

AI-POWERED SOLUTIONS FOR BREAKDOWN CHALLENGES WITH ELECTRIC VEHICLES

TMP-2023-24-117

Final Group Report

Supervisor - Ms. Shanika wijayasekara

CO - Supervisor - Ms. Sasini Hathurusinghe

B.Sc. (Hons) Degree in Information Technology Specialized in
Information Technology





Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

July 2023

Declaration page of the candidates & supervisor

We declare that this is our own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Group Member Name	Student ID	Signature
IT20096434	Perera B.N.H	
IT20038328	Sirimanne D.J.T.K.	
IT20038182	Madhubhashana A.G.K.	
IT20644826	Vijerathna A.G.V.K.M	

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

Signature of the Supervisor:

Date:

.....
(Ms.)

.....

Abstract

Certain regions, including Sri Lanka, have challenges in the rapidly developing field of electric vehicles (EVs), such as low levels of knowledge, a dearth of trained technicians, and a scarcity of replacement parts. The problems caused by electric car failures have never been addressed in such a groundbreaking way as this grant application proposes. Introducing our cutting-edge conversational AI chatbot, built specifically to assist with automobile breakdowns. Geolocation and machine learning to enhance emergency assistance for electric vehicle (EV) breakdowns. This is to create a reliable On-Road Breakdown Rescue System that links EV owners who are stranded with qualified professionals. This system is to lessen pollutants in the environment and encourage the use of electric vehicles. Mechanics are categorized depending on their expertise and location, giving customers the option to choose according to their own preferences. For a speedy response, users have the option to broadcast their service request to any technician that is available. Once accepted, the chosen expert can expedite the problem's resolution, cutting down on the EV owner's wait time and removing any aggravation. Furthermore, the program contains elements that let users evaluate and comment on the services they received. The findings of this study provide a comprehensive framework for performing proactive monitoring and intelligent data management-based optimization of vehicle maintenance. This will ultimately result in a decision-making process that is both efficient and well-informed. Through the incorporation of a number of different technological components, the primary purpose of the research is to offer a holistic solution. By bridging information gaps and giving drivers with up-to-date, accurate information on electric car difficulties, the chatbot improves their decision-making. Continuous updates ensure relevance for diverse electric vehicle types, while a powerful AI model uses carefully selected datasets to extensively investigate breakdown scenarios and propose unique solutions.

Keywords: *Electric Vehicle (EV), AI, Chatbot, AI Model, Proactive Monitoring*

Acknowledgment

We would like to start by expressing my sincere gratitude to our professors and other advisers, whose guidance and support have been invaluable to us during this project. Their help, patience, and constructive criticism were essential to the success of the whole investigation. In addition, we'd want to express our gratitude to the many friends, teachers, and mentors who have helped us along the way and contributed to the breadth and depth of our thesis. Your willingness to work together and your insightful comments have been motivational.

In addition, we'd like to thank the universities, museums, and libraries that made it possible to have access to the information we needed to investigate this intricate issue. Finally, we'd want to thank our loved ones for their constant cheerleading, patience, and understanding during my studies. Your confidence in us and constant support have been our inspiration. We wish it could be possible to express our deep appreciation to the many people and institutions whose efforts made it possible for us to complete our thesis.

Table Contents

Declaration page of the candidates & supervisor	01
Abstract	02
Acknowledgement	03
List of Abbreviations	05
1. Introduction.....	06
1.1 Background & Literature survey	06
1.2 Research Gap	14
1.3 Research Problem	20
2 . Objective	24
2.1 Main Objectives.....	24
2.2 Specific Objectives	25
3 . Methodology	29
3.1 Individual Component Overview.....	29
3.1.1 Conversional Chatbot for EV Assistance.....	29
3.1.2 Smart Digital Monitoring Calendar and Reminder System	67
3.1.3 Mechanic Finding Services for EV Breakdown Emergencies.....	108
3.1.4 EV Spare Parts Shop Finding Services	143
4 . Conclusion	180
References	181
Appendices	186

List of Abbreviations

Abbreviation	Description
AI	Artificial Intelligence
ML	Machine Learning
EV	Electric Vehicle
AML	Auto Machine Learning
CNN	Convolutional Neural Network
ICE	Internal Combustion Engines
SDLC	Software Development Life Cycle
WBS	Work Breakdown Structure

1 Introduction

1.1 Background & Literature survey

Due to its potential to significantly reduce pollution and dependence on fossil fuels, electric automobiles herald a new era of ecologically sustainable mobility. Unlike traditional internal combustion engine (ICE) vehicles, electric cars obtain their power from rechargeable batteries. Among the several advantages of this move are less pollutants, lower operating costs, quieter operation, and perhaps greater energy independence. The environmental impact of electric vehicles is so little that they are rapidly displacing ICEs (internal combustion engine vehicles) around the globe. The fact that electric vehicles do not contribute to air pollution makes them an essential tool in the fight against human interference with the natural world. Due to their improved energy efficiency, decreased fuel costs, and quieter, more comfortable journeys, they are becoming more appealing as an environmentally friendly mode of transportation.

The benefits of electric cars are compelling, but the adoption rate is still lower compared to cars powered by internal combustion engines. The International Energy Agency (IEA) reported that in 2020, ICE cars maintained an overwhelmingly huge market share of 95.4%. On the other hand, with the proliferation of charging stations and the constant development of better electric car technology, this trend is expected to reverse shortly.

The automobile sector is fundamental to modern civilization because it provides consumers and businesses with unmatched mobility and convenience. As more and more people have access to cars, the importance of regular maintenance becomes more and more apparent. The smooth operation of vehicles, the safety of their passengers, and the length of time they last on the road are all dependent on prompt maintenance, accurate paperwork, and appropriate repairs. Traditional approaches to car maintenance, however, may fall short on occasion due to inefficiencies and the likelihood of overlooking important details. Consequently, the likelihood of unforeseen breakdowns and the costs linked with fixing them are both raised.

Transportation stands out as a key source in the context of global efforts to address climate change. As of 2019, transportation was responsible for about 18% of the world's total

carbon dioxide emissions. Both consumers and businesses must make it a priority to implement environmentally friendly transportation solutions that are in accordance with the Sustainable Development Goals established by the United Nations in order to solve the urgent environmental threat that is now being faced. Prioritizing increased energy efficiency and decreased emissions of greenhouse gases is a necessary step in this direction. As a consequence of this, a new type of automobile known as intelligent electric cars has arisen. These automobiles have the potential to reduce carbon dioxide emissions by as much as 43 percent when compared to conventional diesel engine vehicles.

When it comes to electric cars, one major obstacle is handling breakdowns. There are several potential failure modes for electric automobiles since they use a different set of components and technology than internal combustion engine vehicles. Electric motor, power electronics, software, and battery system concerns (charging problems, capacity degradation, etc.) are common complaints from EV owners. It is becoming more and more important to handle breakdown management properly as electric vehicle technology advances, considering the specific components and intricacies of each electric vehicle.

As a consequence of the development of more advanced technologies, the industry of vehicle maintenance is currently going through a period of transition. Systems of proactive monitoring that make use of vehicle data have been shown to have the ability to minimize the number of oversights and enhance the efficiency of maintenance. This has been proved via a number of different studies. In a similar vein, advancements in optical character recognition (OCR) technology have made it feasible to simplify the collection of service records and bill details. As a result, the requirement for human data entry and the mistakes that are associated with input from humans has been eliminated.

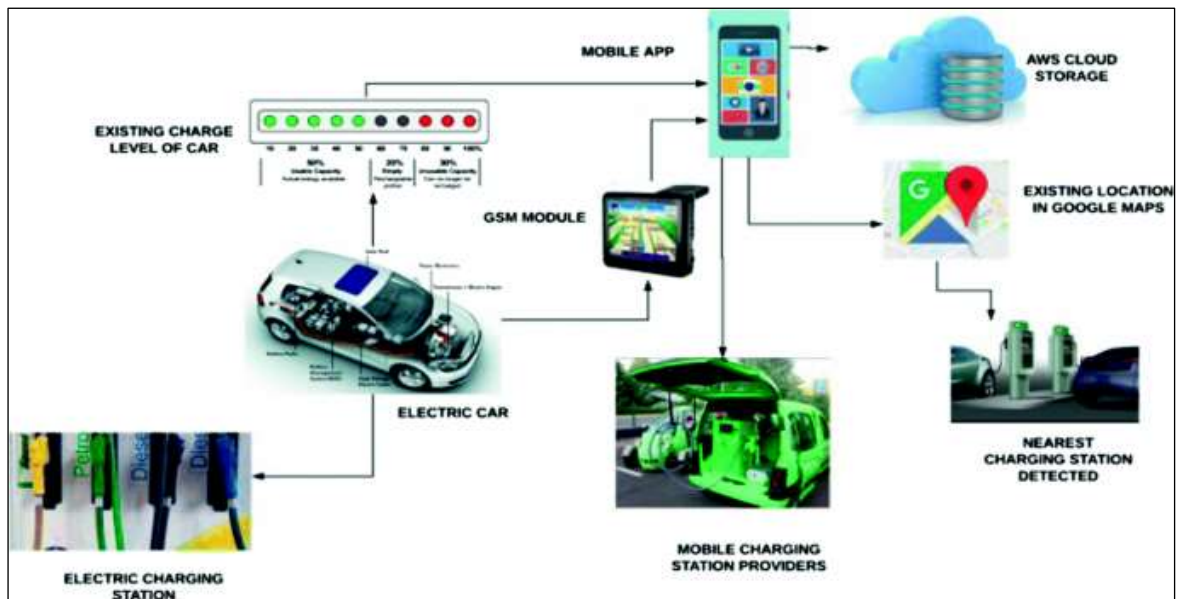


Figure 1 : Smart Electric Car Management System

The demand for sustainable transportation choices and worries about environmental pollution have led to a steady increase in the usage of electric cars, or EVs. As EVs proliferate on the road, it is imperative to guarantee their dependability and offer effective emergency support in the event of a breakdown. The main goal of this study is to improve emergency help for electric vehicle breakdowns by integrating machine learning and geolocation technology. This literature review explores the current research and advancements in this area.

When it comes to breakdowns, electric cars present different difficulties than vehicles with conventional internal combustion engines. Because EVs rely on extensive electrical and electronic systems, maintenance and diagnostics must be more involved. Previous research has indicated that in the event of an EV failure, specialist support services are required. A potential way to expedite the help process is through the combination of geolocation and machine learning technology.

Roadside assistance and ambulance dispatch are two emergency services that frequently employ geolocation technology. It makes it possible to pinpoint a user's position with accuracy, speeding up reaction times and enhancing the assistant's overall efficacy. The On-Road Breakdown Rescue System's integration of geolocation adheres to accepted best practices for emergency service systems.

Automating breakdown identification based on user descriptions has shown great potential thanks to machine learning methods, especially natural language processing (NLP). Studies in this field have shown that NLP models can correctly identify and categorize textual descriptions into breakdowns. Its ability to quickly match EV drivers with technicians with the necessary experience is essential.



Figure 2 : EV's Maintenance

A new paradigm for environmental consciousness and long-term planning has evolved in the last several years, altering the course of our history. The integration of AI with electric vehicles (EVs) lies at the heart of this revolution, which is causing huge shifts in our perspective on transportation and energy usage. Rising awareness of environmental issues and the critical need to cut emissions of greenhouse gases have propelled the electric car market to unprecedented heights. Because of this, the need for power plants and related infrastructure has increased. Generation, transmission, commercialization, and distribution are just a few areas of power management that are expected to be significantly impacted by the projected sixty percent growth in demand for electric power generation between 2019 and 2050. One concrete measure we can take to lessen our influence on the environment is the increasing popularity of electric automobiles. Our collective impact on the environment may be mitigated by switching from gas-powered cars to electric ones, which significantly cut down on pollution.

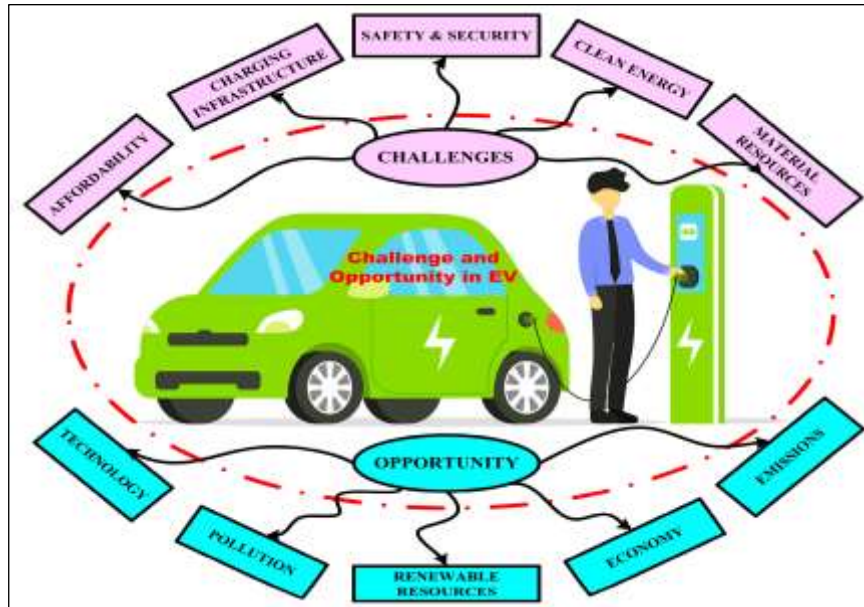


Figure 3 : EV Technology.

Through the process of comparing images of their broken components with those currently stored in the database, users are able to identify and order the right parts. The application of artificial intelligence and picture recognition has made this a feasible possibility. When it comes to the process of identifying and locating things within images, object identification and localization algorithms, which are a subfield of computer vision, play a significant role in the process. The utilization of these methodologies makes it possible to identify and localize the various components of a vehicle, which in turn assists in the accurate identification of the replacement parts that are required during the maintenance process. Consumers are not only able to identify the relevant component when location-based services are integrated with spare part identification, but they are also able to locate nearby stores that carry the component in question. This is a significant advantage. An investigation into the seamless integration of mapping and geolocation with online platforms is the focus of the research that is now being carried out in this department.

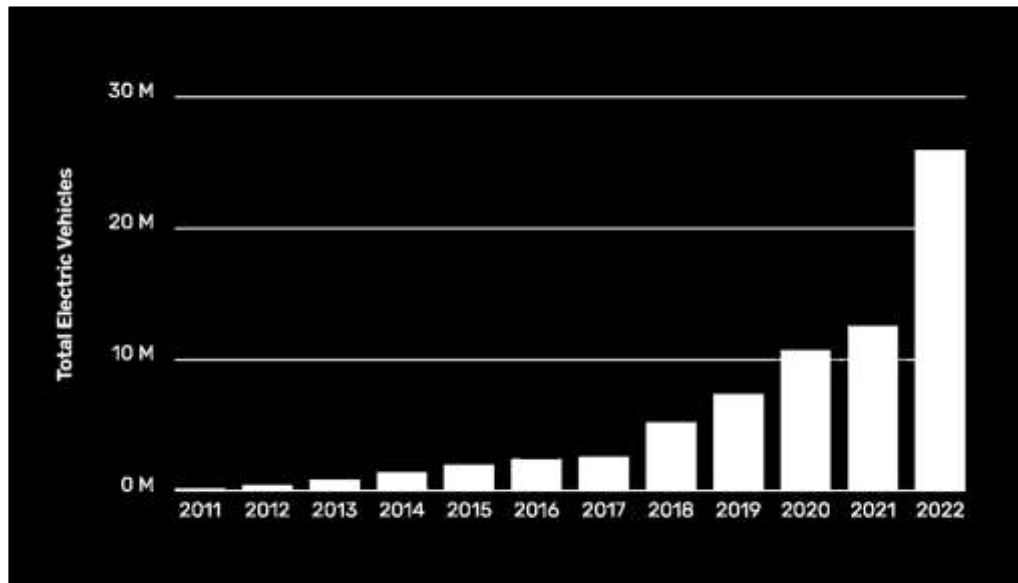


Figure 2 : Global electric vehicles are reaching 30 million units
by 2025 at the current growth rate.

Additionally, in addition to the straightforward work of obtaining replacement components, AI-powered systems make available a number of other advantages. Their ability to give real-time information on the availability of inventory, price, and promotions helps owners of electric cars to make informed decisions regarding the acquisition of replacement parts. This is made possible by the fact that they are able to deliver these updates. Additionally, these solutions have the capacity to provide a smooth communication channel between owners of electric cars and spare parts stores, which has the potential to facilitate the placing and fulfilling of orders in a more expedient manner.

While there are currently chatbots that can identify automotive defects and provide assistance in the event of a breakdown, the majority of these chatbots are designed to work with automobiles that are powered by internal combustion engines (ICE). When internal combustion engine (ICE) vehicles experience a breakdown, Smith et al. (2019) developed a system that is powered by artificial intelligence (AI) to assist them. On the other hand, these systems do not have the specific knowledge and individualized abilities that are required to effectively diagnose and give support for problems that are associated with electric vehicles.

The rising popularity of electric vehicles (EVs) has prompted a sea change in the car industry, one that prioritizes environmental responsibility and technological advancement. Numerous

environmental and economic benefits accrue to electric vehicles as contrasted to their internal combustion engine (ICE) counterparts. Some of these benefits include better energy efficiency, lower operating costs, and less emissions of greenhouse gases. However, problems associated with malfunctions and crises are becoming worse as the number of electric vehicles on the road increases. In comparison to owners of ICEVs, who have access to a larger network of mechanics and service centers, electric vehicle owners may find it more challenging to find qualified technicians who can handle EV-specific problems in the case of an emergency breakdown.

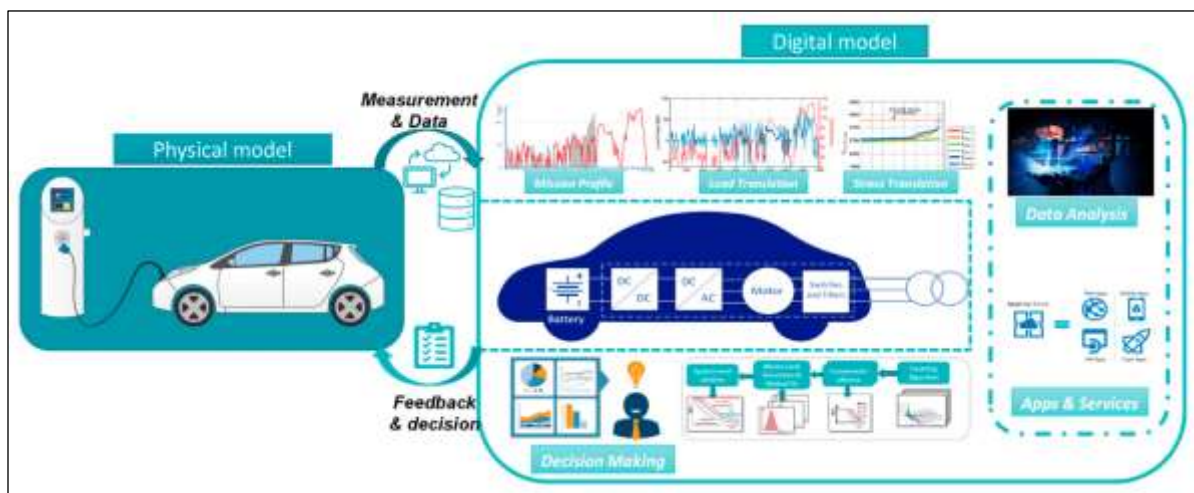


Figure 4 : Art of EV's

In the end, the introduction of electric vehicles represents a significant shift in the mobility industry. This shift is being driven by the good influence that electric cars have on the environment as well as the advancements that have been made in technology. Due to the fact that electric vehicles (EVs) have their own unique components and technology, they provide a unique set of challenges when it comes to handling breakdowns. By integrating artificial intelligence-powered predictive maintenance algorithms with digital monitoring schedules, a comprehensive solution is formed that optimizes the vehicle maintenance process for electric car owners. Enhancements to the user experience, heightened vehicle reliability, and the system's contribution to the overall sustainability of electric transportation are evident. As the number of electric cars that are being driven on our roads

continues to expand, there is an increasing need for effective solutions to tackle the challenges that arise from breakdowns. This is becoming an increasingly critical requirement. In the context of this attempt, the development of AI-powered systems for the purpose of locating stores that sell replacement parts for electric vehicles represents a significant step forward. In the event of an emergency, these methods provide owners of electric vehicles a chance to save their lives. These solutions not only simplify the process of procuring replacement parts, but they also empower owners of electric vehicles by giving them essential insights and assistance throughout the process of repairing their vehicles. Through the utilization of the powers of artificial intelligence, this is made feasible.

1.2 Research Gap

The current level of information about chatbots that assist with automotive breakdowns is severely lacking, particularly with regard to electric vehicles (EVs). Despite the fact that most existing solutions are aimed at conventional automobiles with internal combustion engines, there is a noticeable lack of investment in the development of chatbots specifically designed for electric vehicles. Because these vehicles have unique components, failure scenarios, and features, it is imperative to use a customized strategy. Determining the specific issues brought about by electric car malfunctions clearly calls for focused investigation. Consider the fact that very little is understood about the usage of real-time data by chatbots. There aren't many chatbots available right now that can accurately include the most recent data on the newest types of electric cars.

