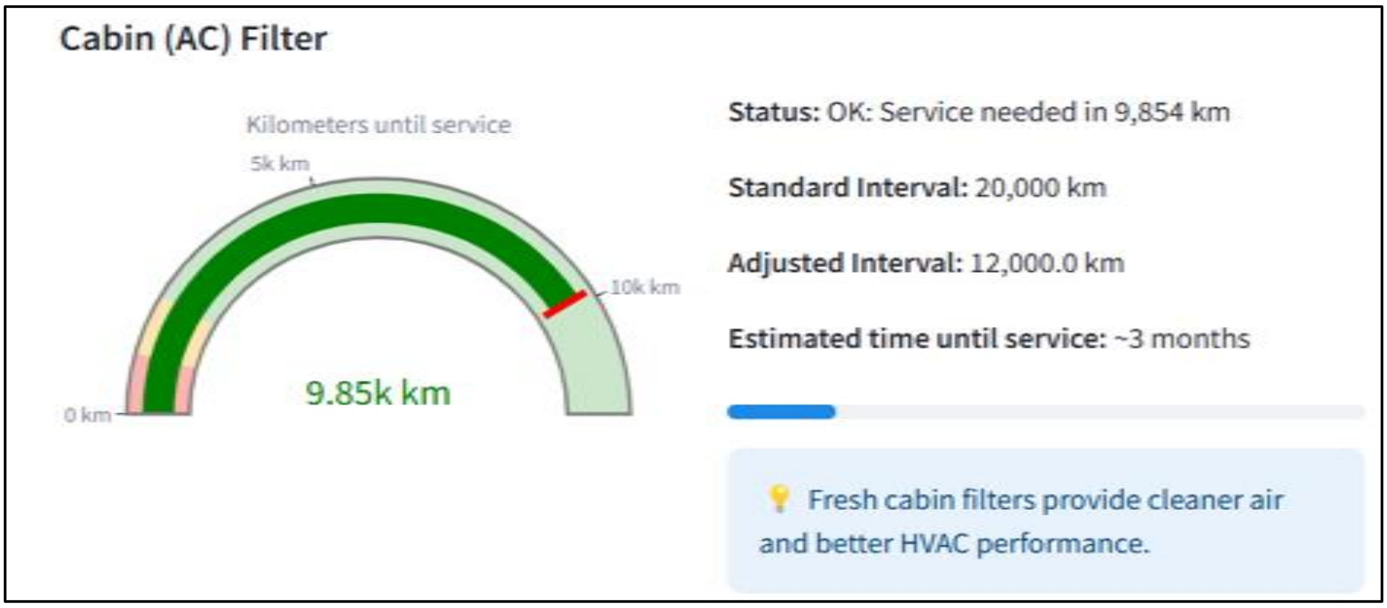
Standard Interval: 20,000 km

Adjusted Interval: 15,000 km

Estimated time until service: ~4 months

💡 Clean air filters improve fuel efficiency and engine power.