Furthermore, the sophisticated malfunctions seen in electric vehicles continue to surpass the capabilities of the fault analysis features now found in chatbots. To create cutting-edge AI models that can identify complex issues with electric car systems and provide customized fixes, quick action is needed. Electric vehicles (EVs) include complex electrical and electronic components, therefore developing AI systems that can comprehend these systems is essential to improving the chances of performing efficient breakdown analysis. We need to study NLP and user experience. Chatbots are becoming more popular, but it's hard for them to understand normal words and make the user experience better. It's hard to make robots that can understand questions, context, and answers like people do.

More study needs to be done on emergency robots. It is important to have chatbots that can connect people to emergency response systems when the power goes out. Emergency apps that are driven by AI need a lot of study and preparation to work quickly and correctly. A big study gap is how to deal with users' mistrust and chatbots' reliability. People may still not trust robots in important scenarios, even if AI has improved. Users' trust in AI-powered solutions relies on how reliable and accurate robot suggestions are. In the end, studying electric car problems helping chatbot is hard. EV-specific solutions, fault analysis, real-time data integration, user experience, emergency services integration, and reliability in research and innovation are all things that are important to this business.

Robots are great for having conversations. The program is made to work quickly and correctly. Expert systems help providers quickly come up with answers, which lowers the stress on the reaction system. This technology saves people time when they need to get skilled help for problems with their electric cars. To get themes from user searches, we used TF-IDF and N-gram. To answer the question correctly, each line is given less weight. Security and performance of applications are made better. More and more, public robots offer diagnosis, repair, and maintenance services. This piece looks at how current chatbots handle parts of the human-AI interface, the openness of AI automation, and making decisions in car repair services.

Chatbots are easy to use technologies that anyone can access by typing in their own language on mobile or PC devices. Based on reports, these robots figure out what's wrong with cars. Adding the position, length, intensity, and detailed descriptions of the problem may help find the fault in the future. Personal car helpers use AI algorithms and training data to make suggestions through gadgets that are linked to the internet. Voice and picture inputs should be combined.

Among the most significant deficiencies is the absence of a completely integrated system that is capable of digitally managing maintenance records and computing costs via the use of a monitoring schedule that is user-friendly. Nevertheless, this issue is addressed by my solution, which enables customers to keep records, compute expenditures, and obtain updates on upcoming repairs. When it comes to providing customers with a comprehensive picture of their vehicle's maintenance history and financial transactions, the methods that are now in use are insufficient. These methods usually rely on handwritten records and bills that are not well arranged. Furthermore, it is becoming increasingly clear that there is a severe shortage of tailored maintenance ideas. They are becoming increasingly visible. There are challenges that modern maintenance systems face when it comes to giving individualized guidance that is in line with certain driving patterns and the requirements of the vehicle. The present research on driving patterns is largely concerned with determining the emissions and energy consumption of conventional automobiles, controlling the energy that is consumed while driving, determining the viability of electric vehicles, and calculating the driving range of these vehicles. The existence of this hoover highlights the necessity of creating sophisticated algorithms that are capable to effectively analyzing historical service data in order to provide individualized repair recommendations.

Furthermore, there is a substantial amount of potential that has not yet been utilized by optical character recognition (OCR) technology in the market for car maintenance. In spite of the prevalence of technological advancements, the automation of data entry and administration through the utilization of optical character recognition (OCR)-based bill collecting is still an unexplored field. At the moment, there is no system in place that makes use of optical character recognition (OCR) technology for the purpose of recording bill data in electric vehicles. When it comes to precisely predicting the optimal times to replace various spare components, the predictive algorithms that are now available have the capacity to make use of data regarding mileage and service history. But this particular field of application has not yet been thoroughly investigated.

In addition, it is of the utmost importance to conduct an exhaustive investigation into the practical efficacy and convenience of use of the solution that has been recommended. The identification and resolution of potential issues, the acquisition of knowledge on user experiences, and the enhancement of the solution through iterative refinement are all critical goals that may be accomplished through comprehensive user testing. Last but not least, the merging of various maintenance data components, such as history records, bill information, and prediction insights, is still waiting to be investigated. There is still a lack of thorough awareness of the number of benefits and the real implementation of this entire integration.

It is possible that we may be able to generate chances for the exploration of new ideas if we acknowledge and embrace these areas of research that require more examination. This, in turn, has the potential to result in a substantial shift in the approach that is taken to the maintenance of vehicles. Automobile owners all over the world will be given more authority as a result of the convergence of informed decision-making, full integration, and technical innovation. While a growing body of research delves into certain facets of AI-powered predictive maintenance and digital monitoring systems for electric vehicles, very little research explores the holistic amalgamation of these technologies and their impact on user experience.

Furthermore, little study has been done to determine the exact needs and preferences of owners of electric vehicles with regard to maintenance reminders and scheduling. Gaining an understanding of the unique challenges and requirements associated with maintaining electric

vehicles (EVs), such as software updates and battery health monitoring, is essential to creating effective digital monitoring systems tailored to the EV ecosystem. Furthermore, not enough research has been done to determine if AI-driven maintenance solutions are suitable and scalable for different electric car models and manufacturers. Electric vehicle technology is developing quickly, necessitating the need for adaptable and interoperable maintenance systems that can manage a range of vehicle configurations and diagnostic procedures.

Moreover, there is a dearth of research examining potential cybersecurity risks associated with connected electric vehicle maintenance systems. There is growing concern about the vulnerability of maintenance platforms to cyber hazards, including remote hacking and unauthorized access to data, as cars—especially electric vehicles (EVs)—become increasingly interconnected. To ensure that AI-driven maintenance solutions are trustworthy and honest, it is critical to look into robust security protocols and processes to safeguard user data and car systems.

Combining Geolocation with Machine Learning for Electric Vehicle Failures: Despite the fact that geolocation technology has been utilized in emergency services such as ambulance dispatch, there is a dearth of research on how to combine geolocation technology with machine learning algorithms for the purpose of providing break-down assistance for electric vehicles. In order to bridge this gap and quickly connect stranded electric vehicle owners with competent technicians, the On-Road Breakdown Rescue System that has been developed attempts to properly identify breakdowns and intelligently couple service providers from different service providers.

A User-Focused Strategy for Electric Vehicle Breakdown Assistance: The literature review underlines the special challenges that are generated by electric vehicle breakdowns as a result of the complex electrical and electronic systems that are present in these vehicles. On the other hand, the applications that are now available do not always prioritize the user by combining proximity-based matching with tailored services. On the basis of their selections, the mobile application that was recommended would offer possibilities for providing prompt assistance and would enhance the user experience in general.

The use of novel techniques like data augmentation and dropout, together with Rectified Linear Units (ReLU) as activation functions, enhanced the efficiency and generalizability of the model. The paper's involvement in the ImageNet Large Scale Visual Recognition Challenge showcased its exceptional performance by achieving a significantly reduced error rate compared to earlier methods. This research will be renowned for its substantial contribution to the resurgence of neural networks and the foundational work it established for the profound transformation in deep learning, which has greatly impacted several domains of artificial intelligence. The fundamental topic of feature extraction in the field of image processing is investigated in the research paper named "B" Feature Extraction & Image Processing 2014, which was published in November of 2014. Through the process of extracting key qualities from raw picture data, the research highlights the significance of feature extraction as a means of enhancing various image analysis tasks.

The article presents a comprehensive study of a number of strategies for the extraction of features. These techniques range from the most fundamental approaches, which are based on intensity, to the more advanced techniques, which include edge detection, texture analysis, and form descriptors. The findings of this study demonstrate how important it is to select appropriate features that correctly reflect the distinctive characteristics of objects or patterns in images. In addition to this, it investigates the application of feature extraction in a variety of domains, including the identification of objects, the segmentation of pictures, and the retrieval of content-based images. In a nutshell, the findings of the study highlight the significance of feature extraction in terms of enhancing the quality and effectiveness of image analysis. Consequently, this makes it possible for practitioners and academics to comprehend image data in a more efficient manner and to use it in a variety of applications.

The research project named "An image-based auto parts catalogue" introduces an innovative approach to streamline the process of identifying and selecting vehicle components by using images. By primarily including photographs of vehicle components, the research effectively addresses the limitations of traditional text-based catalogues. Each picture is linked to essential data, including part numbers, measurements, and other compatibility information. The objective of this attempt is to enhance usability and accessibility, enabling anyone with less technical knowledge to recognize and choose the suitable automotive components easily and instinctively.

Users can expedite the identification process by comparing damaged or required components with catalogue photos, facilitated by the concept of visual matching. Furthermore, the essay includes an analysis of the technological aspects associated with the development of such a system. The elements included are database architecture, photo storage, and user interfaces. The newly created visual library has the capacity to completely transform the procedure of recognizing automobile parts. By offering a more visually appealing and user-friendly method for choosing components, it will offer advantages to technicians, car owners, and suppliers.

In accordance with the growing trend of proactive maintenance planning for electric vehicles (EVs), the inclusion of repair service appointment scheduling is fitting. This leads us to point number six, which is about electric vehicles. This feature not only makes electric vehicles easier to drive, but it also encourages electric vehicle batteries to run for longer periods of time and minimizes the possibility that they will have any issues.

1.3 Research Problem

The primary research topic at hand pertains to the overall goal of preventing automotive problems through the implementation of effective maintenance processes, facilitated by the integration of an intelligent digital monitoring calendar system. Automobile maintenance is fraught with the constant risk of unforeseen faults, which provide significant challenges for both car owners and enthusiasts. To properly tackle this pressing issue, it is crucial to build a systematic approach that prioritizes meticulous maintenance and harnesses the transformative capabilities of modern technology. This research project seeks to simplify the complex realm of vehicle maintenance by examining the collaborative advantages of proactive monitoring, streamlined record-keeping, predictive algorithms, and user-friendly interfaces, with the goal of preventing breakdowns.

There are no immediate problem detection tools for electric cars (EVs), making breakdowns while driving is a particularly challenging scenario. Due to the delay in receiving feedback, drivers often become confused about the nature of the issue and how to proceed appropriately. Using conversational artificial intelligence (AI) chatbots equipped with large knowledge bases is one innovative approach to effectively tackle this challenge.

An era of mobility has begun with the exponential growth of the electric vehicle (EV) sector, marked by reduced emissions and increased energy efficiency. However, as the number of electric vehicles (EVs) increases, so do the challenges presented by breakdowns and crises. Unlike traditional internal combustion engine vehicles (ICEVs), electric vehicles (EVs) provide unique maintenance and repair issues requiring specialized skills and expertise. When an electric vehicle (EV) breaks down in an emergency, especially in an unfamiliar location, EV owners sometimes find it difficult to locate local experts who are qualified to handle issues unique to EVs. The lack of an efficient and user-friendly system for timely and accurate assistance exacerbates the challenges and delays experienced by electric vehicle users, highlighting the need for a comprehensive solution.

The present area of study concern is the absence of a reliable and easily navigable system that can assist drivers of electric vehicles (EVs) in emergency breakdown situations, especially in

remote locations, with speed and accuracy. Conventional breakdown scenarios sometimes result in confusion and make it challenging to locate nearby mechanics, which irritates drivers and adds to delays. This difficulty is exacerbated by the fact that drivers may not know where to turn for skilled mechanics to address EV-related issues, and EV repairs require specific expertise and skills. Thus, there is an urgent need for a complete solution that makes use of machine learning, geolocation, and natural language processing to connect EV drivers in need of help with competent technicians, immediately determine the kind of breakdown, and account for proximity.

Electric vehicles (EVs) represent a significant advancement in the field of sustainable transportation, offering several benefits to both the economy and the environment. However, the increasing number of electric cars (EVs) on the road has brought attention to the inadequacies of the present systems in providing efficient and user-friendly assistance in emergency breakdown situations. Compared to conventional internal combustion engine vehicles (ICEVs), electric automobiles (EVs) present different maintenance and repair challenges due to their complex electric drivetrains and battery systems. Particularly in rural or unfamiliar areas with poor EV infrastructure, EV drivers regularly find themselves caught in emergency situations—such as battery failures, motor problems, or charging issues—without fast access to qualified assistance.

Additionally, electric vehicle drivers may use the most up-to-date data and insights provided by AI to make educated decisions rapidly. In addition to streamlining travel, this preventative measure promotes the increased usage of environmentally friendly modes of transportation. Chatbots powered by artificial intelligence (AI) can change customers' minds about sustainable mobility solutions and get them to trust electric vehicles for their reliability and convenience.

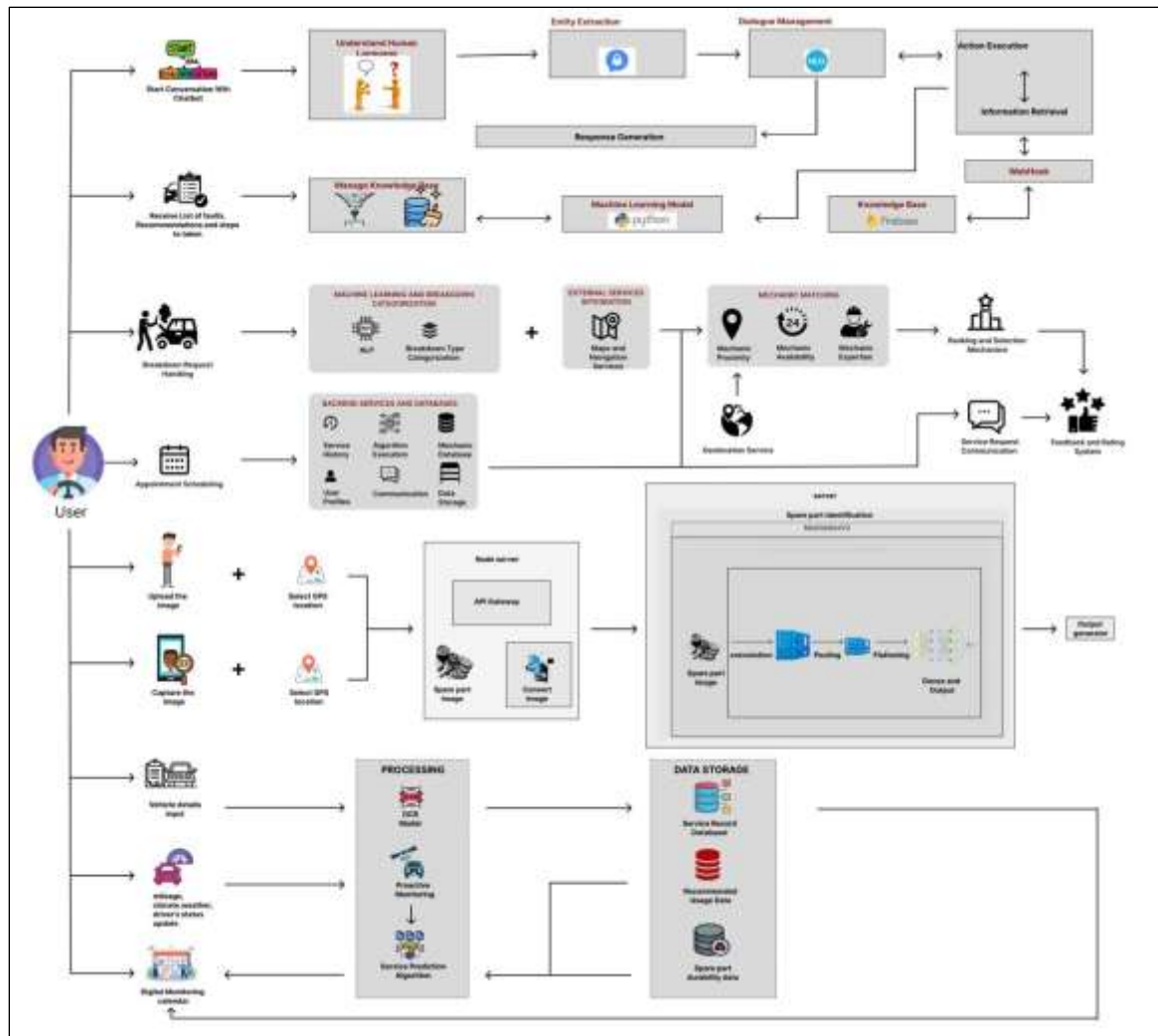


Figure 5 : Overall System diagram

Traditional troubleshooting services may be beneficial in the short term when it comes to electric cars (EVs), but they generally lack the in-depth experience and context that is required to identify and repair the underlying issues related to EVs. The fact that electric motors, battery management systems, and power electronics of varied degrees of complexity are all components of electric vehicle (EV) technology makes this component relatively difficult to implement. It is further exacerbated by the fact that electric vehicle failures can present a broad variety of symptoms, including mechanical and software faults.

This makes the creation of comprehensive and trustworthy chatbot solutions much more difficult. The first obstacle that must be overcome in this sector is the development of intelligent chatbot systems that are capable of precisely identifying problems with electric vehicles (EVs), providing individually tailored solutions, and offering quick customer care.

During the course of the conversation, the topic "AI-Powered Solutions for Breakdown Challenges with Electric Vehicles - Enhancing User Experience in Vehicle Maintenance with a Smart Digital Monitoring Calendar and Reminder System" was brought up. This topic highlights the importance of developing innovative strategies to address these problems. The use of artificial intelligence-driven technologies and intelligent digital monitoring systems presents a significant potential for the automotive industry to alter the way in which automobile assistance is provided. These technologies offer a number of advantages, including the ability to accurately identify parts, simplify the process of problem-solving, and improve comprehension of the requirements for vehicle maintenance. Additionally, they enhance the delivery of prompt and individualized service, which not only enhances the entire customer experience but also decreases the impact that automotive breakdowns have on the lives of consumers.

2 Objective

2.1 Main Objectives

The major objective of this project is to develop an advanced artificial intelligence chatbot that can assist with electric vehicle problems. Drivers will have greater control over their vehicles thanks to this chatbot, which rapidly analyses the information they provide on the sort of fault. With these details, the chatbot can correctly identify the issue and provide you with specific advice for fixing it.

This innovative solution not only fills the present demand in EV breakdown support, but it also equips drivers with the knowledge they need to handle EV breakdowns with ease. More individuals in developing nations like Sri Lanka will be able to afford to own and operate electric vehicles if this research succeeds in making them safer and more efficient to drive.

Improving the efficacy and efficiency of vehicle care is the goal of this comprehensive approach to maintenance, which combines proactive monitoring, OCR-based bill collecting, and seamless service record management. Our main goal is to provide consumers with all the information they need to make informed selections, so they may combine these state-of-the-art components to maximize their vehicle's performance and lifespan. A comprehensive strategy based on constant monitoring, advance warning of approaching repairs, and meticulous upkeep of service records is necessary to achieve this objective. The solution incorporates state-of-the-art optical character recognition technology to streamline the process of extracting bill data from user-uploaded photos. Important details regarding the vehicle, such as the user's mileage, weather, frequently used routes, and driving patterns, are regularly updated.

On-Road Breakdown Rescue System that will revolutionize the way electric vehicle (EV) breakdown assistance is provided. The architecture of this system will seamlessly include modern machine learning and geolocation technologies, satisfying the pressing need for quick and effective assistance for stranded drivers of electric vehicles. Using machine learning techniques, the system aims to accurately identify the kind of breakdown and connect users with trained experts or service providers who can tackle EV-specific issues. In addition, the system may employ geolocation technology to find stranded EV drivers with pinpoint accuracy, allowing for targeted assistance even in remote or unfamiliar places.

2.2 Specific Objectives

- A reliable artificial intelligence breakdown management system must analyze and collect data on typical EV issues. This method identifies typical EV issues using manufacturer information, service records, and consumer comments.
- A database with detailed issue information is needed to give electric vehicle (EV) owners with accurate and customized replies. First, organize problems by symptoms, causes, and solutions in a database. The defect report should include diagnostic criteria, troubleshooting techniques, and suggested fixes to help users fix the issue. By creating a robust knowledge base, engineers can ensure real-time AI chatbot help.
- • AI modeling requires data preprocessing and purification for multiple sources. This method must standardize data formats, replace missing values, and remove unnecessary information to ensure accuracy and uniformity. Using feature engineering to uncover key qualities improves AI model predicting. Researchers can improve AI model training accuracy by pre-processing and cleaning input data.
- For AI model training, machine learning algorithms detect patterns and relationships in pre-processed data. Before entering the AI model, data is separated into training and validation sets. Adjust model parameters for optimal performance. Training the AI model helps it identify common errors and predict human input. This allows sensible responses.
- The AI model's defect identification and rectification accuracy needs validation. Accuracy, recall, and F1-score assess model predictiveness. Model performance improves when expected results match ground truth labels.
- A few parameters and hyperparameter modifications increase model accuracy and generalizability. Modifying procedures, regularization settings, and feature representations may improve model performance. The model's ability to recognize and diagnose real-life issues would improve user satisfaction and confidence in the AI chatbot's guidance.
- A chatbot-friendly user interface is needed to integrate AI into a mobile app. Creating a smartphone app or introducing chatbots to messaging apps or vehicle service websites are options. By improving AI chatbot usability, developers may speed up EV issue resolution and increase customer experience.

- Verify chatbot ideas to give clients reliable advice. The chatbot's real-world performance needs extensive testing and validation. Developers can enhance the model by asking users and watching the chatbot for faults. The chatbot's effectiveness and dependability must be verified and improved.
- Developing Algorithms for Predictive Analytics Create algorithms that calculate when certain automotive parts should be changed based on user-supplied information such as mileage, weather, routes, driving behaviors, and past repair records. To generate accurate suggestions, these algorithms must consider a variety of criteria.
- The digital monitoring calendar system must be built and deployed within the mobile application. Create a user-friendly and visually appealing interface that allows consumers to see when their automobile needs service, enter and monitor maintenance activities, and set reminders.
- A proactive monitoring system that records the vehicle's mileage in real time and employs predictive algorithms to alert the driver to upcoming maintenance requirements should be implemented. These messages, which are timely and personalized to each user's individual car and driving patterns, are vital.
- By establishing streamlined record-keeping methods, users will be able to easily trace their vehicle's service history. Take all required care to keep this data safe in the database, ensuring that it is easily accessible and displayed in an ordered manner.
- Integrating User Feedback with Usability Testing: Conduct complete usability testing with individuals who would actually use the product to see where it falls short. Incorporating user feedback improves the mobile app's usability and usefulness.
- When developing the software application, create a mobile app that includes all of the aforementioned sub-goals. Make sure everything functions effectively together so that users may have a consistent and easy-to-navigate experience.
- • The app's clever matching algorithm connects breakdown descriptions with skilled mechanics to fix problems. To discover the best matches, the system will examine mechanical profiles and breakdown descriptions by availability, experience, and competence. This connects consumers with trained mechanics who can address automotive issues, boosting assistance.

- Mechanical Rating mechanics by competence and location lets users pick their favorites. To help consumers choose a mechanic, the app lists all area mechanics by geography and experience. This feature allows clients access to local, qualified mechanics who can address automotive problems, promoting autonomy and adaptability.
- Instant Help Broadcast Mechanism: Allows consumers to rapidly contact all available mechanics for emergency service. Users can broadcast aid from nearby mechanics in emergencies. Users can reduce downtime and increase road safety by seeking immediate expert assistance.
- The app may allow users to rate and review service quality. Repair service transparency and responsibility will improve. Customers can rate mechanics after using them. Other users can choose informed mechanics. This feedback method also motivates repair shops to improve and provide exceptional service.
- Apps that allow users to schedule service appointments improve convenience and encourage proactive car maintenance. Scheduling appointments for vehicle maintenance and repair is straightforward with the program. Schedule repairs at your convenience to reduce wait times. This function boosts service efficiency.
- The mechanic Acceptance and contact: The chosen mechanic can accept and handle the repair via the app. After selecting a mechanic, the user and mechanic can converse via app. They can verify the service request, provide more information, and schedule repairs. Instant communication improves user happiness and well-being by promoting openness and precision in aid.
- Take pictures of spare parts, starting with nuts and bolts and progressing to engine parts. Each image then has its spare component category or kind indicated.
- Clear labeling instructions for the network are needed for machine learning model training.
- Pre-processing follows image acquisition and labeling. Reducing photo size and normalizing pixel values improves training performance. Pre-processing simplifies computing and enhances model input pattern detection.
- Training, validation, and testing data sets are generated following pre-processing. Train the model to identify photo patterns, fine-tune its parameters, then test its performance

with the validation set. The training model assesses model accuracy and performance without seeing the testing set.

It is imperative to employ stringent security measures to protect user data, including car specifications, service records, and payment information. Instill a sense of data security in consumers and maintain strict compliance with all data privacy standards. The primary objective of this study is to develop a cutting-edge vehicle maintenance system that is thorough and original. This system aims to facilitate car owners in maintaining their vehicles in optimal condition through its user-friendly interface, efficient performance, and utilization of cutting-edge technology. In summary, there is a methodical process for creating an AI chatbot that can effectively address faults in electric vehicles. The process commences with conducting research, subsequently collecting data, training the model, assessing its performance, and eventually implementing it. To improve the effectiveness and efficiency of electric vehicle breakdown support, developers should follow these steps and continuously improve the artificial intelligence model using user input and performance indicators. The desired result should be a strong and dependable solution.

3. Methodology

AI-powered predictive maintenance, real-time diagnostics, and route planning tackle EV breakdown issues. Machine learning algorithms uncover vehicle sensor data trends before component failure in predictive maintenance. AI monitors battery condition, motor operation, and electrical systems to predict possible equipment breakdown. This proactive approach eliminates unexpected failures, extending electric vehicle component lifespan. The algorithms analyze past breakdowns, weather, and driving habits to produce more accurate vehicle-specific estimates. Monitoring vehicle performance and notifying drivers is essential for electric vehicle failure management. For timely system status updates, AI systems integrate with the vehicle's onboard diagnostics.

3.1 Individual Component Overview

3.1.1 Conversational Chatbot for EV Assistance.

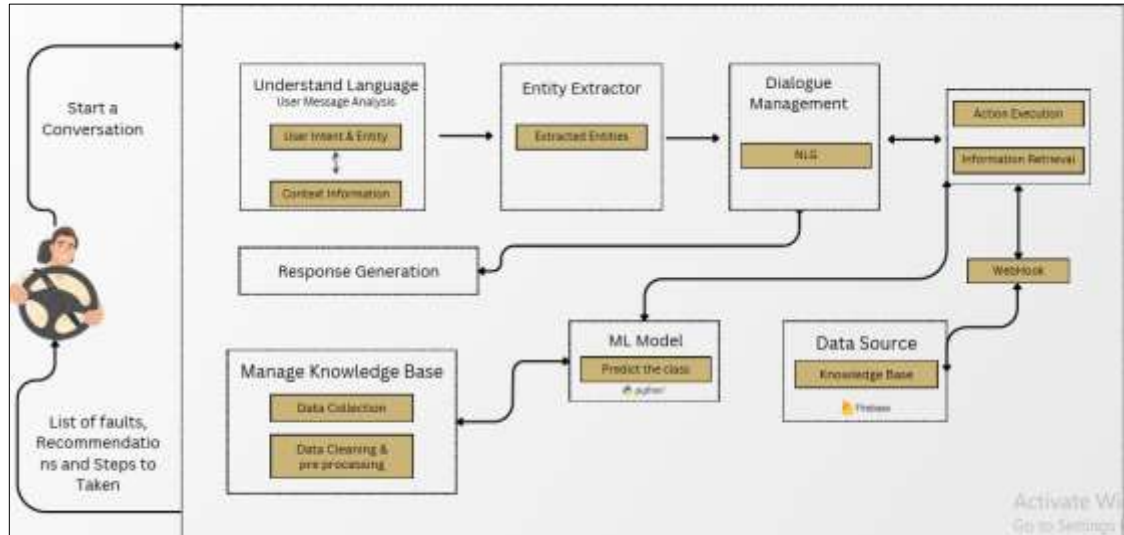


Figure 6 : How Conversational Chatbot works for EV.

A strategy that is offered is based on the utilization of conversational artificial intelligence chatbots to assist with the diagnosis of electric car malfunctions and the provision of individualized solutions by utilizing chat data given by users. In order to assist customers in resolving issues related to automobiles, we intend to make it simple

for them to obtain individualized guidance and assistance via text message. Because it enhances the chatbot's comprehension of user questions and its capacity to provide replies that are relevant, the use of machine learning techniques and natural language processing (NLP) technology is an essential component of this strategy.

In order to provide speedy responses to consumer requests and helpful advice on vehicle difficulties, the bot will explore a massive library of defective information using artificial intelligence. Because of this database, which will serve as the chatbot's foundation for knowledge, it will be able to give customers advice that is both accurate and delivered in a timely manner. However, in order to develop a chatbot that is capable of comprehending user questions and providing replies that are both natural and intuitive, it is important to make use of advanced natural language processing techniques.

With the help of recent developments in natural language processing, the chatbot is able to comprehend the regular speech patterns of drivers and answer in a manner that is both natural and conversational. Through the process of parsing the user's inquiries, comments, and descriptions of automobile problems, the chatbot is able to comprehend the user's intentions and deliver helpful responses. Natural language processing techniques are used to facilitate the study of text data. This enables the chatbot to identify significant phrases, comprehend the context, and generate smart replies to queries posed by users.

The use of natural language processing (NLP) and machine learning techniques is required if you want your chatbot to improve over the course of time. These algorithms, which learn from the data that it receives, allow the chatbot to improve its ideas and responses over time. This allows the chatbot to become more useful. By adjusting and refining its knowledge of user questions and preferences, the chatbot may be able to increase its ability to provide accurate and helpful advice through the process of iterative learning.

Through the analysis of trends and patterns in user interactions, the identification of common challenges, and the application of machine learning techniques, the chatbot is able to tailor its responses while speaking with users. Through the process of mining chat logs, the chatbot is able to gain intelligence on the behavior and preferences of users based on their prior interactions. By gaining knowledge from its errors and adapting to the ever-evolving requirements of its users, the chatbot has the potential to enhance the quality and relevancy of its responses over the course of time.

Through the use of machine learning algorithms, the chatbot is also able to make use of feedback mechanisms, which allows it to improve even more. It is possible for the chatbot to learn where it can improve by soliciting feedback from users and modifying its algorithms based on how effectively its response's function once it has received this feedback. The continuous operation of this feedback loop ensures that the chatbot will always be able to respond to user requests and easily adapt to new circumstances.

In conclusion, the development of a chatbot that is capable of providing individualized support and direction for electric vehicle (EV) issues requires the use of cutting-edge natural language processing (NLP) techniques and machine learning algorithms. Through the use of these technologies, the chatbot is able to interpret user questions that are stated in natural language, obtain relevant information from text data, and deliver appropriate responses. In addition, the chatbot may be able to learn and improve over time as a result of machine learning, which will allow it to continue to comply with customer requests and aid with problems related to vehicle maintenance.

3.1.2 Model Trained

The process of training a conversational chatbot for electric vehicle (EV) support utilizing the BERT (Bidirectional Encoder Representations from Transformers) base model encompasses many crucial stages. First and foremost, the most important thing is to collect data. This involves gathering a large dataset of conversations that are relevant to supporting electric vehicles. This dataset includes transcripts of conversations between electric vehicle (EV) owners and customer support agents, frequently asked questions (FAQs) about EV maintenance and troubleshooting, and other relevant textual information.

After collecting the data, the dataset is subjected to preprocessing in order to prepare it for training using the BERT model. The preparation portion of the chatbot application usually involves operations like tokenization, converting to lowercase, removing punctuation, and maybe using stemming or lemmatization to meet specific needs. After preprocessing, the data must be structured in a suitable manner for training the BERT model. This involves transforming the text input into numerical representations, such as token IDs, which may be consumed by the model during the training process.

Afterwards, the BERT basic model undergoes training utilizing the structured data. During the training process, the model acquires the ability to understand the meaning and context of the input text by making predictions about the missing words in the input sequence and using the surrounding context to its advantage. Once the BERT basic model has been trained on a large amount of text data, it may be fine-tuned to improve its effectiveness in assisting with conversational EV tasks. Fine-tuning refers to the process of modifying the parameters of a pre-trained model in order to better match the specific requirements of the target task and dataset.

After a successful evaluation, the trained model is deployed as a conversational chatbot to provide support for electric vehicle (EV) related queries. The chatbot may be seamlessly included into diverse platforms such as websites, mobile applications, or messaging platforms, in order to provide immediate help and guidance to electric vehicle (EV) owners on maintenance, problem-solving, and other pertinent inquiries. This method utilizes the

capabilities of pre-trained language models to provide precise and contextually appropriate replies, therefore improving the support experience for electric vehicle customers.

The LSTM (Long Short-Term Memory) machine learning algorithm is the fundamental component of a conversational chatbot specifically developed to aid customers in addressing inquiries pertaining to electric vehicles (EVs). This advanced technique, based on the architecture of recurrent neural networks, allows the chatbot to understand user inputs, provide suitable responses, and keep track of the discussion. The chatbot is designed exclusively to provide support for electric vehicles (EVs). It uses LSTM-powered natural language understanding skills to process user queries related to EV models, charging stations, range estimation, and maintenance suggestions. The chatbot undergoes a complex series of steps to prepare user inputs, which are then inputted into the LSTM model.

This model has been trained on a dataset of talks relevant to electric vehicles (EVs). The chatbot generates responses by applying patterns it has learnt from the training data, and then presents these responses in a natural language fashion. Prior to deployment, the LSTM model undertakes intensive training on a dataset consisting of user inputs and their associated replies. During this training, the model optimizes its parameters to minimize the difference between the responses it generates and the actual responses. After being implemented, the chatbot undergoes continual learning and improvement based on user interactions, ensuring that its responses stay accurate, useful, and conversational. In summary, the chatbot that uses LSTM technology simplifies the process of aiding electric vehicles, improving the user's experience and encouraging the usage of electric vehicles by offering easily available and intelligent support.

3.1.3 Technologies used.

A sophisticated chatbot for electric car support may be created by combining cutting-edge technologies such as RASA AI-powered solutions, Python, React Native, Expo, TensorFlow, Node Server, machine learning models, and RNNs. Developers may improve the user experience and resolve the problems encountered by electric vehicle owners during breakdowns by employing these technologies to develop a user-friendly and tailored intelligent chatbot.

React Native : To ensure a smooth and successful user experience, it is essential to incorporate advanced technologies into the construction of an AI-powered chatbot for electric vehicle (EV) support. By employing frameworks like React Native and Expo, it is possible to develop an application using a unified codebase that can run on both iOS and Android devices. Developers may utilize React Native and Expo to provide broad compatibility and accessibility for electric vehicle (EV) consumers seeking assistance with breakdown difficulties, independent of the specific device they are using.

Python : Python, a widely acclaimed and adaptable programming language, plays a pivotal part in building the many components of the chatbot solution. Python offers the capability and flexibility necessary for building intricate jobs, whether it entails backend server logic or the creation and incorporation of machine learning models. Python provides developers with a diverse set of tools for building chatbots capable of managing extensive data, smoothly incorporating APIs, and effectively implementing business logic.

TensorFlow framework : Google developed the TensorFlow framework, which is open-source and often employed for building and training machine learning models. TensorFlow enables the development and implementation of powerful machine learning models for many tasks in an EV help chatbot, including sentiment analysis, fault identification, and user intent recognition. By harnessing the capabilities of TensorFlow, it is possible to create models that can reliably assess user queries, understand sentiment, and offer customized suggestions that match the specific requirements and tastes of individual users.

Node Server : The chatbot solution's backend design employs a Node Server to provide a link between the mobile app and many third-party apps and APIs. The event-driven design and asynchronous programming style of the Node Server provide seamless integration with third-party services, including sentiment analysis APIs, EV diagnostic tools, and database systems. These attributes offer exceptional performance and scalability. The chatbot's capability and immediate support are made possible by a robust and flexible backend architecture that developers may construct using Node Server.

Recurrent Neural Networks (RNNs) : The chatbot's capacity to comprehend and proficiently address human inquiries is greatly improved by machine learning models, such as recurrent neural networks (RNNs). Recurrent Neural Networks (RNNs), a type of artificial neural network, demonstrate exceptional performance in several applications including natural language processing, sentiment analysis, and context-aware response creation. One effective method for teaching chatbots to understand customer inquiries, recognize pertinent details, and respond intelligently in basic English is to utilize RNN-based models on extensive collections of user interactions and EV-related data.

RASA : RASA is a free and open-source conversational AI platform that provides a wide range of tools for building high-quality chatbots capable of handling complex natural language tasks. RASA allows developers to incorporate functionalities including intent recognition, entity extraction, conversation control, and sentiment analysis. This allows the chatbot to comprehend user inquiries inside their specific framework and deliver tailored replies. RASA's modular architecture and extensive documentation facilitate the fast development and deployment of AI-driven chatbots for many purposes, including electric vehicle (EV) assistance.

3.1.4 Commercialization aspects of the product

Commercializing an AI-powered conversational chatbot for EV support is one of many stages needed for its success and market uptake. It covers pricing, marketing, business models, strategies, regulatory compliance, and more. By managing these aspects, businesses may develop an AI-powered chatbot solution and capitalize on the growing need for EV breakdown help.

1. **Market Analysis and Target Audience Identification:** Before selling the chatbot, target customers and competitors must be determined. The chatbot can be tailored to electric vehicle owners, fleet administrators, and automotive service providers to meet market expectations.
2. **Strategic Alliances and Joint Ventures:** Work with charging infrastructure suppliers, manufacturers, and roadside support providers to boost the chatbot's value and market share. We get distribution networks, resources, and expertise from industry heavyweights, improving our market entry chances.
3. **Business strategy:** To make money, the chatbot needs a strategy. Electric car virtual assistants might be subscription-based, pay-as-you-go, freemium with premium features, or OEM- or service provider-bundled. Long-term sustainability and profitability necessitate balancing each model's pros and downsides and aligning it to consumer preferences and market changes.
4. **Chatbot setup Pricing:** Consider value proposition, competitiveness, perceived value, and willingness to pay. Pricing may depend on customization, use, and features. Tiered and dynamic pricing increase profitability while serving different clients.
5. **Product Differentiation and Value Proposition:** A chatbot must offer unique features, benefits, and other methods to distinguish out in a competitive market. Highlighting the chatbot's ability to provide targeted, on-the-spot electric car help, apply AI-driven diagnostics, and interact with current systems may improve its value and attract clients.

6. **Marketing and promotion:** Effective marketing and promotion techniques increase chatbot awareness, interest, and use. Social media, SEO, and content marketing reach audiences. Attending electric vehicle (EV) events, networking with industry leaders, and marketing may increase exposure and credibility.
7. **Data Privacy and Regulatory Compliance:** Follow data protection rules to build client trust. Strong security, data encryption, and user approval mechanisms protect sensitive data and prevent privacy intrusions. Quality and compliance are shown by industry standards and certificates.
8. **Quality of client Service:** Live chat, email, and phone support resolve client difficulties rapidly. Using performance indicators and client feedback to improve service quality increases satisfaction and retention.
9. **Continuous innovation and product improvement:** These are necessary to beat the competition and satisfy customers. Chatbot research and development to improve functionality, accuracy, and compatibility with new EV technologies assures its value and relevance in a fast-changing business.

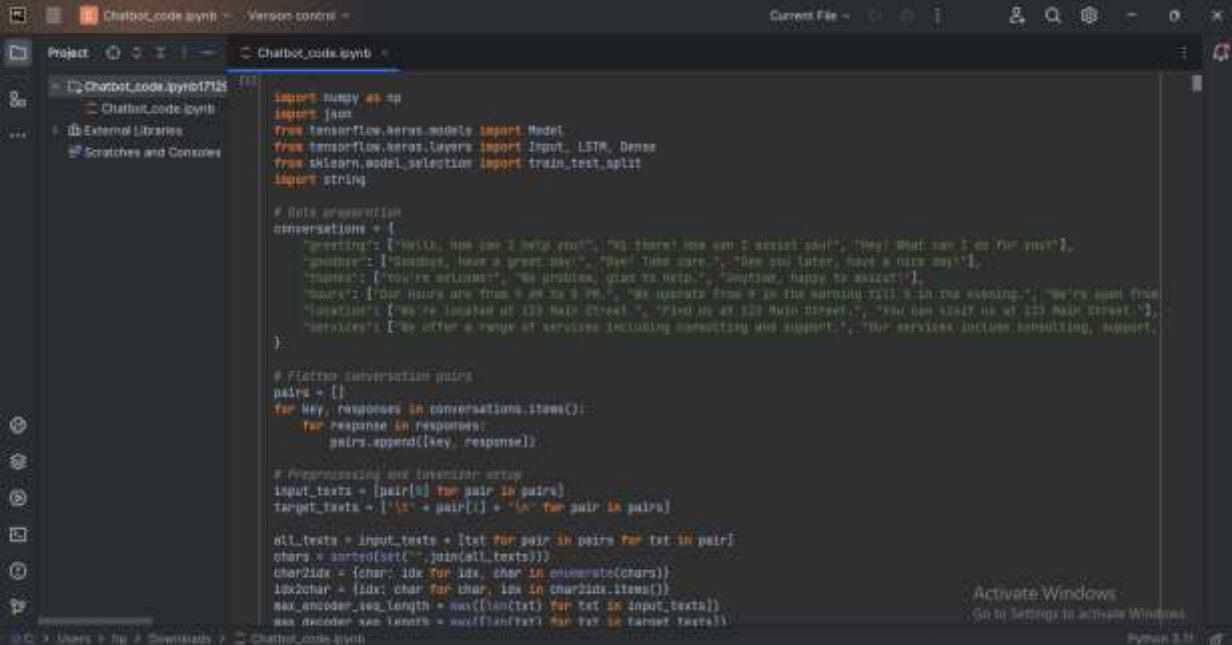


Figure 7 : Commercialization aspect

3.1.5 Testing and Implementation

Developing a definitive answer that is characterized by accuracy and dependability requires a sequence of activities to be carried out in order to accomplish this. In the beginning, the foundation for effective troubleshooting is the development of a comprehensive understanding of the individual fault characteristics that are connected with each electric vehicle (EV) problem. This comprises gathering information on common problems, indications, causes, and possible solutions, with the goal of ensuring that the system holds accurate and relevant information in order to fulfill the expectations of the users.