Fig 10 Driving Profile and Air Cleaner Filter



Status: OK: Service needed in 9,854 km

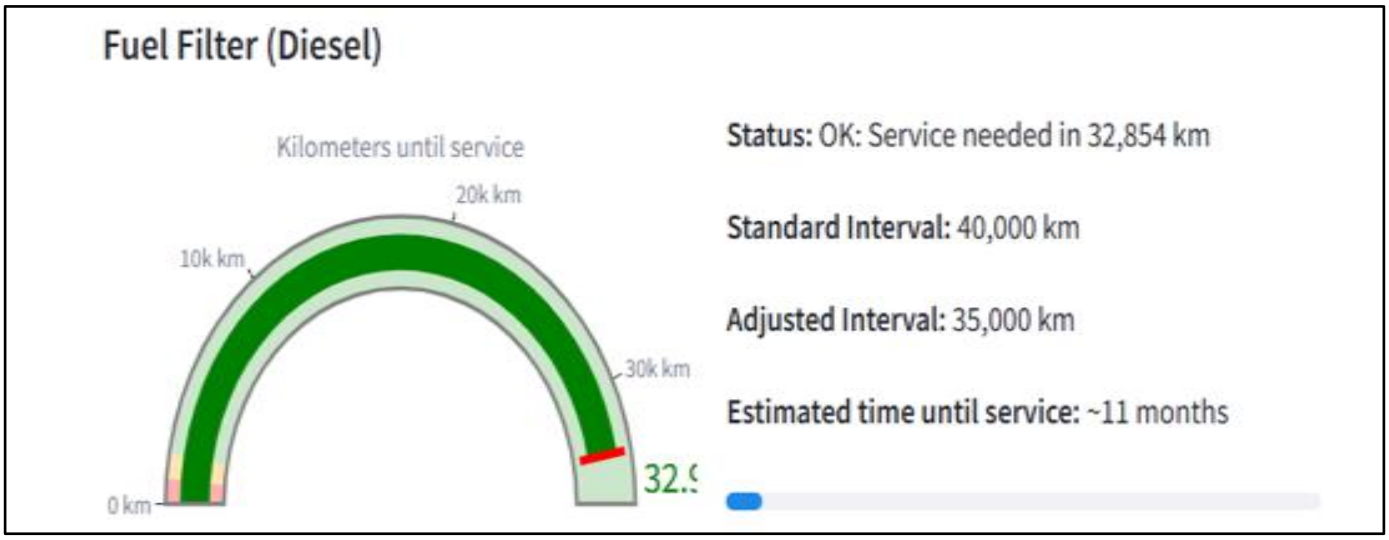
Standard Interval: 20,000 km

Adjusted Interval: 12,000.0 km

Estimated time until service: ~3 months

💡 Fresh cabin filters provide cleaner air and better HVAC performance.

Fig 11 Cabin (AC) Filter



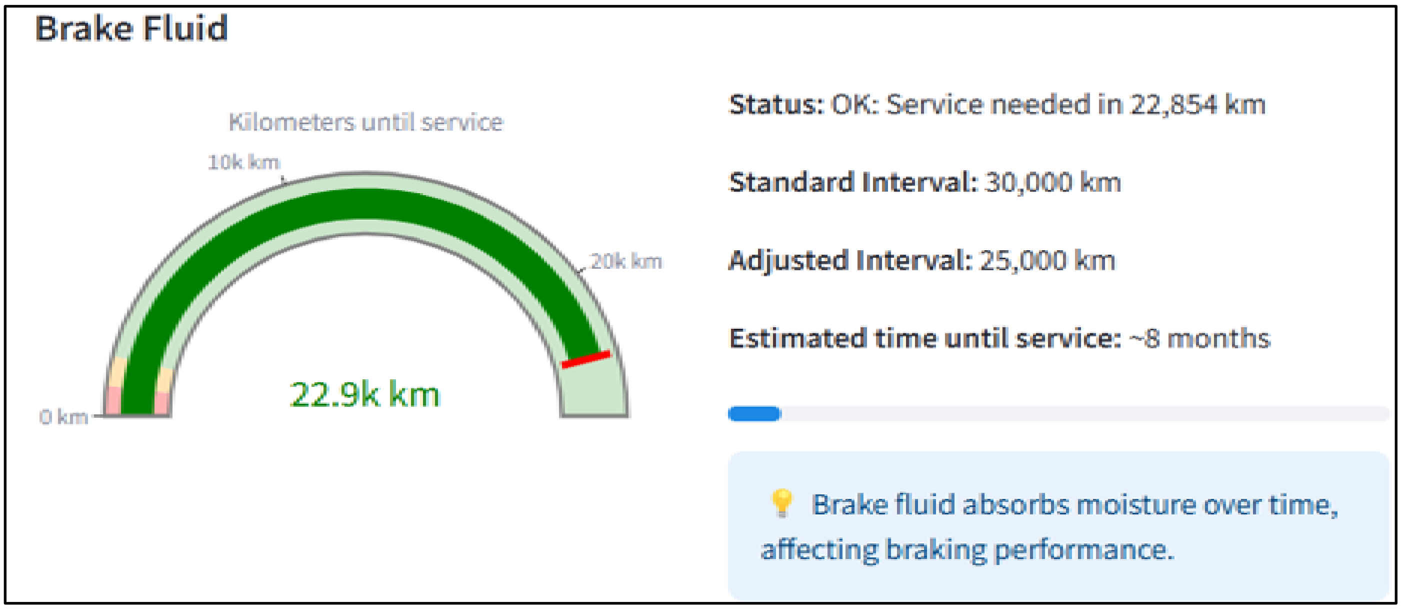
Status: OK: Service needed in 32,854 km

Standard Interval: 40,000 km

Adjusted Interval: 35,000 km

Estimated time until service: ~11 months

Fig 12: Fuel Filter (Diesel)