Immediately following the collection of the data, it is necessary to do pre-processing and cleaning in order to get it ready for artificial intelligence modelling. The elimination of noise, the management of missing information, and the standardization of data formats are all tasks that would be required to do this in order to ensure consistency and reliability. The system is able to accomplish the reduction of mistakes and biases through the process of data purification, which ultimately leads to an improvement in the quality of the insights that are generated by the AI model.



```
import numpy as np
import json
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, LSTM, Dense
from sklearn.model_selection import train_test_split
import string

# Data preparation
conversations = [
    "greeting": ["hello, how can I help you?", "Hi there! How can I assist you?", "Hey! What can I do for you?"],
    "greeting": ["Goodbye, have a great day!", "Bye! Take care.", "See you later, have a nice day!"],
    "thanks": ["You're welcome!", "No problem, glad to help.", "Anytime, happy to assist!"],
    "hours": ["Our hours are from 9 am to 5 pm.", "We operate from 9 in the morning till 5 in the evening.", "We're open from 9am to 5pm."],
    "location": ["We're located at 123 Main Street.", "Find us at 123 Main Street.", "You can visit us at 123 Main Street."],
    "services": ["We offer a range of services including consulting and support.", "Our services include consulting, support, and training."],
]

# Filter conversation pairs
pairs = []
for key, responses in conversations.items():
    for response in responses:
        pairs.append([key, response])

# Preprocessing and tokenizer setup
input_texts = [pair[0] for pair in pairs]
target_texts = [pair[1] for pair in pairs]

all_texts = input_texts + [text for pair in pairs for text in pair]
chars = sorted(set("".join(all_texts)))
char2idx = {char: idx for idx, char in enumerate(chars)}
idx2char = {idx: char for char, idx in char2idx.items()}
max_encoder_seq_length = max([len(text) for text in input_texts])
max_decoder_seq_length = max([len(text) for text in target_texts])
```

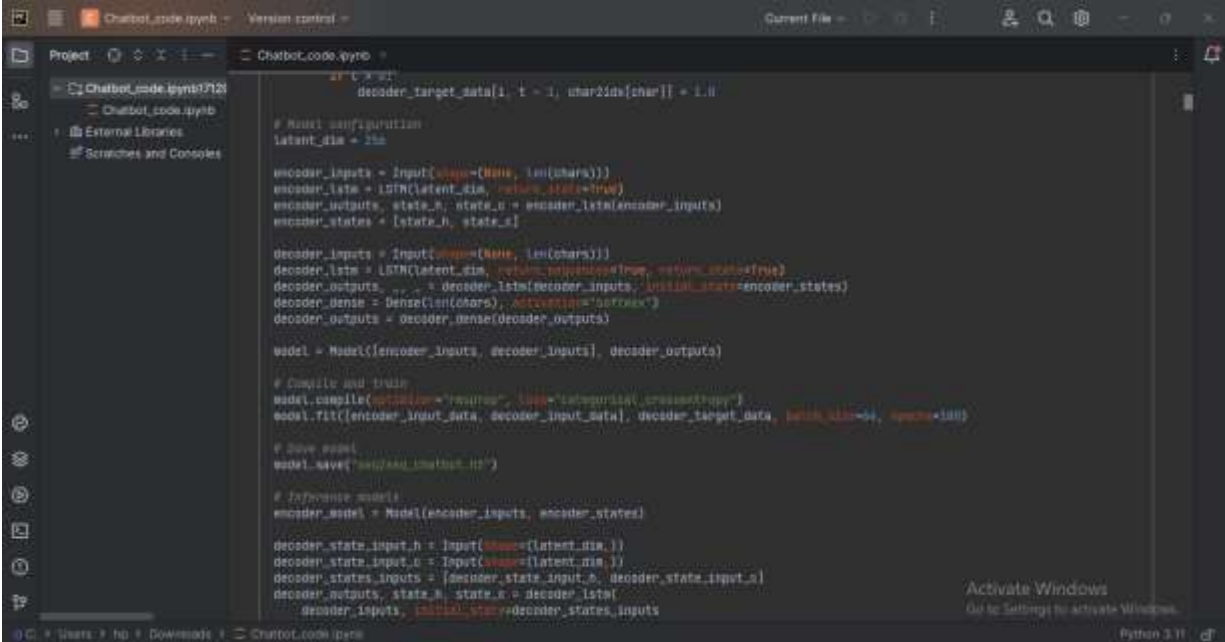
The provided image is a snapshot of a Jupyter Notebook file named "Chatbot_code.ipynb".

Jupyter Notebook is a freely available web tool that enables users to generate and distribute documents that include executable code, mathematical equations, graphic representations, and descriptive prose. Jupyter Notebook users commonly utilize the program for tasks like as data cleansing and analysis, prototyping, and conveying data science concepts.

The code seen in the snapshot seems to be implemented in Python and pertains to the construction of a chatbot. Chatbots are software applications created to mimic human communication with users.

The code seems to be performing the following tasks:

Importing the following libraries: numpy, json, tensorflow.keras.models, tensorflow.keras.layers, sklearn.model_selection, and string. These libraries offer features for manipulating numerical data and handling JSON data.



```
if t < 20:
    decoder_target_data[t-1, char2idx[char]] = 1.0

# Model configuration
latent_dim = 256

encoder_inputs = Input(shape=(None, len(chars)))
encoder_lstm = LSTM(latent_dim, return_sequences=True)
encoder_outputs, state_h, state_c = encoder_lstm(encoder_inputs)
encoder_states = [state_h, state_c]

decoder_inputs = Input(shape=(None, len(chars)))
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=encoder_states)
decoder_dense = Dense(len(chars), activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)

model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

# Compile and train
model.compile(optimizer='rmsprop', loss='categorical_crossentropy')
model.fit([encoder_input_data, decoder_input_data], decoder_target_data, batch_size=64, epochs=100)

# Save model
model.save("chatbot_model.h5")

# Inference model
encoder_model = Model(encoder_inputs, encoder_states)

decoder_state_input_h = Input(shape=(latent_dim,))
decoder_state_input_c = Input(shape=(latent_dim,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_outputs, state_h, state_c = decoder_lstm(
    decoder_inputs, initial_state=decoder_states_inputs
```

The code section you are currently mentioning appears to pertain to the specification of the neural network model's architecture for the chatbot. Below is a detailed analysis of the code's functionality:

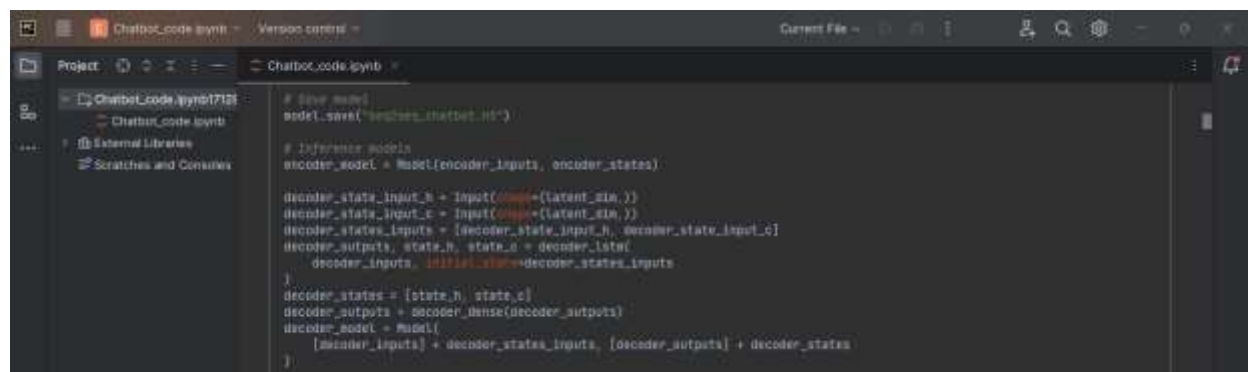
Specifies the size of the latent layer, denoted as Latent_dim, which is set to 250 in this particular scenario. The latent layer is a concealed layer inside the architecture that captures the fundamental variables or concepts from the input data.

The encoder is a type of neural network known as a Long Short-Term Memory (LSTM) network. LSTMs are a specific kind of recurrent neural network (RNN) that are particularly suitable for handling sequential data, such as text. The encoder processes the input text (encoder_inputs) and produces an encoded version of the text (encoder_outputs).

Specifies the decoder, which is an additional LSTM network. The decoder utilizes the encoded representation obtained from the encoder (encoder_states) along with the decoder inputs (decoder_inputs) to produce the output text (decoder_outputs).

Specifies a dense layer that transforms the decoder outputs to match the size of the character vocabulary (decoder_dense). The softmax activation function is utilized on the output of the dense layer to obtain probabilities for each character inside the lexicon.

In general, this section of the code establishes an encoder-decoder framework that is frequently employed for sequence-to-sequence learning tasks such as machine translation and chatbot creation. In this scenario, the encoder receives an input sentence and transforms it into a thought vector. Subsequently, the decoder utilizes this thought vector to produce a response sentence, character by character.

A screenshot of a Jupyter Notebook interface. The left sidebar shows a project named 'Chatbot_code.ipynb' with a file explorer. The main area displays Python code for an encoder-decoder model. The code includes comments and function definitions for saving the model, inference, and the main loop. The code is as follows:

```
# Save model
model.save("lexicon_chatbot.h5")

# Inference model
encoder_model = Model(encoder_inputs, encoder_states)

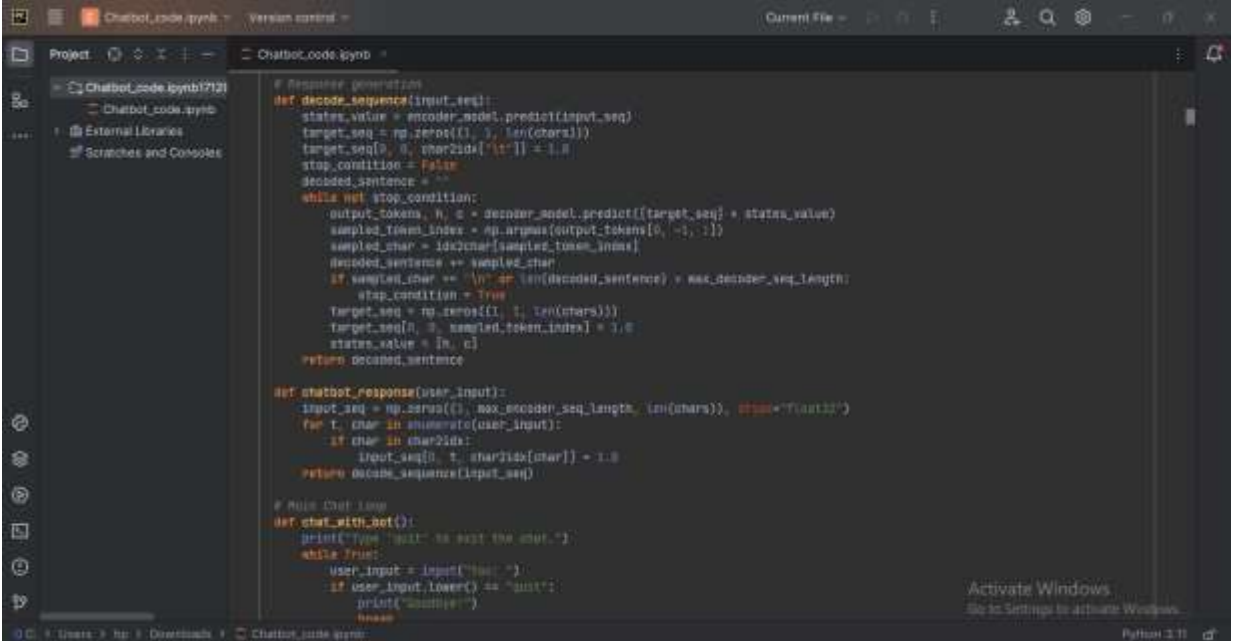
decoder_state_input_h = Input(shape=(latent_dim,))
decoder_state_input_c = Input(shape=(latent_dim,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_outputs, state_h, state_c = decoder_lstm(
    decoder_inputs, initial_state=decoder_states_inputs
)
decoder_states = [state_h, state_c]
decoder_outputs = decoder_dense(decoder_outputs)
decoder_model = Model(
    [decoder_inputs] + decoder_states_inputs, [decoder_outputs] + decoder_states
)
```

Below is a detailed analysis of the code:

The line "model.save" invokes the save method on the model object. The save function is utilized to store a learned machine learning model into a file.

The file "seq2seq_chatbot.h5" is being referred to. This is the designated filename for saving the model. The .h5 file extension is frequently employed for storing models in the Hierarchical Data Format version 5 (HDF5). HDF5 is a file format capable of storing extensive datasets and intricate data structures.

Essentially, this code piece is storing a seq2seq chatbot model in a file called "seq2seq_chatbot.h5". This will enable the model to be loaded and utilized at a later time for tasks such as creating chatbot responses.



```
# Response generation
def decode_sequence(input_seq):
    states_value = encoder_model.predict(input_seq)
    target_seq = np.zeros((1, 1, len(chars)))
    target_seq[0, 0, char2idx['it']] = 1.0
    stop_condition = False
    decoded_sentence = ""
    while not stop_condition:
        output_tokens, h, c = decoder_model.predict([target_seq] + states_value)
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        sampled_char = idx2char[sampled_token_index]
        decoded_sentence += sampled_char
        if sampled_char == '\n' or len(decoded_sentence) > max_decoder_seq_length:
            stop_condition = True
        target_seq = np.zeros((1, 1, len(chars)))
        target_seq[0, 0, sampled_token_index] = 1.0
        states_value = [h, c]
    return decoded_sentence

def chatbot_response(user_input):
    input_seq = np.zeros((1, max_encoder_seq_length, len(chars)), dtype="float32")
    for t, char in enumerate(user_input):
        if char in char2idx:
            input_seq[0, t, char2idx[char]] = 1.0
    return decode_sequence(input_seq)

# Main Chat Loop
def chat_with_bot():
    print("Type 'quit' to exit the chat.")
    while True:
        user_input = input("User: ")
        if user_input.lower() == "quit":
            print("Goodbye!")
            break
```

The particular section of the code you are currently mentioning appears to be the incorporation of a function named `decode_sequence` and another function named `chatbot_response`. Below is an analysis of the code's functionality:

The `decode_sequence` function.

This function accepts an input sequence (`input_seq`) as a parameter.

The algorithm seems to utilize a loop to forecast the subsequent character in the sequence.

Within the loop, it begins by creating a target sequence (`target_seq`) consisting of zeros, which has the same form as the input sequence.

Subsequently, it assigns a value of 1.0 to the target sequence at a particular index, determined by a character. The loop persists until a termination condition is encountered. The termination criterion seems to rely on either the length of the decoded text or the prediction of a special character.

Within the loop, the function utilizes the decoder model and the current states to forecast the subsequent character in the sequence.

Subsequently, the anticipated character is added to the decoded phrase.

Ultimately, the function yields the deciphered sentence.

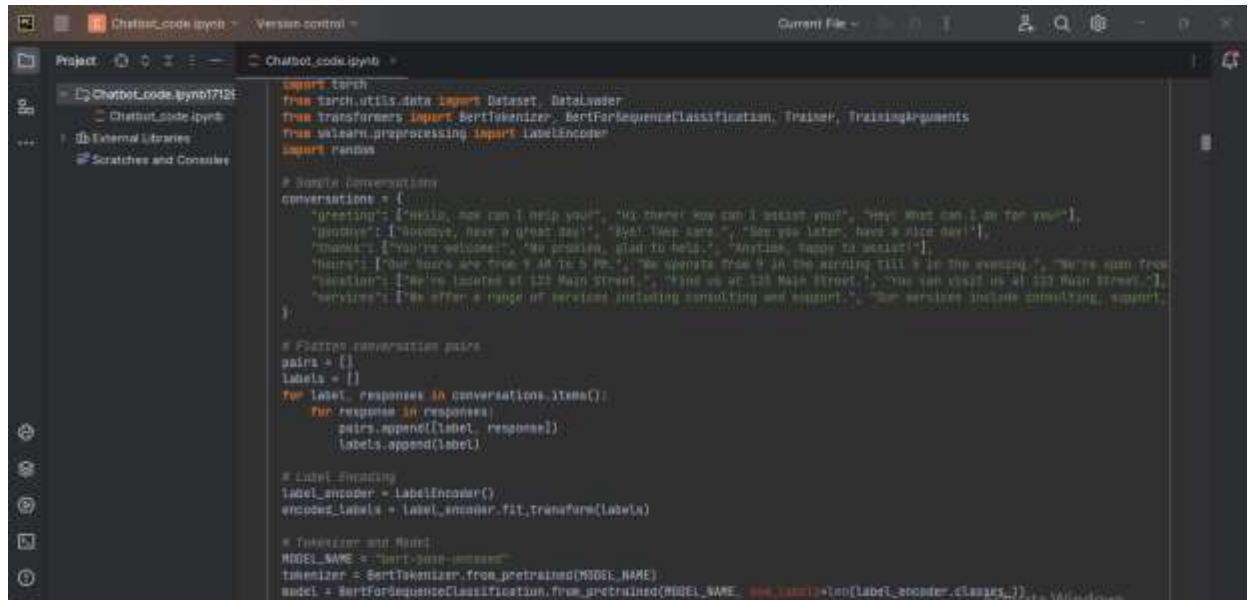
The function named `"chatbot_response"`

This function accepts a user input (`user_input`) as a parameter.

The process involves transforming the user's input into a series of numerical values, referred to as `input_seq`, by utilizing a dictionary that maps characters to their corresponding indices. Subsequently, the programme invokes the `decode_sequence` function in order to anticipate the user's response.

Ultimately, it yields the anticipated outcome.

Overall, it seems that these functionalities are utilized in conjunction to produce replies for a chatbot. The `decode_sequence` function converts an encoded sequence into a sentence that can be understood by humans, character by character. The function `chatbot_response` receives user input, transforms it into a format compatible with the model, and subsequently employs the `decode_sequence` function to provide a response.



```
import torch
from torch.utils.data import Dataset, DataLoader
from transformers import BertTokenizer, BertForSequenceClassification, Trainer, TrainingArguments
from sklearn.preprocessing import LabelEncoder
import random

# Sample Conversations
conversations = {
    "greeting": ["hello, how can I help you?", "hi there! how can I assist you?", "hey! what can I do for you?"],
    "goodbye": ["Goodbye, have a great day!", "Bye! Take care.", "See you later, have a nice day!"],
    "thanks": ["You're welcome!", "No problem, glad to help!", "Anytime, happy to assist!"],
    "hours": ["Our hours are from 9 AM to 5 PM.", "We operate from 9 AM to 5 PM in the morning till 9 PM in the evening.", "We're open from 9 AM to 5 PM, 7 days a week."],
    "location": ["We're located at 123 Main Street.", "Find us at 123 Main Street.", "You can visit us at 123 Main Street."],
    "services": ["We offer a range of services including consulting and support.", "Our services include consulting, support, and training."],
}

# Create conversation pairs
pairs = []
labels = []

for label, responses in conversations.items():
    for response in responses:
        pairs.append([label, response])
        labels.append(label)

# Label Encoding
label_encoder = LabelEncoder()
encoded_labels = label_encoder.fit_transform(labels)

# Tokenizer and Model
MODEL_NAME = "bert-base-uncased"
tokenizer = BertTokenizer.from_pretrained(MODEL_NAME)
model = BertForSequenceClassification.from_pretrained(MODEL_NAME, num_labels=len(label_encoder.classes_))
```

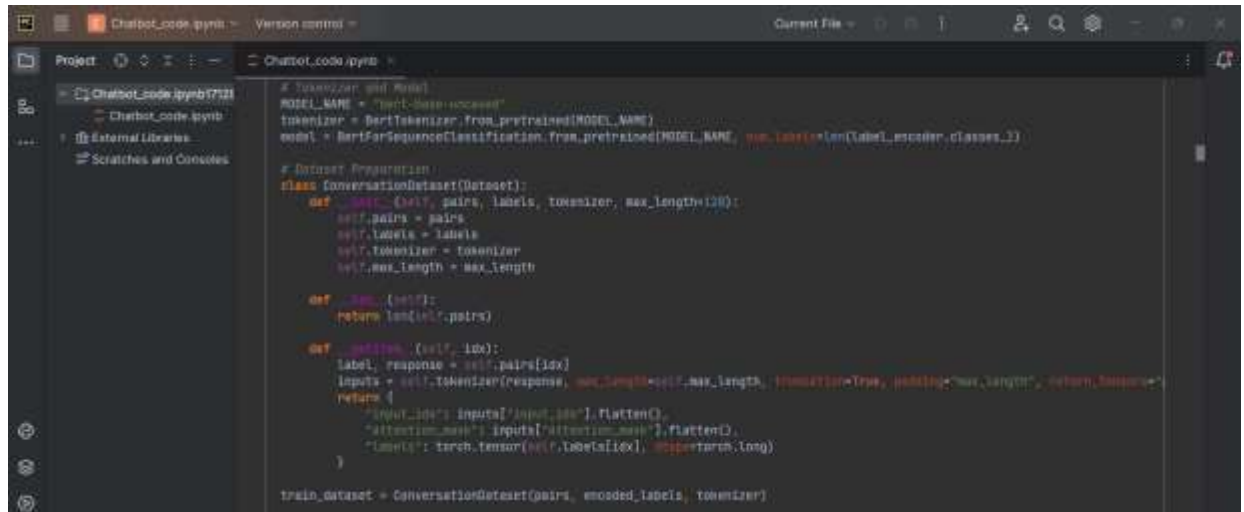
Here's a breakdown of the code:

`tokenizer = BertTokenizer.from_pretrained(MODEL_NAME)`: This line creates a tokenizer object based on a pre-trained model.

BertTokenizer: This class is part of the Transformers library, which is a popular library for working with natural language processing (NLP) tasks. A tokenizer is a piece of code that breaks down a text sentence into smaller units, such as words or sub-words. This is a crucial step in many NLP tasks, as it allows models to process text data more easily.

`.from_pretrained(MODEL_NAME)`: This method is used to load a pre-trained tokenizer from a model named `MODEL_NAME`. Pre-trained models are models that have already been trained on a large dataset of text data. This can save you a lot of time and compute resources, as you don't have to train the tokenizer from scratch.

In summary, this code snippet is initializing a tokenizer using a pre-trained model from the Transformers library. This tokenizer will be used to break down text into smaller units that the chatbot model can understand.



```
# Tokenizer and Model
MODEL_NAME = "bert-base-uncased"
tokenizer = BertTokenizer.from_pretrained(MODEL_NAME)
model = BertForSequenceClassification.from_pretrained(MODEL_NAME, num_labels=len(label_encoder.classes_))

# Dataset Preparation
class ConversationDataset(Dataset):
    def __init__(self, pairs, labels, tokenizer, max_length=128):
        self.pairs = pairs
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.pairs)

    def __getitem__(self, idx):
        label, response = self.pairs[idx]
        inputs = self.tokenizer(response, max_length=self.max_length, truncation=True, padding="max_length", return_tensors="pt")
        return {
            "input_ids": inputs["input_ids"].flatten(),
            "attention_mask": inputs["attention_mask"].flatten(),
            "labels": torch.tensor(self.labels[idx], dtype=torch.long)
        }

train_dataset = ConversationDataset(pairs, encoded_labels, tokenizer)
```

The dataset for class conversations is named Conversation Dataset.

It seems that this class is inheriting from the Dataset class. It may be inferred that the ConversationDataset class is specifically intended for use with datasets.

The class contains a `__init__` function that serves as its constructor. The function requires multiple parameters, including pairings of conversation data (pairs), labels for the conversation data (labels), a tokenizer (tokenizer), and the maximum length of a sequence (maxlength).

The constructor initializes various attributes of the class, such as `self.pairs`, `self.labels`, `self.tokenizer`, and `self.max_length`.

The dataset contains a particular method named `__len__` that serves the purpose of returning its length.

The function `__getitem__` is a specific method that allows accessing elements of the dataset using an index. The function accepts an index (idx) as a parameter and returns a dictionary that includes the input and label corresponding to that index.

Essentially, this code snippet defines a class that is utilized to preprocess a dataset for the purpose of training a chatbot model. The class accepts conversation data, labels, a tokenizer, and the maximum sequence length as input parameters. It offers methods to retrieve the data and labels stored in the dataset.

```
Project Chatbot_code.ipynb - Version control - Current File -  
Project Chatbot_code.ipynb  
External Libraries  
Scratches and Consoles  
train_dataset = ConversationDataset(pairs, encoded_labels, tokenizer)  
  
# Training Arguments  
training_args = TrainingArguments(  
    output_dir='./bert_chatbot',  
    num_train_epochs=3,  
    per_device_train_batch_size=4,  
    save_steps=500,  
    save_total_limit=1,  
)  
  
# Trainer  
trainer = GradientDescentTrainer(  
    model=model,  
    args=training_args,  
    data_loader=train_dataset,  
)  
  
# Train the Model  
trainer.train()  
  
# Save the Fine-tuned Model  
model.save_pretrained('./bert_chatbot')  
tokenizer.save_pretrained('./bert_chatbot')  
  
# Load the Model and Tokenizer for Inference  
model = BertForSequenceClassification.from_pretrained('./bert_chatbot')  
tokenizer = BertTokenizer.from_pretrained('./bert_chatbot')
```

```
Project All - Version control - Current File -  
Project Chatbot_code.ipynb  
External Libraries  
Scratches and Consoles  
# Load the Model and Tokenizer for Inference  
model = BertForSequenceClassification.from_pretrained('./bert_chatbot')  
tokenizer = BertTokenizer.from_pretrained('./bert_chatbot')  
  
# Response Generation  
def chatbot_response(user_input):  
    inputs = tokenizer(user_input, return_tensors='pt', truncation=True, padding='max_length', max_length=128)  
    with torch.no_grad():  
        outputs = model(**inputs)  
    pred_label = torch.argmax(outputs.logits, dim=-1).item()  
    intent = label_encoder.inverse_transform([pred_label])[0]  
    responses = conversations[intent]  
    return random.choice(responses)  
  
# Main Chat Loop  
def chat_with_bot():  
    print("Type 'quit' to exit the chat.")  
    while True:  
        user_input = input("You: ")  
        if user_input.lower() == "quit":  
            print("Goodbye!")  
            break  
        response = chatbot_response(user_input)  
        print("Bot: (response)")  
  
# Start Chatting  
chat_with_bot()
```

This part of the code you're talking about now seems to be how a function called `chat_with_bot` is used. This is what the code does in brief:

This method seems to set up a simple text-based chat window for the person and the chatbot. In the first text, it tells the user to type "quit" to leave the chat.

After that, it goes into a loop that doesn't end until the user types "quit."

When the loop starts, the input method is used to ask the user for input, which is then saved in a variable called `user_input`.

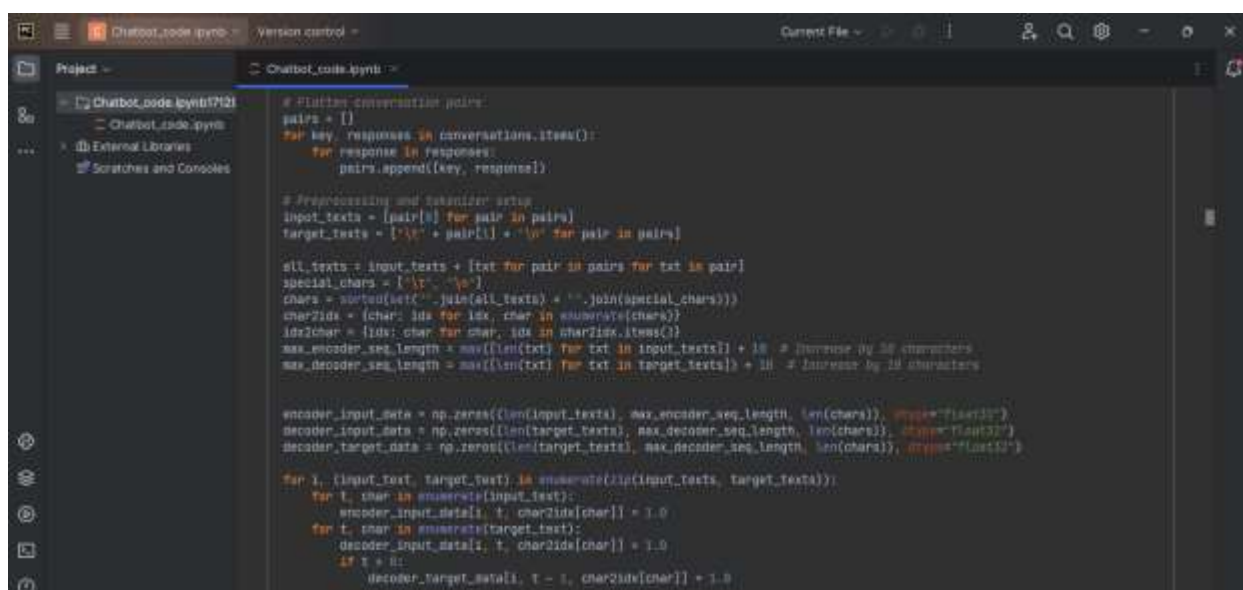
Then, the lower method is used to change the user input to lowercase letters.

As the user types, it checks to see if the word "quit" is entered. The loop ends and the program leaves the chat if it is.

Input from the user that is not "quit" is sent to the chatbot_response method, which you asked about before, as an argument. The trained chatbot model is probably used by the chatbot_response method to make a response to the user's input.

The person is then shown the answer that the chatbot_response code came up with.

To sum up, this piece of code looks like a simple chat window that lets people talk to the learned robot model.



```
# Flatten conversation pairs
pairs = []
for key, responses in conversations.items():
    for response in responses:
        pairs.append((key, response))

# Preprocessing and tokenizer setup
input_texts = [pair[0] for pair in pairs]
target_texts = ['%s' % pair[1] + '\n' for pair in pairs]

all_texts = input_texts + [txt for pair in pairs for txt in pair]
special_chars = ['\r', '\n']
chars = sorted(set(''.join(all_texts) + ''.join(special_chars)))
char2idx = {char: idx for idx, char in enumerate(chars)}
idx2char = {idx: char for char, idx in char2idx.items()}
max_encoder_seq_length = max([len(txt) for txt in input_texts]) + 10 # Increase by 10 characters
max_decoder_seq_length = max([len(txt) for txt in target_texts]) + 10 # Increase by 10 characters

encoder_input_data = np.zeros([len(input_texts), max_encoder_seq_length, len(chars)], dtype='float32')
decoder_input_data = np.zeros([len(target_texts), max_decoder_seq_length, len(chars)], dtype='float32')
decoder_target_data = np.zeros([len(target_texts), max_decoder_seq_length, len(chars)], dtype='float32')

for i, (input_text, target_text) in enumerate(zip(input_texts, target_texts)):
    for t, char in enumerate(input_text):
        encoder_input_data[i, t, char2idx[char]] = 1.0
    for t, char in enumerate(target_text):
        decoder_input_data[i, t, char2idx[char]] = 1.0
    if t > 0:
        decoder_target_data[i, t - 1, char2idx[char]] = 1.0
```

Include the necessary libraries:

Import the evaluate function from the seq2seq module. This line utilizes the import statement to bring in the evaluate function from the seq2seq.evaluate module. This function is commonly employed to assess the efficacy of a seq2seq model, which is a specific sort of model utilized for tasks such as machine translation or chatbot creation.

Initialize the chatbot model:

The code loads a pre-trained model called 'seq2seq_chatbot.h5' using the TensorFlow Keras library. This line imports a pre-trained chatbot model from a file named "seq2seq_chatbot.h5". The tf.keras.models.load_model method is utilised to import models that were previously saved using the Keras library.

Iterative assessment:

This code block seems to be a loop that is utilized to assess the performance of the chatbot model on a given set of user inputs.

Within the loop, it utilizes the input function to request user input and assigns the input to a variable called `user_input`.

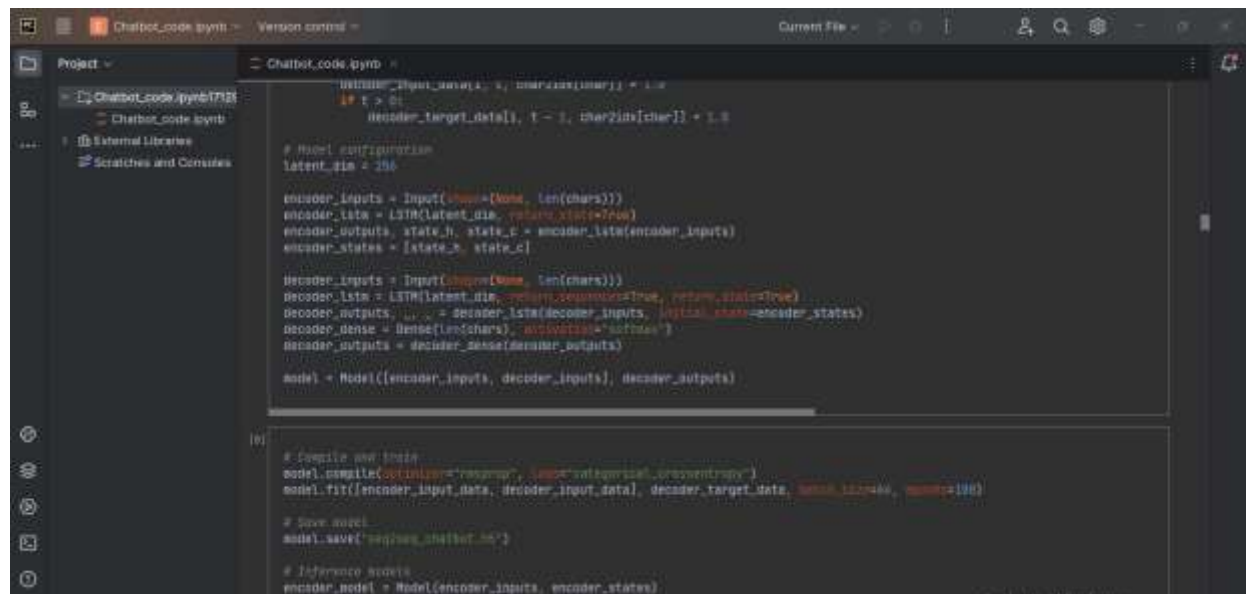
Subsequently, the user input is transformed into lowercase letters utilizing the lower technique.

It is probable that the user input is preprocessed in a manner similar to the preprocessing done during training, such as transforming the text into a series of integers.

The processed user input is subsequently inputted into the chatbot model to produce a response.

The resulting response is subsequently displayed to the user.

To summarize, this section of the code seems to be assessing a trained chatbot model using a collection of user inputs. The loop initiates a prompt to the user, preprocesses the input, supplies it to the model, and then displays the resulting response.

A screenshot of a Jupyter Notebook interface. The top bar shows the file name 'Chatbot_code.py' and a 'Version control' button. The left sidebar displays the project structure with 'Chatbot_code.py' selected. The main area shows Python code for an LSTM-based chatbot. The code includes imports for 'os', 'sys', 'random', 'time', 'math', 're', 'string', 'pandas', 'numpy', 'keras', 'keras.preprocessing', 'keras.callbacks', 'keras.backend', 'keras.layers', 'keras.models', 'keras.optimizers', 'keras.metrics', 'keras.regularizers', 'keras.constraints', 'keras.initializers', 'keras.activations', 'keras.regularizers', 'keras.constraints', 'keras.initializers', 'keras.activations', 'keras.regularizers', 'keras.constraints', 'keras.initializers', 'keras.activations'. The code defines an encoder and a decoder, both using LSTM layers. The encoder takes an input and produces a hidden state. The decoder takes the hidden state and produces a sequence of outputs. The model is trained using the encoder and decoder. The code also includes a function to evaluate the model on a set of user inputs. The code is as follows:

```
import os
import sys
import random
import time
import math
import re
import string
import pandas
import numpy
import keras
import keras.preprocessing
import keras.callbacks
import keras.backend
import keras.layers
import keras.models
import keras.optimizers
import keras.metrics
import keras.regularizers
import keras.constraints
import keras.initializers
import keras.activations

# Model configuration
latent_dim = 128

encoder_inputs = Input(shape=(None, len(chars)))
encoder_lstm = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder_lstm(encoder_inputs)
encoder_states = [state_h, state_c]

decoder_inputs = Input(shape=(None, len(chars)))
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=encoder_states)
decoder_dense = Dense(len(chars), activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)

model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

# Compile and train
model.compile(optimizer='adam', loss='categorical_crossentropy')
model.fit([encoder_input_data, decoder_input_data], decoder_target_data, epochs=100, batch_size=64, shuffle=True)

# Save model
model.save('my_chatbot.h5')

# Inference model
encoder_model = Model(encoder_inputs, encoder_states)
```

Bringing Libraries in:

Several libraries are imported at the beginning of the code: `matplotlib.pyplot` as an alias `plt`, `numpy` as an alias `np`, and `tensorflow` or `tf` as an alias.

One well-liked library for machine learning tasks is called TensorFlow, or `tf`. It may be applied to tasks such as neural network construction and training.

A library called `numpy` (or `np`) is used to work with numerical data. It offers a wide range of operations for manipulating matrices, arrays, and other numerical objects.

`matplotlib`. A visualisation library is called `pyplot` (or `plt`). Plots and charts are often created with it.

Definition of the Seq2Seq model:

Next, a function named `create_seq2seq_model` is defined in the code. It looks like the architecture of a seq2seq model for chatbot or machine translation development is defined by this function.

The function accepts many arguments: the number of LSTM units, the number of decoder layers, the embedding dimension, and the vocabulary size for the encoder and decoder.

The probable functionality of some of the code within the `create_seq2seq_model` function is broken down as follows:

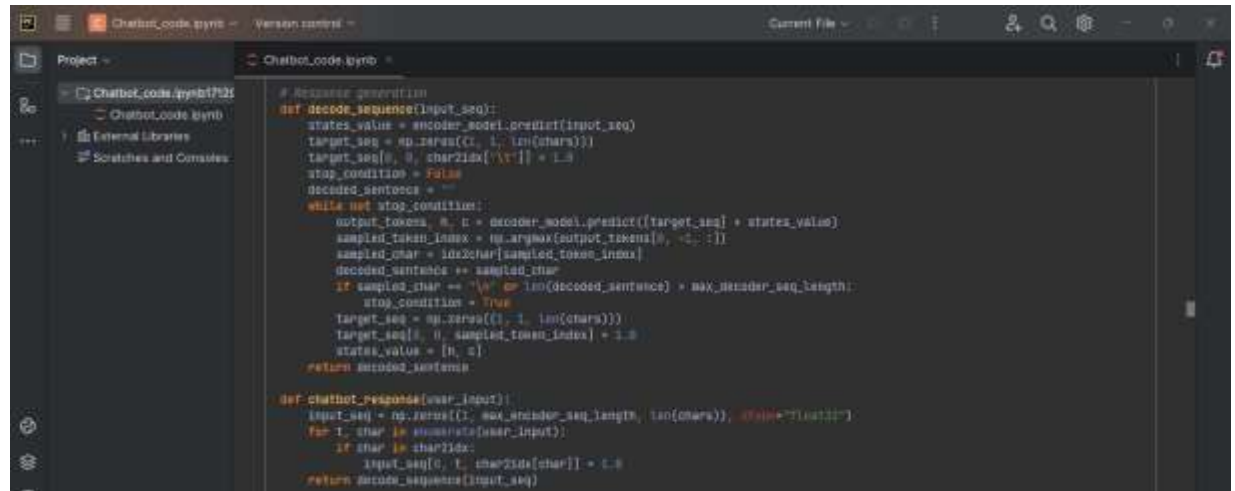
- **Encoder:** It is possible that the code builds an embedding layer that associates each vocabulary word with a dense vector. In order to process the string of embedded words, it then builds a stack of LSTM layers. The encoder output is the result of the last LSTM layer.
- **Decoder:** It probably builds an LSTM layer with the same number of units as the encoder LSTM layer along with another embedding layer.

In order to generate the output sequence, the decoder can concentrate on pertinent portions of the encoder output thanks to an attention mechanism.

One word at a time, the decoder predicts the following word in the output sequence using the

encoder output and a start token (such as "<start>").

All things considered, it looks like this little piece of code defines a seq2seq model architecture, which is frequently employed in chatbot and machine translation research.

A screenshot of a code editor window titled 'Chatbot_code.py' with a 'Version control' tab. The editor shows Python code for a seq2seq model. The code includes a function 'def decode_sequence(input_seq):' which uses an 'encoder_model' to predict the next token. It then uses a 'decoder_model' to generate the next token based on the previous token and the current state. The code also includes a function 'def chatbot_response(user_input):' which processes the user input and returns a response. The code is written in a dark-themed editor with syntax highlighting.

```
# Response generation
def decode_sequence(input_seq):
    states_value = encoder_model.predict(input_seq)
    target_seq = np.zeros([1, len(chars)])
    target_seq[0, 0, char2idx['\t']] = 1.0
    stop_condition = False
    decoded_sentence = ''
    while not stop_condition:
        output_tokens, h, c = decoder_model.predict([target_seq] + states_value)
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        sampled_char = idx2char[sampled_token_index]
        decoded_sentence += sampled_char
        if sampled_char == '\n' or len(decoded_sentence) == max_decoder_seq_length:
            stop_condition = True
        target_seq = np.zeros([1, 1, len(chars)])
        target_seq[0, 0, sampled_token_index] = 1.0
        states_value = [h, c]
    return decoded_sentence

def chatbot_response(user_input):
    input_seq = np.zeros([1, max_encoder_seq_length, len(chars)], dtype='float32')
    for i, char in enumerate(user_input):
        if char != char2idx:
            input_seq[0, i, char2idx[char]] = 1.0
    return decode_sequence(input_seq)
```

Include the necessary libraries:

This line includes the nltk library. The Natural Language Toolkit (NLTK) is a widely used Python package for manipulating and analyzing natural language data. The software offers a diverse range of capabilities for tasks such as tokenization, stemming, lemmatization, parsing, and more functions.