Status: OK: Service needed in 22,854 km

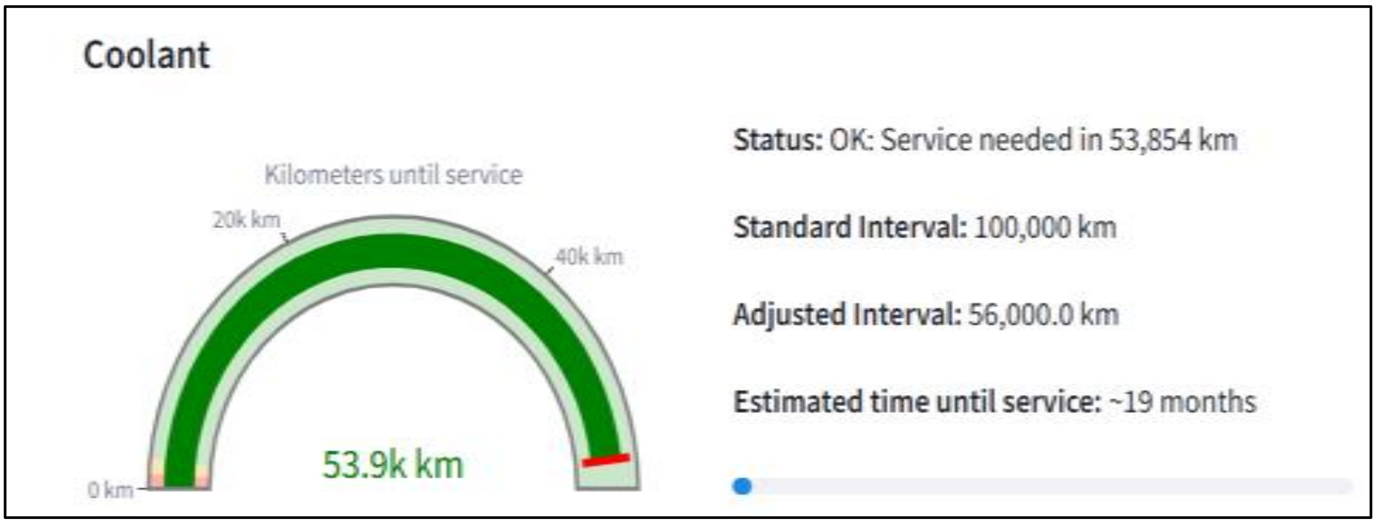
Standard Interval: 30,000 km

Adjusted Interval: 25,000 km

Estimated time until service: ~8 months

💡 Brake fluid absorbs moisture over time, affecting braking performance.