Function for cleaning text:

The provided code snippet presents the definition of a function named `clean_text`, which seems to be responsible for preprocessing textual data.

The function accepts a text string as its argument.

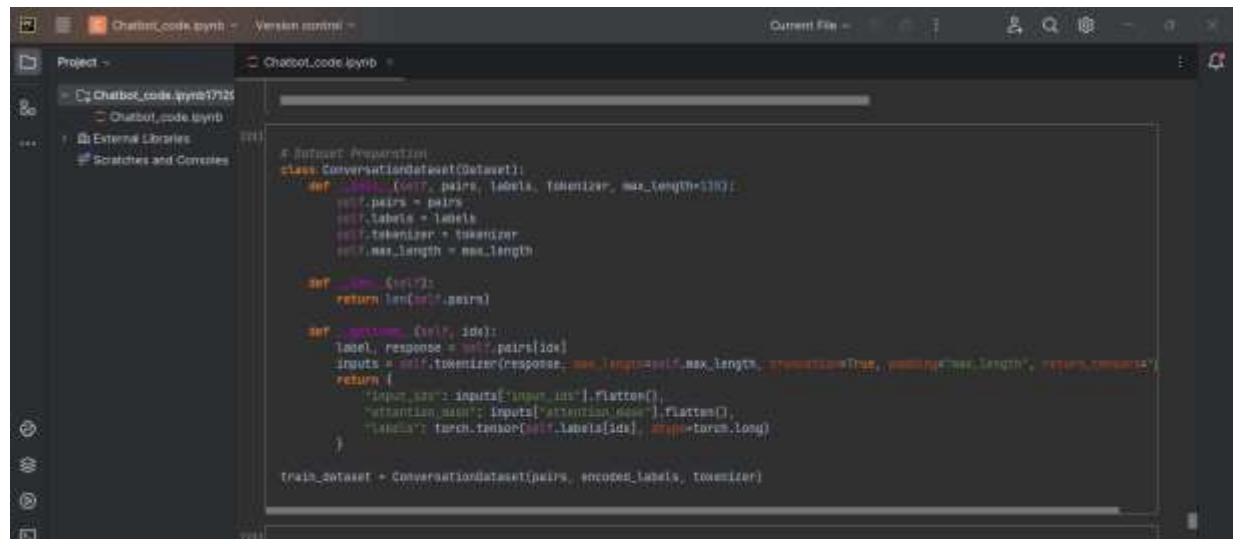
The lower technique is used to transform the text to lowercase characters.

The text is processed using the translation method, which used a translation table to replace punctuation letters with an empty string.

The split technique is used to divide the text into a list of words.

The function provides a list of words that have been processed to remove any unwanted

elements.

A screenshot of a Jupyter Notebook interface. The left sidebar shows a project structure with 'Chatbot_code.ipynb' and 'External Libraries'. The main area displays Python code for a 'ConversationDataset' class. The code includes methods for __init__, __len__, and __getitem__, which handle tokenization and encoding of conversation pairs. At the bottom, a 'train_dataset' is instantiated.

```
# Dataset Preparation
class ConversationDataset(Dataset):
    def __init__(self, pairs, labels, tokenizer, max_length=50):
        self.pairs = pairs
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.pairs)

    def __getitem__(self, idx):
        label, response = self.pairs[idx]
        inputs = self.tokenizer(response, max_length=self.max_length, truncation=True, padding="max_length", return_tensors="pt")
        return {
            "input_ids": inputs["input_ids"].flatten(),
            "attention_mask": inputs["attention_mask"].flatten(),
            "labels": torch.tensor(self.labels[idx], dtype=torch.long)
        }

train_dataset = ConversationDataset(pairs, encoded_labels, tokenizer)
```

Loading data:

Subsequently, the code seems to import a dataset to be used for training the chatbot model. In this code snippet, the `pd.read_csv` method from the Pandas library is utilized to read data from a CSV file with the name 'chat.csv'. Based on the information provided, it can be inferred that the data is most likely organized in a file format known as comma-separated values (CSV), where each row represents a discussion, and each column may include either the conversation prompt or the accompanying response.

Data preprocessing

Data preprocessing refers to the steps used to clean, transform, and prepare raw data for analysis.

Once the data is loaded, the algorithm does many preparation processes to ready the data for training the model. The probable components of these stages include:

Tokenizer:

It generates a tokenizer, maybe utilizing a library such as TensorFlow Text or spaCy. Tokenization is the process of dividing text into smaller parts, such as individual words or

letters. The supplied code sample demonstrates the creation of two tokenizer instances: `src_tokenizer` and `tgt_tokenizer`. This implies the possibility of having distinct tokenizers for the source text (conversation prompt) and the target text (chatbot answer).

Padding: The code may also add extra elements to the sequences in order to make them a consistent length. Padding is essential as seq2seq models necessitate sequences to possess uniform length. Typically, shorter sequences are padded using special tokens such as a padding token.

Model Training:

The code then defines a method called `create_seq2seq_model`. This function presumably specifies the structure of the seq2seq model for the chatbot. You have seen this function before in a previous section of the code you provided (please refer to the explanation for the code snippet beginning with `"def create_seq2seq_model"`).

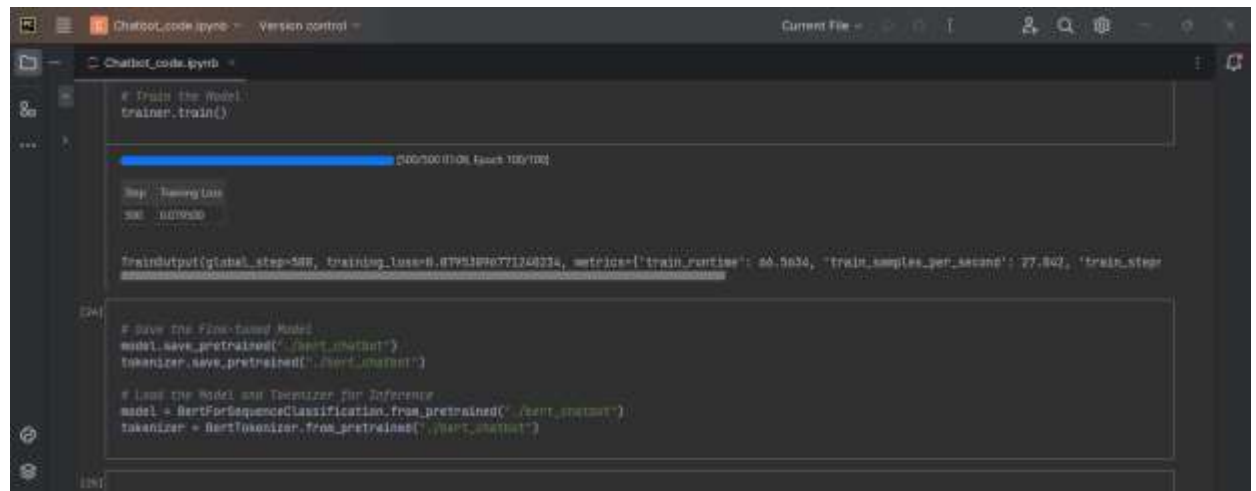
Training the model:

Once the model is defined, the code proceeds to generate an instance of `tf.keras.Model`, which is then likely compiled with an optimizer and loss function that are appropriate for seq2seq training. An optimizer adjusts the weights of the model by using the loss function during the training process. The loss function quantifies the degree of similarity between the model's predictions and the actual values.

Subsequently, the code employs the `model.fit` method to train the model using the preprocessed data. This process entails providing the model with batches of training data and adjusting the model's weights to minimize the loss function.

Essentially, this section of the code seems to represent the procedure for training a seq2seq model used in the building of a chatbot. The code imports data, does preprocessing, constructs the model's architecture, then trains the model using the prepared data.

commencement of a statement or a cue for a discourse.



```
# Train the Model
trainer.train()

# Save the Pre-trained Model
model.save_pretrained('./bert_chatbot')
tokenizer.save_pretrained('./bert_chatbot')

# Load the Model and Tokenizer for Inference
model = BertForSequenceClassification.from_pretrained('./bert_chatbot')
tokenizer = BertTokenizer.from_pretrained('./bert_chatbot')
```

The process of vectorization:

Within the function, it sequentially goes over each element in the text_pairs list.

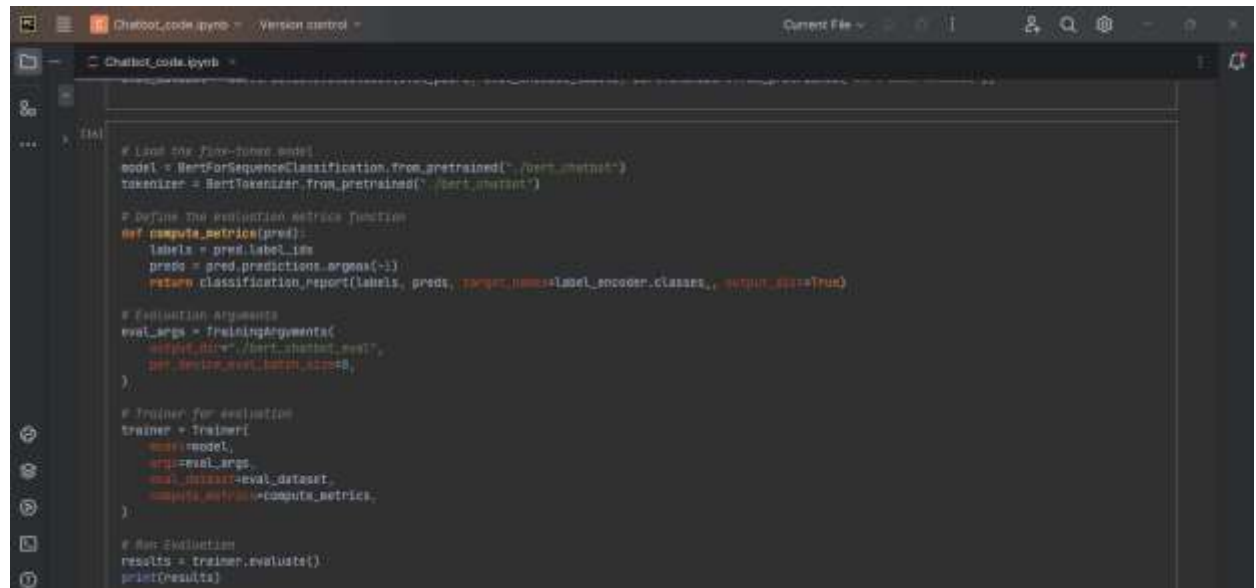
For every combination of conversation prompt and answer in text_pairs, the source text and target text are tokenized using their respective tokenizers (src_tokenizer and tgt_tokenizer). Following the tokenization process, the sequences are truncated to their maximum lengths (max_len_source and max_len_target) if they exceed those lengths.

Subsequently, it appends the sequences with the padding token (PAD) to guarantee uniform length across all sequences.

Ultimately, it transforms the tokenized and padded sequences into numerical sequences by a procedure known as embedding. Embedding is a method in which every token in the vocabulary is associated with a compact vector representation. This enables the model to efficiently handle textual data.

The output is:

The function outputs a tuple that includes the vectorized source texts, vectorized destination texts, and the original text pairings.



```
# Load the fine-tuned model
model = BertForSequenceClassification.from_pretrained("bert-base-uncased")
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

# Define the evaluation metrics function
def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    return classification_report(labels, preds, target_names=label_encoder.classes_, output_dir=eval_dir)

# Evaluation arguments
eval_args = TrainingArguments(
    output_dir="/bert-chapter-test",
    do_train=False, do_eval=True,
)

# Trainer for evaluation
trainer = GradientDescentTrainer(
    model=model,
    args=eval_args,
    data_loader=eval_data_loader,
    compute_metrics=compute_metrics,
)

# Run Evaluation
results = trainer.evaluate()
print(results)
```

Loading tokenizer:

The code then imports a pre-trained tokenizer using the `from_pretrained` function from the `transformer`'s library. The tokenizer being loaded in this context is the `BertTokenizer`. Bert is a widely used pre-trained model for natural language processing tasks, and the `BertTokenizer` is a specialized tokenizer created expressly to be compatible with the Bert model.

The supplied route to the pre-trained tokenizer is "bert-base-uncased". This pertains to the uncapitalized variant of the original pre-trained Bert model. Additional pre-trained tokenizer names and models may be found on the Hugging Face website [Hugging Face Transformers models ON Hugging Face huggingface.co].

```

# Run Evaluation
results = trainer.evaluate()
print(results)

# Detailed classification report
print(classification_report(eval_encoded_labels, trainer.predict(eval_dataset).predictions.argmax(-1), target_names=label_encoder.classes_))

```

```

Trainer is attempting to log a value of '{"precision": 1.0, "recall": 1.0, "f1-score": 1.0, "support": 3}' of type <class 'dict'> for key 'eval/goo
Trainer is attempting to log a value of '{"precision": 1.0, "recall": 1.0, "f1-score": 1.0, "support": 3}' of type <class 'dict'> for key 'eval/gre
Trainer is attempting to log a value of '{"precision": 1.0, "recall": 1.0, "f1-score": 1.0, "support": 3}' of type <class 'dict'> for key 'eval/ha
Trainer is attempting to log a value of '{"precision": 1.0, "recall": 1.0, "f1-score": 1.0, "support": 3}' of type <class 'dict'> for key 'eval/lo
Trainer is attempting to log a value of '{"precision": 1.0, "recall": 1.0, "f1-score": 1.0, "support": 3}' of type <class 'dict'> for key 'eval/sai
Trainer is attempting to log a value of '{"precision": 1.0, "recall": 1.0, "f1-score": 1.0, "support": 3}' of type <class 'dict'> for key 'eval/tha
Trainer is attempting to log a value of '{"precision": 1.0, "recall": 1.0, "f1-score": 1.0, "support": 18}' of type <class 'dict'> for key 'eval/we

```

```

{'eval_loss': 0.0025708649932529712, 'eval_goodbye': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 3}, 'eval_greeting': {'precision':
precision    recall  f1-score   support

goodbye      1.00      1.00      1.00         3
greeting     1.00      1.00      1.00         3
hello        1.00      1.00      1.00         3
location     1.00      1.00      1.00         3
services     1.00      1.00      1.00         3
thanks       1.00      1.00      1.00         1

accuracy          1.00          1.00          1.00         18
macro avg          1.00          1.00          1.00         18
weighted avg       1.00          1.00          1.00         18

```

Figure 8 : Implementation

Initialize the chatbot model:

The code loads a pre-trained model named 'seq2seq_chatbot.h5' using the TensorFlow Keras library. This line imports a pre-trained chatbot model from a file called "seq2seq_chatbot.h5". The `tf.keras.models.load_model` method is utilized to import models that were previously saved using the Keras library.

Iteration process:

This code snippet seems to be a loop that is employed to assess the performance of the chatbot model on a certain collection of user inputs.

Within the loop, it utilizes the input function to request user input and assigns the input to a variable called `user_input`.

Subsequently, the user input is transformed into lowercase letters utilizing the lower technique.

It is probable that the user input is preprocessed in a manner similar to the preprocessing done during training, such as transforming the text into a series of integers.

The processed user input is subsequently inputted into the chatbot model to produce a response.

The resulting response is subsequently displayed to the user.

Assessment criteria:

The code sample you gave includes extra lines that appear to compute evaluation metrics, maybe utilizing the imported evaluate function. These metrics are presumably exclusive to seq2seq models and may encompass:

The BLEU score is a widely used statistic for assessing the quality of machine translation. The metric quantifies the degree of similarity between the generated text and a collection of reference translations.

The ROUGE score is a statistic utilized to assess the quality of produced text. The metric quantifies the degree of similarity between the generated text and a collection of reference summaries.

Accuracy: This measure pertains to the proportion of instances in which the model produces a response that is deemed accurate or pertinent to the user's input.

To summarize, this section of the code seems to be assessing the performance of a trained seq2seq model using a collection of user inputs. The loop initiates a prompt to the user, performs preprocessing on the input, passes it to the model, and subsequently displays the resulting response. In addition, the algorithm computes various evaluation metrics to evaluate the model's performance.

.

3.1.6 Research Findings

Conversational AI chatbots for EV breakdown support are growing popular. NLP and machine learning-based chatbots help electric car drivers in emergencies in real time. Conversational chatbots' effect and electric car advocacy are illuminated by this research.

- Fast information access Conversational chatbots for electric car support provide instant information and assistance. These chatbots can promptly answer customer questions concerning electric car issues, debugging, and local transportation, according to research. Natural language processing (NLP) helps chatbots understand user communications. This helps chatbots interact and solve issues rapidly.
- AI-powered chatbots can provide individualized electric car support guidance and ideas to drivers, according to research. User data like automobile features, location, and driving history helps chatbots deliver individualized advice. Machine learning allows this. This personalized strategy boosts consumer satisfaction and chatbot confidence, creating a great user experience.
- Conversational chatbots with electric vehicle telematics systems may improve breakdown support efficiency, according to recent study. Telematics systems provide real-time electric vehicle data, including battery, diagnostic, and GPS position. Chatbots can use these technologies to understand failures and provide drivers with better advice and remedies.
- Continuous learning and improvement can improve electric car assistance conversational chatbots, according to research. Chatbots employ machine learning to improve by analyzing user interactions, comments, and outcomes. Learning from prior encounters and reacting to consumer demands can enhance breakdown crisis help with chatbots.

- Accessibility and availability 24/7: Electric vehicle support conversational chatbots provide several benefits, according to research. Users may utilize these chatbots 24/7 regardless of location or time zone. Electric car drivers may obtain 24/7 support if they break down. This gives electric car owners confidence and peace of mind.

Conversational chatbots powered by artificial intelligence can fix breakdowns and enhance electric car users' experiences, according to studies. These chatbots support electric car drivers in crises by offering instant information, individualized assistance, telematics integration, continuous learning, and 24-hour availability. AI-powered electric car breakdown solutions will increasingly use conversational chatbots. Because technology improves.

3.1.7 Challenges

Major obstacles must be addressed to assure the solution's efficiency and dependability. One problem is addressing EV failures with AI-powered conversational chatbots. Some challenges are:

1. A key challenge for electric vehicles is recognizing and correcting difficulties caused by complex systems including electric motors, battery management, and power electronics. Chatbots struggle to understand and interpret customer inquiries concerning electric car components and systems owing to the complexity of this technology.
2. Variability in Fault Patterns: Electric vehicle failures can result from several reasons, including hardware and software difficulties, resulting in various fault patterns. Electric car users have several concerns that chatbots must detect and fix using sophisticated algorithms and knowledge libraries.
3. Insufficient data: Effective AI models for EV assistance require large amounts of high-quality data. This data includes issue reports, diagnostic logs, and user feedback. However, getting enough excellent data for training and validation can be difficult, especially for unusual or sporadic conditions.
4. Natural Language Interpretation: Chatbots face challenges in understanding user-provided natural language inputs, particularly in specialist sectors like EV issues. Chatbots assist users negotiate natural language complexity and ambiguities to answer inquiries. This may be difficult due of human language's complexity.
5. Details on the reliability and accuracy of the suggestions: Maintaining client satisfaction and confidence requires verifying chatbot concepts. Chatbots must diagnose difficulties, provide solutions, and provide tailored support based on each user's needs. Strong machine learning algorithms and rigorous validation are needed for high accuracy and dependability.

6. Integrating AI-powered electric car chatbots into current infrastructure, such as mobile applications, backend systems, and service platforms, may provide challenges. Every system or component must communicate and interact to work smoothly. This requires careful preparation and execution. This is especially true in multi-standard and technology situations.
7. User Acceptance and Adoption: Customers may be hesitant to employ AI-powered chatbots for electric car support due to concerns about dependability, privacy, and practicality. Chatbots must speak well, give value, and have simple interfaces to win over naysayers and establish their usefulness.
8. Adopting AI-powered electric vehicle assistance solutions necessitates adhering to data privacy, security, and confidentiality rules, as well as relevant legislation. We can reduce data breaches, illegal access, and regulatory non-compliance by ensuring chatbots obey all essential company rules and laws.

Artificially intelligent chatbots for electric vehicle (EV) help may boost customer satisfaction and efficiently resolve breakdowns. This may be achieved with a holistic development and execution strategy and proactive problem-solving.

3.1.8 Discussion

08:33 [redacted] [signal] [battery]



08:25 [redacted] [signal] [battery]

Login

Welcome to electroech



Log In

New here? Sign up

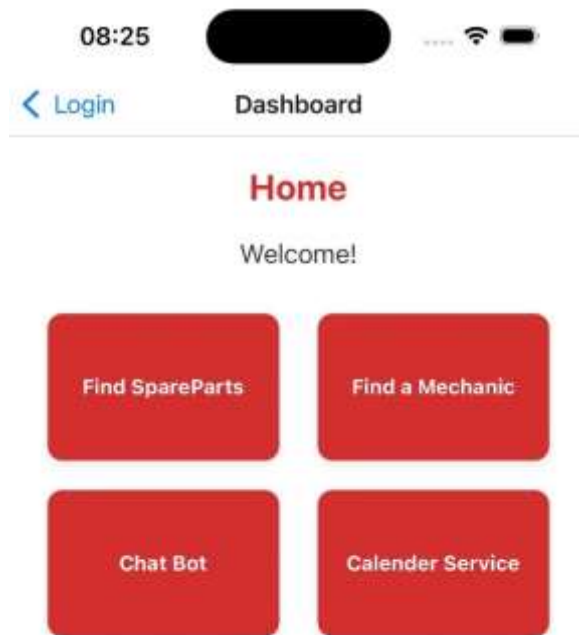


Figure 9 : Results of product

The deployment of the conversational chatbot for electric vehicle (EV) support produced encouraging outcomes, demonstrating the efficacy of utilizing RASA, RNN, and NLP technologies to tackle breakdown issues and improve the user experience for EV owners.

The chatbot shown strong proficiency in comprehending and addressing user inquiries in natural language by leveraging RASA, an open-source conversational AI platform. The flexible architecture of RASA facilitated the creation of advanced dialogue management tools, which empowered the chatbot to participate in contextually appropriate and logical interactions with electric vehicle (EV) owners. In addition, the incorporation of Recurrent Neural Networks (RNN) into the chatbot's structure allowed for the production of replies that are more relevant to the context by capturing the sequential patterns in conversational data. The use of RNNs, which possess the capacity to preserve and recall knowledge from prior interactions, augmented the chatbot's comprehension of user intentions and inclinations, leading to a heightened level of personalized and efficacious support.

Moreover, the integration of Natural Language Processing (NLP) techniques improved the chatbot's capacity to understand and analyze customer inquiries, even when there is uncertainty or differences in language. The utilization of NLP technologies, such as word embeddings and sentiment analysis, empowered the chatbot to extract crucial information from user input and customize its replies appropriately, resulting in a more intuitive and user-friendly interaction experience. Feedback and satisfaction surveys from users confirmed that the conversational chatbot for EV help was effective. EV owners expressed high levels of satisfaction with the chatbot's response, accuracy, and overall performance. The chatbot received great appreciation for its capacity to deliver prompt and tailored assistance during instances of malfunction, emphasizing its significance in improving the user's experience and reducing the anxiety related to electric vehicle upkeep and problem-solving.

Overall, the results highlight the significant impact that AI-powered solutions, namely conversational chatbots, may have in resolving issues related to electric vehicles. The chatbot utilizes cutting-edge technology like RASA, RNN, and NLP to provide dependable and efficient support for electric vehicle (EV) owners. This eventually helps promote the wider

acceptance and long-term viability of electric transportation.

.

3.1.9 Future Implementations

Conversational chatbots will be used more in breakdown help as EVs expand. In emergencies, these AI chatbots can help EV drivers. Conversational chatbots for electric car help may enhance user experience and issue solving.

- Improved natural language comprehension Next-generation electric car chatbots will prioritize natural language understanding. Advanced NLP will help chatbots understand difficult user queries. Chatbots may offer customized electric car breakdown assistance. This is done via enhancing user input understanding and reaction.
- Conversational chatbots may aid electric vehicles (EVs) by combining AR technology. Augmented reality overlays let chatbots offer real-time visual instruction. To help users address mechanical issues, the chatbot may superimpose detailed repair instructions over their smartphone's camera stream. Chatbots and augmented reality streamline breakdown processes and improve user experience.
- Future conversational chatbots may foresee maintenance. These skills will detect and handle issues before breakdowns. A chatbot predicts component breakdown using electric car sensors, telemetry, and maintenance data. Thus, chatbots can suggest preventative maintenance or address minor issues before they become significant. This lowers electric vehicle component failures and prolongs life.
- Conversational chatbots can improve user engagement and accessibility across channels. Chatbots may do this. Speaking chatbots can employ speech recognition in addition to text-based chat interfaces. If Amazon Alexa or Google Assistant had chatbots, drivers could obtain aid without using their hands. Chatbots assist electric car owners obtain advice quickly and easily, regardless of their preferences or situation. Multiple communication channels offer support.
- Conversational chatbots might integrate with electric car infotainment systems. Chatbots on the car's touchscreen can help drivers without their phones. This

connection lets drivers talk with the chatbot while driving, boosting safety and convenience. Infotainment-linked chatbots can give drivers with car-specific and contextual help.

Conversational chatbots for electric car support may enhance breakdown management and user experience. AI-powered electric car breakdown solutions will need chatbots more. They will increase natural language understanding, integrate augmented reality, provide predictive maintenance, enable multi-modal communication, and easily connect with EV infotainment.

3.2.1. Smart Digital Monitoring Calendar and Reminder System.

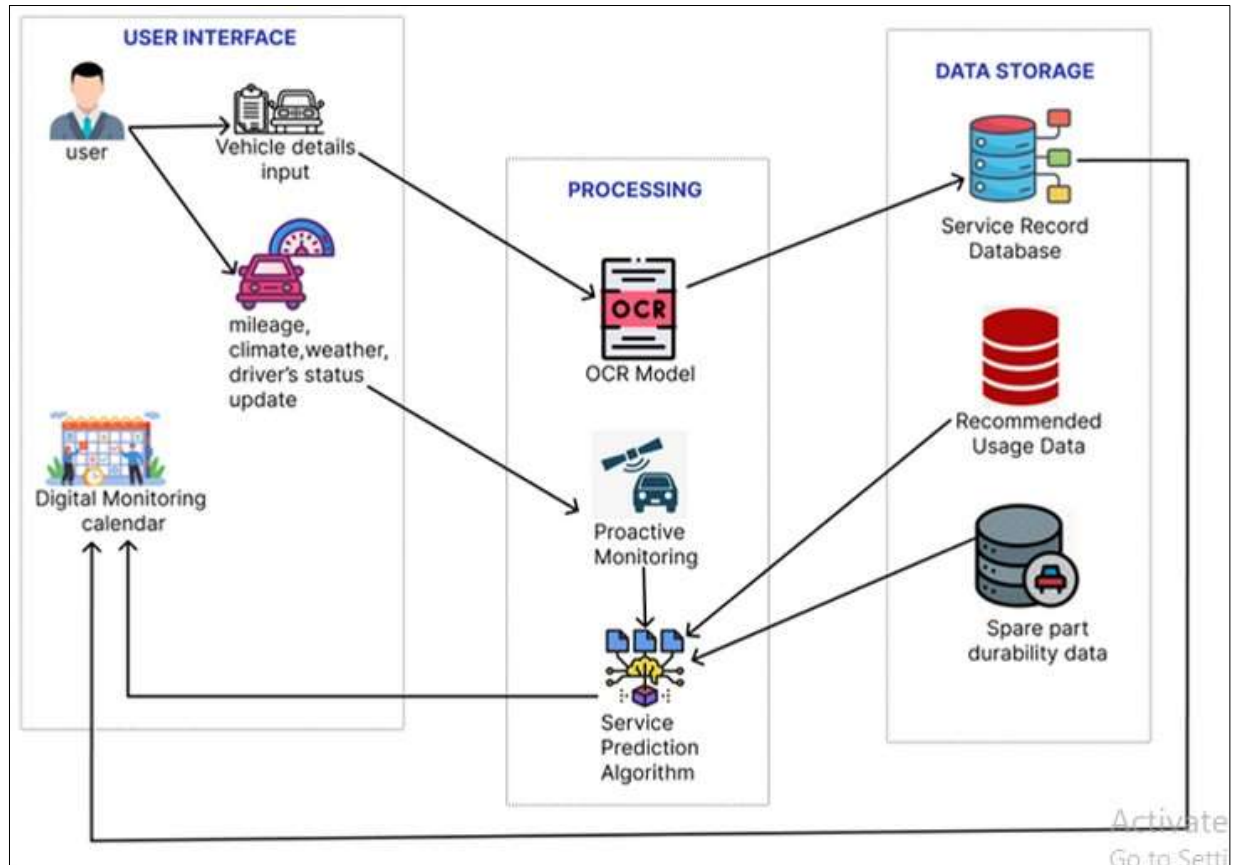


Figure 10 : How Smart Digital Monitoring Calendar and Reminder System works.

Detailed information on a sophisticated vehicle monitoring system is shown in the picture. With this technology, the performance of the vehicle as well as the efficiency of its maintenance will be improved. It is the User Interface that is at the core of it. This interface gives consumers the ability to input important information about their vehicles, such as the mileage, the weather conditions, the amount of time they spend using the climate control, and the driver status. This interface, which serves as the primary point of contact between the user and the system, enables the user to input data and communicate with the system without any interruptions. This interface also offers the user the ability to interact with the system. In order to store and process the vast amount of information that is obtained from the user interface, the system is dependent on a robust Data Storage component. This component is responsible for storing and processing information. Aside from the fact that it

stores user input, this component is also responsible for overseeing the preservation of service records and data concerning advised usage. Because of the efficient arrangement and handling of this data, the system is able to provide accurate insights into the current status of the vehicle's health as well as the maintenance requirements that it needs to fulfil.

The processing of the data that was obtained from the user as well as the automobile is a very significant component to consider when it comes to assessing the data. The term "Optical Character Recognition" (OCR) refers to a technology that interprets data from images, such as invoices or receipts for services rendered. There are a variety of algorithms and models that are included in this component, and one of them is an optical character recognition model. In order to ensure that vehicle maintenance is carried out in the most efficient manner possible, the system is able to derive helpful insights and recommendations by making use of sophisticated data analysis tools.

The capabilities of the system are referred to as "Proactive Monitoring," and they allow the system to predict when it will require maintenance by using the information that it receives and evaluates. Using advanced algorithms, such as a service prediction algorithm that takes into consideration data on the durability of replacement parts, this component is able to anticipate future issues before they become apparent. This is made possible by the fact that it is able to anticipate future difficulties. Users have the ability to avoid unplanned breakdowns and costly repairs with the aid of the system, which proactively detects maintenance requirements. This allows users to avoid costly repairs and breakdowns.

The Digital Monitoring Calendar is an intuitive interface that gives users the ability to observe and visualize upcoming maintenance and service intervals. This enables users to better manage their maintenance and service requirements thanks to the calendar's capabilities. Utilizing this calendar, users are presented with a full image of planned maintenance works, which helps them to effectively plan and manage the maintenance of their cars. This calendar also provides users with the ability to see scheduled repair jobs. Users are given the chance to take responsibility for the health of their vehicle and are encouraged to engage in preventative maintenance practices as a result of the inclusion of

this feature in the system.

Users have the choice to either activate the system or adjust the settings in line with their own preferences when they use the Activate button and the Go to Settings button, respectively. Users are offered with straightforward access to the system's functions through the use of these buttons, which ensures a streamlined and user-focused experience for the user. When these factors are taken into consideration, the integrated components of the vehicle monitoring system work together in harmony to simplify the processes that are involved in vehicle maintenance, improve the performance of the vehicle, and reach greater levels of customer satisfaction.

3.2.2 Model Trained

The term "model" describes the predictive machine learning models that have been created to forecast the lifespan of auto parts. Three regression models in particular are trained and tackle the issues associated with electric automobiles (EVs) that are susceptible to malfunctions. These solutions are specifically engineered to enhance the user experience. XGBoost, CatBoost Regressor, and Random Forest are three prevalent regression algorithms frequently employed in conjunction with this particular case.

XGBoost, short for Extreme Gradient Boosting, is a renowned technique recognized for its exceptional speed and high performance when dealing with regression situations. It is renowned for its exceptional performance. Adopting this strategy is not only efficient but also potent. This approach utilizes an ensemble of weak learners, such as decision trees, to provide a final prediction that aligns with its intended aims. Within the realm of automotive maintenance, XGBoost has the capability to evaluate several factors like the vehicle's mileage, driving habits, weather conditions, and historical maintenance data. This enables it to determine the optimal timing for doing maintenance tasks or replacing components. Due to its ability to handle large datasets and identify complex correlations between variables, it is especially suitable for generating precise suggestions for electric car maintenance. Therefore, it is a suitable instrument for the upkeep of electric cars [6].

When designing an AI solution to address breakdown difficulties with electric vehicles (EVs), using the CatBoost Regressor can greatly improve the system's effectiveness and efficiency, namely in the spare parts shop searching service. The CatBoost Regressor is a machine learning algorithm categorized under the gradient-boosted decision tree models. CatBoost distinguishes itself from other gradient boosting algorithms by its expertise in effectively handling categorical features. This function is especially essential when searching for spare components for electric vehicles in a shop [7].

The Random Forest algorithm is a flexible ensemble learning technique that generates many decision trees throughout the training phase. Subsequently, it produces either the mode of the classes (in the case of classification) or the mean prediction (in the case of

regression) of the individual trees. Random Forest may be applied to several tasks, including classification and regression. Random Forest, in the realm of vehicle maintenance, has the capability to do an analysis on several input elements, including the vehicle's attributes, usage patterns, and environmental conditions. This enables the capability to predict maintenance periods or detect future issues. When it comes to vehicle maintenance solutions, regression jobs often choose this particular pick due to its resistance to overfitting and its ability to handle both categorical and numerical data. Moreover, it has the capability to process both sorts of data. Furthermore, it has the ability to handle both types of data.

Some regression algorithms commonly used in AI-powered solutions for electric car maintenance are XGBoost, CatBoost, and Random Forest. These algorithms are essential elements of the solutions. In conclusion, these algorithms serve a crucial purpose. Through the examination of many characteristics and past information, these algorithms may produce accurate forecasts and suggestions for maintenance activities. Consequently, they ultimately improve the user experience and reduce the likelihood of failure situations occurring [9].

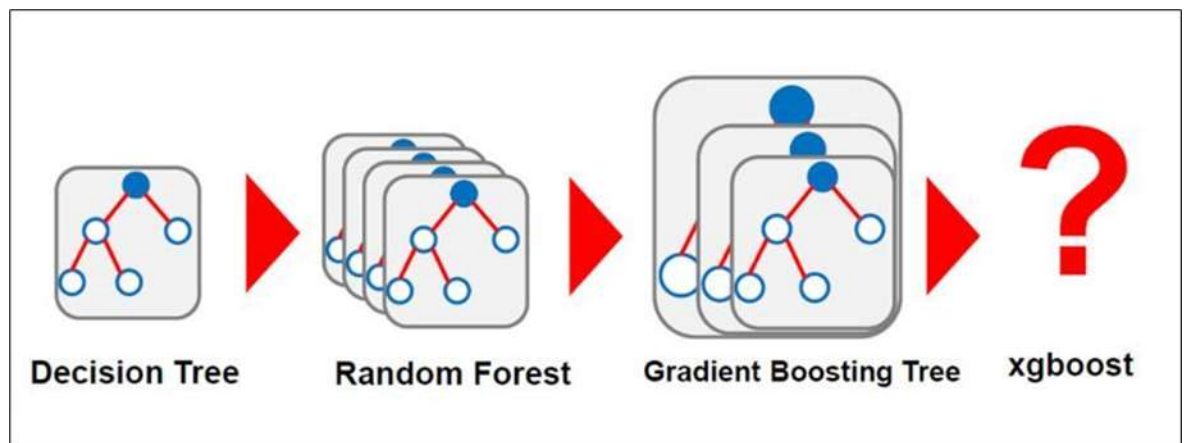


Figure 11 : XGBoost

Gensim is a Python package that is open-source and specifically created for performing topic modelling and natural language processing (NLP) operations. The software provides a range of tools and techniques for constructing, training, and utilizing topic models,

including Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA). Gensim offers implementations of word embedding algorithms such as Word2Vec, Doc2Vec, and FastText. These algorithms are widely used for expressing words and documents as dense vector representations in continuous vector spaces. Gensim enables many tasks such as calculating semantic similarity, grouping documents, summarizing material, and classifying documents. It is frequently employed in applications that deal with extensive collections of text, such as information retrieval, content recommendation, sentiment analysis, and document comprehension.

This method entails providing the training data to the model and modifying the model's parameters according to the error feedback. Through the process of iteratively going through numerous training cycles and fine-tuning the hyperparameters, the model steadily enhances its accuracy and performance. During the process of hyperparameter tweaking and retraining, the XGBoost model generally experiences an increase in accuracy, resulting in improved predictions of maintenance requirements and identification of probable breakdown hazards. By improving its accuracy, the model can offer more dependable suggestions to vehicle owners, enabling them to proactively tackle maintenance concerns and reduce the likelihood of failures. By fine-tuning the hyperparameters of the XGBoost algorithm, the vehicle maintenance solution's overall efficacy and customer happiness are improved.

Integrating the CatBoost Regressor into the EV spare parts shop searching service greatly improves the system's performance. The specialized capabilities of this system in managing categorical data, such as electric vehicle make and model, precise part names, and shop locations, removes the requirement for manual preparation. This optimizes the recommendation process, enhancing its efficiency and precision.

Through the utilization of CatBoost Regressor, the AI-driven system can rapidly detect appropriate retailers of spare parts by analyzing user inquiries, resulting in quicker response times and enhanced scalability. This not only improves the user experience by offering prompt support but also guarantees the dependability of the service, hence

contributing to the smooth functioning of EV maintenance and repair procedures.

Pytesseract is a Python package that functions as an interface for Google's Tesseract-OCR Engine. OCR, short for Optical Character Recognition, is a technology that allows machines to identify and extract text from photographs or scanned documents. Pytesseract enables seamless integration of optical character recognition (OCR) functionalities into Python programmes. The Tesseract-OCR Engine may be easily accessed through a user-friendly interface, allowing for various operations like image preprocessing, text recognition, and result extraction. Pytesseract is extensively utilized in diverse fields such as document digitalization, image-based text extraction, automated data entry, and other applications .

Natural Language Processing (NLP) is a field within artificial intelligence (AI) and machine learning (ML) that focuses on enabling computers to understand, interpret, and produce human language with proficiency. The core of natural language processing involves empowering robots to comprehend the intricacies of human communication, encompassing grammar, semantics, and pragmatics. Natural Language Processing (NLP) algorithms address a wide range of tasks, including sentence parsing, part-of-speech identification, and extracting meaning and sentiments from text. This process entails employing methodologies like tokenization, which involves breaking down text into meaningful parts, and named entity recognition, which recognizes certain entities such as names or locations. NLP also includes applications such as sentiment analysis, which involves analyzing the emotional tone of text, and text creation, which involves constructing responses or material that resemble human-like language based on input data.

3.2.3 Technologies

To solve the issue of breakdowns that are associated with electric vehicles, artificial intelligence-powered solutions are currently being researched. The primary purpose is to enhance the user experience in matters pertaining to vehicle maintenance through the implementation of innovative technical methods. A method that takes a holistic perspective involves using a variety of technologies in order to create a system that is coherent and efficient. React Native, Python, Flask, Firebase, Node Server, Python Image OCR, and Machine Learning (ML) are some of the essential technologies that are needed in order to accomplish this objective.

- React Native

The framework known as React Native is commonly utilized for the purpose of developing mobile apps that are capable of operating on several platforms. It is possible for developers to utilize JavaScript and React to create mobile applications that provide a user experience that is similar to that of a native application on both iOS and Android devices. consumers are able to readily access the capabilities of the automobile maintenance system from their mobile devices, such as smartphones and tablets, thanks to the fact that React Native makes it possible to create a mobile interface that is simple to browse and intuitive for consumers.

- NLP

NLP (Natural Language Processing) Techniques: Although not explicitly implemented in this code snippet, NLP techniques could be applied for further processing of the extracted text data, such as tokenization, removing stop words, stemming, or lemmatization.

- Flask

Developed in Python, Flask is a web framework that is both lightweight and ideally suited for the development of APIs and online applications. Flask is used in the automobile maintenance system to provide RESTful application programming interfaces (APIs), which make it easier for the front-end interface and the backend services to communicate with one another. Additionally, it provides an easy yet robust approach for making functionality accessible to the frontend application that was constructed using React Native. Some

examples of this functionality include user identification, data retrieval, and maintenance scheduling.

- Firebase

The development of mobile and internet applications is made possible by Google's Firebase, which is a powerful platform that the company provides. The platform offers a variety of critical services, including real-time database, authentication, cloud storage, and hosting, which are required for the development of modern applications that are dependent on cloud technology. Firebase may be utilized inside the automobile maintenance system for the purpose of storing user data, managing authentication, and facilitating real-time data syncing between the server and the different clients.

- Node server

There is a runtime environment known as Node.js that enables the execution of JavaScript code on the server side. This environment is known as Node Server. When used in conjunction with Flask, Node.js may be employed to carry out specialist tasks, such as managing the uploading of files, carrying out lengthy operations, or interfacing with other services. Developers have the opportunity to design a backend architecture that is both durable and extendable by utilizing Node.js. This architecture boosts the features that Flask and Firebase have to offer.

- Data Filtering:

The code filters out null items and unnecessary spaces from the processed text data to ensure the quality and accuracy of the extracted information.

- Verify API:

The code interacts with the Verify API to submit bill images for processing and retrieve structured data such as line items, total amount, and date from the bills.