Fig 13 Brake Fluid



Status: OK: Service needed in 53,854 km

Standard Interval: 100,000 km

Adjusted Interval: 56,000.0 km

Estimated time until service: ~19 months

Fig 14 Coolant

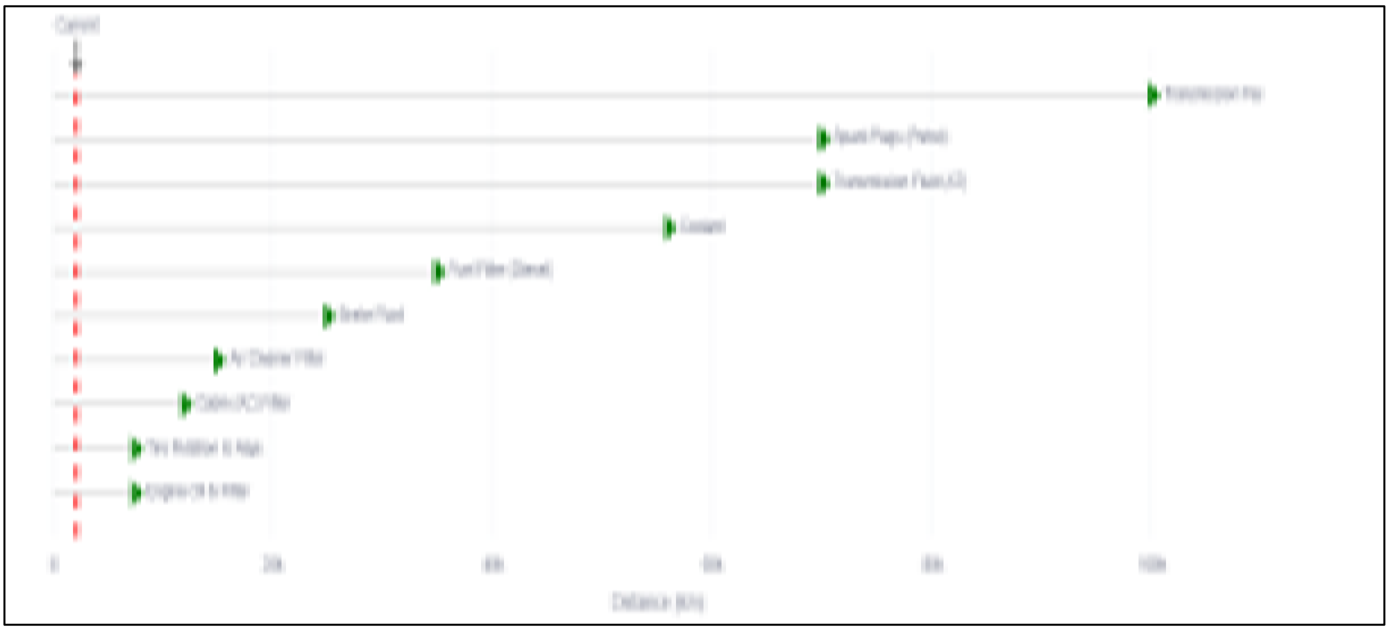


Fig 15 Maintenance Timeline for KIA SUV

➢ *Troubleshooting*
  Common Issues:

- *Missing Dependencies*

✓ *Error:* ModuleNotFoundError

✓ *Solution:* Install the missing package using pip install <package name>

- *Model Loading Error*

✓ *Error:* Failed to load the prediction model

✓ *Solution:* Ensure the model file exists in the correct location and has the expected format.

- *Visualization Errors*

✓ *Error:* Plot visualization not rendering correctly

✓ *Solution:* Ensure Plot is properly installed and browser supports JavaScript.

VI. CONCLUSION

The SUV Maintenance Predictor is an intelligent, web-based platform developed to support proactive maintenance planning for SUV owners. Built with Streamlit and powered by machine learning, the system personalizes maintenance recommendations by analyzing user-defined parameters such as average driving speed, daily usage duration, gear usage, road types, traffic and weather conditions, and vehicle load. These inputs are processed by a trained machine learning model that predicts the estimated monthly distance driven, which in turn drives the dynamic generation of personalized maintenance schedules.

Unlike traditional one-size-fits-all service reminders, this system creates adaptive maintenance plans tailored to individual driving patterns. It takes into account real-world environmental and operational factors, adjusting recommended service intervals accordingly. For example, SUVs driven under heavy loads or in harsh conditions (like off-road terrain or extreme weather) receive maintenance recommendations sooner than those driven in light, controlled environments.

The core of the system includes a modular architecture:

- A prediction engine that uses historical and synthetic driving data.
- Brand-specific service schedules stored in configurable datasets.
- A visual dashboard featuring gauges, timelines, and interactive widgets for intuitive understanding of upcoming service needs.
- By visualizing the driving profile and predicted wear on components, users are empowered to schedule services pre-emptively—avoiding costly repairs and reducing the likelihood of sudden breakdowns. The application supports brands like KIA, TATA, Mahindra, Maruti, and Toyota, and can be extended to more via simple configuration updates.

The integration of Lottie animations, Plotly visualizations, and session state management further enhances user experience, making the application not only informative but engaging and seamless across interactions. Additionally, the flexibility in customization (e.g., adding new brands or modifying service intervals) ensures long-term scalability of the platform. Ultimately, the SUV Maintenance Predictor achieves its mission: to deliver a smart, user-centric, and data-driven approach to vehicle maintenance. By combining predictive analytics with intuitive UX design, it improves vehicle reliability, extends component life, and provides SUV owners with peace of mind through actionable maintenance intelligence.

A. Scope for Future Work

To further improve the functionality, reach, and intelligence of the SUV Maintenance Predictor, the following enhancements are envisioned:

➢ *User Account Management:*
Enable users to create secure accounts to save their vehicle profiles, track historical maintenance data, and receive personalized reminders across sessions and devices.

➢ *Integration with OBD-II and Telematics Data:*
Incorporate direct data streaming from On-Board Diagnostic (OBD-II) devices and telematics systems to automate input collection and provide more accurate, real-time analytics.

➢ *Mobile Application Deployment:*
Develop a mobile-friendly version or standalone app for Android/iOS, allowing users to monitor and manage their maintenance needs on the go.

➢ *Expanded Brand and Model Support:*
Broaden the database to include a wider variety of SUV brands and specific models, enabling more granular maintenance scheduling and user adoption.

With these upgrades, the SUV Maintenance Predictor can transform into a comprehensive vehicle lifecycle management system, catering to the requirements of both individual owners and fleet operators within the expanding landscape of connected mobility.

REFERENCES

[1]. A. Lombard, T.S. Hattingh1 and E. Davies: Improving Vehicle Service Schedules at an Automobile Company, Transportation Technologies 2020.
[2]. Guixiong Liu, Yi Gao, and Jianlong Xu: Study and Simulation of Scheduling Strategies on Vehicle Operating Safety State Monitoring System, Automotive safety 2012.
[3]. Tan, M.H., Zheng, Y.B. and Li. W.H: A Set of Tracking Car Scheduling Management System, Transportation Technologies, 11, 660-668, 2021.
[4]. Kang Wang: Logistics Transportation Vehicle Monitoring and Scheduling Based on the Internet of Things and Cloud Computing, Advanced Computer Science and Applications,2024.
[5]. Ravi Aravind, Chirag Vinalbhai Shah Manogna Dolu Surabhi: Machine Learning Applications in Predictive Maintenance for Vehicles: Case Studies, Engineering and Computer Science, 2022.
[6]. Andreas Theissler, Judith Pérez-Velázquez, Gordon Elger: Predictive Maintenance Enabled by Machine Learning: Use Cases and Challenges in the Automotive Industry, Reliability Engineering,2021.
[7]. Raman Kumar, Anuj Jain: Driving Behavior Analysis and Classification by Vehicle OBD Data Using Machine Learning, Computer Science, Automotive Engineering, 2023.
[8]. Prajit Sengupta, Anant Mehta, Prashant Singh Rana: Predictive Maintenance of Armoured Vehicles using Machine Learning Approaches, Mechanical Engineering, Machine Learning, 2023.
[9]. Abenezer Girma, Xuyang Yan, Abdollah Homaifar: Driver Identification Based on Vehicle Telematics Data using LSTM-Recurrent Neural Network, Computer Science, Automotive Security, 2019.
[10]. Oscar Serradilla, Ekhi Zugasti, Urko Zurutuza: Deep Learning Models for Predictive Maintenance: A Survey, Comparison, Challenges and Prospect, Artificial Intelligence, Industrial Engineering, 2020.
[11]. M.A. Uddin, N. Hossain, A. Ahamed, et al.: Abnormal Driving Behavior Detection: A Machine and Deep Learning Based Hybrid Model, Transportation Systems, Machine Learning, 2025.
[12]. Dr. Opeoluwa Fawole: Machine Learning-based Predictive Maintenance for Autonomous Vehicle Components, Artificial Intelligence, Autonomous Vehicles, 2023.
[13]. Amir Hossein Baradaran: Predictive Maintenance of Electric Motors Using Supervised Learning Models: A Comparative Analysis, Electrical Engineering, Machine Learning, 2025.