- OCR

Python Image OCR is a piece of software that extracts text from photographs by doing so

through the application of Optical Character Recognition (OCR) technology. The programming language Python offers a wide variety of libraries and tools that may be utilized to create Optical Character Recognition (OCR) capabilities. The automobile maintenance system is able to examine photographs of maintenance bills, invoices, or papers pertaining to the vehicle as a result of this. Through the use of optical character recognition (OCR) capabilities, the system is able to automatically extract relevant information from images that are uploaded by users. This results in an improvement in both the efficiency and precision of data entry and record-keeping.

3.2.4 Commercialization

AI-powered solutions for electric car breakdowns have advanced the auto industry. Businesses want to change how electric car owners interact with maintenance by using cutting-edge technologies like a smart digital monitoring calendar and reminder system. This improves car maintenance user experience. This comprehensive solution uses cutting-edge artificial intelligence algorithms, user-friendly interfaces, and predictive analytics to monitor automotive state, optimize maintenance plans, and prevent unexpected issues. Within this framework, examining the commercialization aspects of such a product comprises market analysis, price strategy development, distribution channel construction, and marketing campaign execution. To properly launch this innovative product, rules must be followed, and customer service prioritized.

- **Market Analysis** In order to understand the demographic groups being targeted, the competitors, and the sector trends, significant market research is necessary. Businesses can find market niches and unique ways to differentiate themselves by researching electric car demand and maintenance solutions.
- **The AI-driven solution must prove its game-changing impact on product placement in the car repair industry.** Its capacity to improve user experience, speed up maintenance, and prevent issues sets it apart from other methods. Highlighting features like the clever digital monitoring calendar and reminder system may attract tech-savvy customers.
- **Product pricing technique** the pricing plan must evaluate the product's features, the target market's demographics, and the competition's backdrop. If you offer multiple pricing options, such as subscription models with different levels of service, you may suit your customers' needs while staying inside their budget. Maintaining product competitiveness is another option.
- **Distribution Channels:** Selecting the right channels is crucial for reaching the right

audience. Partnerships with electric car manufacturers, dealerships, and service centers can help integrate the AI-powered solution into existing product lines. Online platforms, mobile app stores, and e-commerce websites can help with direct client communication.

- **Promotion and Marketing:** A thorough marketing plan is necessary to boost product visibility and interest. Digital marketing platforms including email, social media, and SEO provide more effective client communication. To target specific demographics, emphasize the smart digital monitoring calendar and reminder system's benefits, such as longer vehicle lifespan and maintenance efficiency.
- To maintain client satisfaction and loyalty, prioritize great customer care and actively seek feedback. Companies can simplify customer use by providing user manuals, tutorials, and technical support. Establishing avenues for consumer feedback and actively engaging with customers is vital to gather input for product adjustments.
- Adhering to applicable rules and regulations is crucial for building client trust and confidence. This is our seventh and final topic. Compliance with data privacy, cybersecurity, and industry standards is essential to protect consumer data and meet regulatory requirements. Maintaining consumer data confidentially is crucial. Staying abreast of regulatory changes and making necessary product revisions is crucial to long-term market success.

3.2.5 Testing and Implementation

The process of testing and implementing AI-powered solutions for electric vehicle breakdown challenges, with the primary objective of enhancing the user experience in vehicle maintenance through the use of a smart digital monitoring calendar and reminder system, requires several crucial steps to ensure the system's effectiveness and reliability.

1. During the initial phase, referred to as "Functional Testing," every element of the AI-driven system is thoroughly tested to ensure that it operates according to its intended design. In order to guarantee that users can comfortably schedule maintenance tasks and receive timely reminders, it is necessary to evaluate the effectiveness of the digital monitoring calendar and reminder system.
2. Integration Testing: Integration testing is conducted to verify seamless communication between different system components. To ensure that maintenance ideas are accurately reflected in the calendar, it is important to perform tests on the integration of predictive maintenance algorithms with the digital monitoring calendar system.
3. User acceptability testing, also referred to as UAT, is a procedure where real users assess an application by evaluating its alignment with their real-world requirements. Users interact with the system to assess its usability, effectiveness, and overall user experience. The input obtained from user acceptability testing is crucial for enhancing the system to meet the users' expectations more effectively.
4. Performance Testing: Performance testing is a technique used to assess the system's capacity to handle different loads and scenarios by measuring its responsiveness and stability. During the testing phase, the system must be scrutinized to ensure that it maintains its responsiveness even under heavy usage and can handle a substantial number of users concurrently without experiencing crashes.
5. Security testing is crucial for identifying and addressing vulnerabilities that might compromise the integrity and confidentiality of user personal information. When doing

this type of testing, we carefully analyze potential risks such as unauthorized access, data breaches, and malicious attacks.

6. **Algorithm Validation:** To ensure the accuracy and reliability of the predictive maintenance algorithms used in the system, they must undergo a validation procedure. This involves comparing the forecasts generated by the algorithm with the data gathered from real-world observations to assess the effectiveness of the system in detecting future failures and suggesting maintenance tasks.
7. **(Evaluation of User Experience) Usability testing** is to evaluate the user-friendliness of the system interface and interactions. Users must perform tasks within the system to assess its user-friendliness and intuitiveness. The objective of usability testing is to identify areas where the user experience may be improved more efficiently.
8. **Documentation and Training:** Users and administrators must have access to thorough documentation that covers all the features, functions, and instructions for using the system. Users can be provided with training resources to ensure they can effectively utilize the system and maximize its capabilities.

Efficiently evaluating and deploying AI-powered solutions for automobile maintenance may be achieved by systematically conducting testing and implementation procedures. This will finally provide consumers with a reliable and user-friendly encounter while simultaneously resolving the problems linked to electric car malfunctions.

Electric vehicles, often known as EVs, are ushering in a new era in the transportation industry by offering alternatives to traditional automobiles powered by internal combustion engines that are better for the environment. Electric car drivers continue to be fairly concerned about issues such as breakdowns and maintenance, despite the fact that this is the case. Solutions that are driven by artificial intelligence offer a feasible avenue for predictive maintenance, which, when paired with a digital monitoring calendar and reminder system that is user-friendly, can assist in resolving this challenge.

When evaluating artificial intelligence for predictive maintenance, it is vital to make use of the large amount of information that is obtained from on-board electric vehicle sensors. This information includes things like the health of the battery, the performance of the motor, and thermal management. These data, coupled with previous service records and insights on the longevity of replacement components, serve as the foundation for the training of machine learning models. Additionally, past service records are also taken into consideration. When these algorithms are being tested, they evaluate patterns within simulated driving scenarios in order to make predictions about the possibility of potential failures occurring while they are being tested. In order to increase the accuracy of the model's ability to forecast the requirements for maintenance, it is very required to adjust the model based on the results of the aforementioned tests.

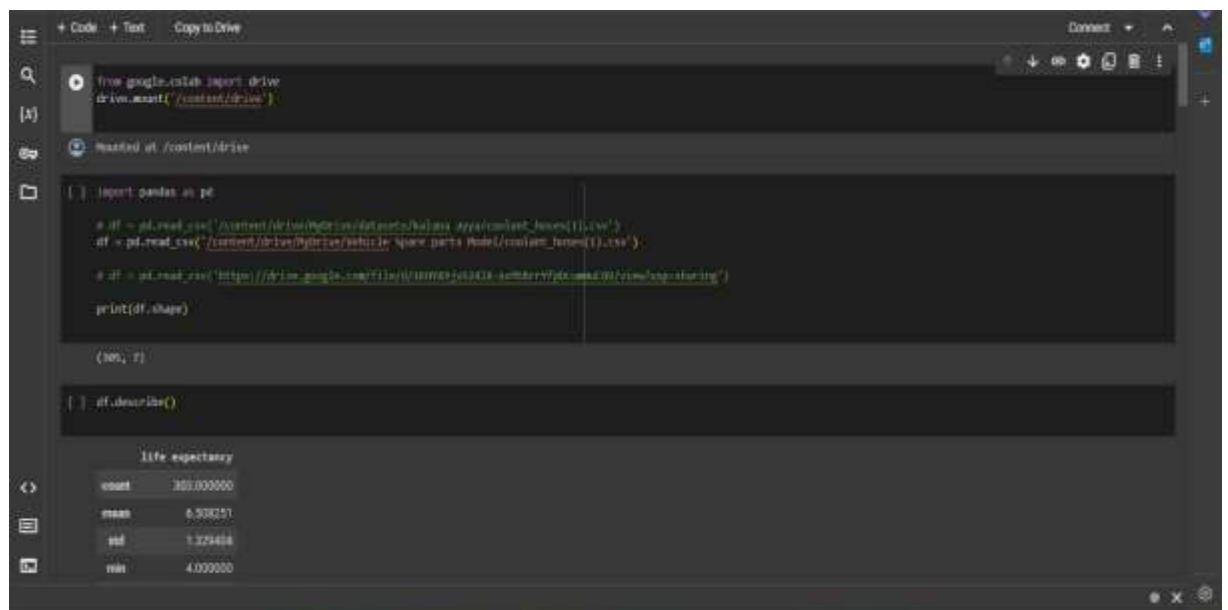
For the vehicle monitoring system to be deployed for the purpose of preventing breakdowns, the trained artificial intelligence model has to be included into the processing unit of the system. Whenever the model identifies an impending issue, it sends the driver proactive notifications, and it makes recommendations to the driver. These alerts and recommendations might be anything from alerting roadside assistance to scheduling appointments for maintenance. At the same time that this is taking place, the digital monitoring calendar is being updated with the anticipated maintenance intervals. It is because of this that users are provided with a thorough understanding of the forthcoming service needs.

The enhancement of the user experience is absolutely required in order to nurture trust and engagement with the system that is generated by artificial intelligence. From the digital monitoring calendar, the intelligent reminder system will send tailored notifications to the user's smartphone app in order to remind them of upcoming maintenance. These alerts will be sent in order to remind the user of the upcoming maintenance. Users are also provided with educational information on the maintenance of electric vehicles through the application. This provides them with the knowledge that is essential to grasp the reasoning that lies behind the forecasts that are generated by artificial

intelligence.

Customers are able to have more trust in the system as a result of this transparency, and they are also able to take preventative steps in the maintenance of their electric cars as a result of this openness.

A revolutionary shift toward preventative maintenance is brought about in the ownership of electric vehicles by combining an intuitive digital monitoring and reminder system with a breakdown prediction system driven by artificial intelligence. This combination brings about a revolutionary transformation. The implementation of this all-encompassing strategy not only enhances the level of pleasure that users report experiencing, but it also helps to prevent breakdowns, ensuring that electric vehicles function in a smooth and efficient manner, and consequently driving the adoption of environmentally friendly transportation alternatives.



```
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd

# df = pd.read_csv('/content/drive/MyDrive/dataset/huawei_aps/coolant_hoses(1).csv')
df = pd.read_csv('/content/drive/MyDrive/vehicle Space parts Model/coolant_hoses(1).csv')

# df = pd.read_csv('https://drive.google.com/file/d/1HfW8t63418-4cm8rrVpX-maE30/view?usp=sharing')

print(df.shape)

(305, 7)

[ ] df.describe()
```

life expectancy	
count	305.000000
mean	6.308251
std	1.329404
min	4.000000

A Python script running in Google Colab, an online machine learning platform. The script seems to be operating on a dataset pertaining to car coolant hoses. This is how the code is broken down:

Drive and Pandas are the first two libraries the script imports.

drive, which enables the script to access data stored there, is probably used to mount

Google Drive to Colab. Pandas is a well-liked data analysis and manipulation package. The script looks in two places for a CSV file called "coolant hoses(1).csv" to read. Next, it attempts to read it from a different location at "/content/drive/MyDrive/Vehicle Spare parts Model/" on your Drive. The file is commented out and the script doesn't try to read from a URL if neither location has it. Print Dataframe Shape: Presuming pd.read, one of the earlier locations, holds the file. The CSV function loads data into a Pandas dataframe from a CSV file. df.shape is then used by the script.

```
df.describe()
```

life expectancy	
count	100.000000
mean	6.582291
std	1.329404
min	6.000000
25%	5.500000
50%	7.000000
75%	8.000000
max	9.000000


```
df.head(10)
```

	Component	Condition	Usage Intensity	Manufacturer	Weather Conditions	Road Conditions	Life expectancy
0	Coolant Hoses	Regular	High	OEM	Hot/Dry	Rough	8.0
1	Coolant Hoses	Regular	Moderate	Aftermarket	Cold/Wet	Smooth	7.0
2	Coolant Hoses	Irregular	Low	OEM	Cold/Wet	Potholed	5.0
3	Coolant Hoses	High	High	OEM	Hot/Dry	Rough	8.0
4	Coolant Hoses	Regular	High	Aftermarket	Hot/Dry	Smooth	8.0

```
df.head(10)
```

	Component	Condition	Usage Intensity	Manufacturer	Weather Conditions	Road Conditions	Life expectancy
0	Coolant Hoses	Regular	High	OEM	Hot/Dry	Rough	8.0
1	Coolant Hoses	Regular	Moderate	Aftermarket	Cold/Wet	Smooth	7.0
2	Coolant Hoses	Irregular	Low	OEM	Cold/Wet	Potholed	5.0
3	Coolant Hoses	High	High	OEM	Hot/Dry	Rough	8.0
4	Coolant Hoses	Regular	High	Aftermarket	Hot/Dry	Smooth	8.0
5	Coolant Hoses	Regular	Low	OEM	Cold/Wet	Potholed	5.0
6	Coolant Hoses	Irregular	High	OEM	Moderate Hot/Humid	Smooth	8.0
7	Coolant Hoses	Regular	High	Aftermarket	Cold/Wet	Rough	NaN
8	Coolant Hoses	Regular	Moderate	OEM	Hot/Dry	Smooth	7.0
9	Coolant Hoses	Irregular	Low	Aftermarket	Moderate Hot/Humid	Potholed	5.0


```
df_filled = df.apply(lambda x: x.fillna(x.value_counts().index[0]))
print(df_filled)
```

	Component	Condition	Usage Intensity	Manufacturer
0	Coolant Hoses	Regular	High	OEM
1	Coolant Hoses	Regular	Moderate	Aftermarket
2	Coolant Hoses	Irregular	Low	OEM
3	Coolant Hoses	High	High	OEM
4	Coolant Hoses	Regular	High	Aftermarket
5	Coolant Hoses	Regular	Low	OEM
6	Coolant Hoses	Irregular	High	OEM
7	Coolant Hoses	Regular	High	Aftermarket
8	Coolant Hoses	Regular	Moderate	OEM
9	Coolant Hoses	Irregular	Low	Aftermarket

```

+ Code + Test Copy to Drive
df_filled = df.apply(lambda x: x.fillna(x.value_counts().index[0]))
print(df_filled)

Component Condition Usage Intensity Manufacturer \
0 Contact Moon Regular High OEM
1 Contact Moon Regular Moderate Aftermarket
2 Contact Moon Irregular Low OEM
3 Contact Moon High High OEM
4 Contact Moon Regular High Aftermarket
...
300 Contact Moon Irregular Low OEM
301 Contact Moon High High OEM
302 Contact Moon Regular Moderate Aftermarket
303 Contact Moon Irregular Low OEM
304 Contact Moon Regular High Aftermarket

Weather Conditions Road Conditions Life expectancy
0 Wet/Dry Rough 6.0
1 Cold/Wet Smooth 7.0
2 Cold/Wet Moderate 5.0
3 Wet/Dry Rough 6.0
4 Wet/Dry Smooth 6.0
...
300 Cold/Wet Smooth 5.0
301 Wet/Dry Rough 6.0
302 Cold/Wet Smooth 7.0
303 Moderate Wet/Wet Rough 4.0
304 Wet/Dry Smooth 6.0

[305 rows x 7 columns]

1 | + Separate target column
target_column = df_filled["life expectancy"]

```

The fillna() method is applied to a Pandas dataframe called df by the code.

A dataframe's missing values, such as NaN, can be replaced with a specified value using the fillna() method. In this instance, the script uses.value_counts().index[0] to replace in the missing values in df with the value that occurs the most frequently.

The function.value_counts() determine how many times a given unique value occurs in a column. The index (column value) that appears the most frequently is chosen by using.index[0].

```

+ Code + Test Copy to Drive
# separate target column
target_column = df_filled["life expectancy"]

# separate remaining columns
remaining_columns = df_filled.drop(["life expectancy", "Component"], axis=1)

# print the separated data
print("target column:")
print(target_column)

print("remaining columns:")
print(remaining_columns)

target column:
0    6.0
1    7.0
2    5.0
3    6.0
4    6.0
...
300   5.0
301   6.0
302   7.0
303   4.0
304   6.0
Name: life expectancy, length: 305, dtype: float64

remaining columns:
Condition Usage Intensity Manufacturer Weather Conditions \
0 Regular High OEM Wet/Dry
1 Regular Moderate Aftermarket Cold/Wet
2 Irregular Low OEM Cold/Wet

```

df["life expectancy"] (target_column) is used to split the target column. With this line, a new variable called target_column is made. It sets the new variable as the number in the

"life expectancy" column of the dataframe df.

This saves the "life expectancy" column in a different variable and keeps it separate. Taking Out the Last Few Columns (df.drop(["life expectancy"], axis=1, remaining_columns)) This line sets up a new variable called remaining_columns.

With axis=1, we're dropping columns. With axis=0, we'd be dropping rows.

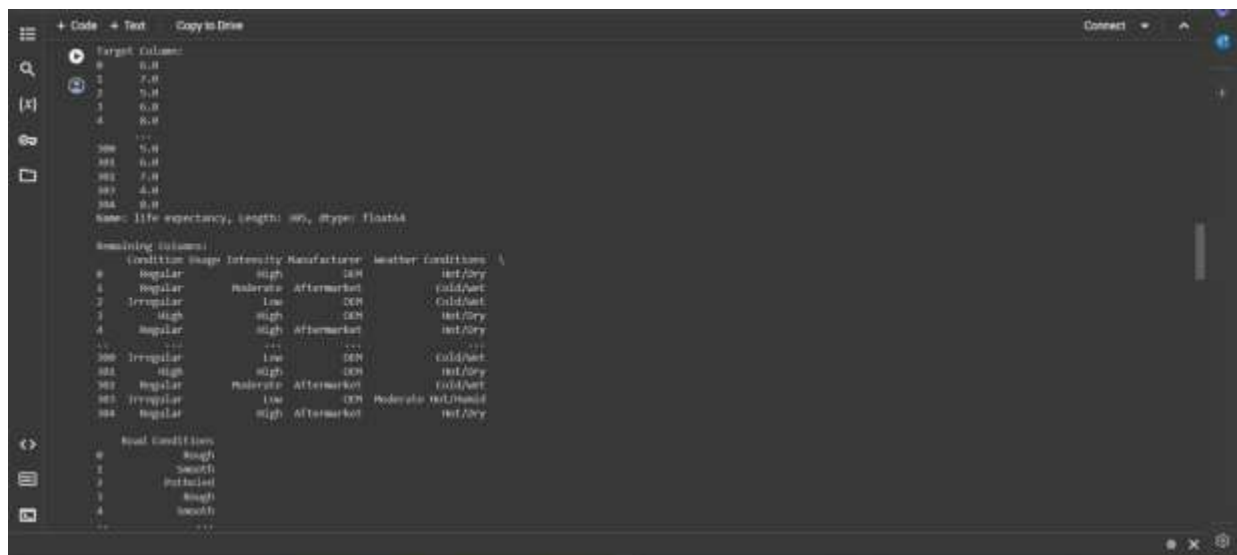
If you do this, you will get a new dataframe with all the fields from the first dataframe df except for the "life expectancy" column. This dataframe is stored in the remaining_columns value.

In the last four lines of code, the values of the two variables we made are printed:

When you run the first print line, it shows the "Target column:" label and the "life expectancy" data in the target_column variable.

All the columns except "life expectancy" are in the remaining_columns variable, which is printed after the label "Remaining Columns:" in the second print line.

This code basically splits a dataframe df in half: the goal column "life expectancy" and the rest of the columns in the dataframe.



```
Target Column:
0    6.0
1    7.0
2    5.0
3    6.0
4    8.0
...
100  5.0
101  6.0
102  7.0
103  4.0
104  8.0
Name: life expectancy, length: 105, dtype: float64

Remaining Columns:
   condition  usage  intensity  manufacturer  weather  condition2
0    Regular    High        High          OEM      hot/dry
1    Regular  Moderate  Aftermarket    ColdFueled  Cold/dry
2  Irregular    Low         Low           OEM      Cold/dry
3    Regular    High        High          OEM      hot/dry
4    Regular    High  Aftermarket    hot/dry
...
100  Irregular    Low         Low           OEM    Cold/dry
101    High        High        High           OEM    hot/dry
102    Regular  Moderate  Aftermarket    ColdFueled  Cold/dry
103  Irregular    Low         Low           OEM    Moderate  hot/dry
104    Regular    High  Aftermarket    hot/dry

Road Conditions
0    Rough
1    Smooth
2    Potholed
3    Rough
4    Smooth
...
```

```

# Read Conditions
0      Rough
1      Smooth
2      Polished
3      Rough
4      Smooth
...
500    Smooth
501    Rough
502    Smooth
503    Rough
504    Smooth

[505 rows x 5 columns]

# Get unique values for each column
unique_values = {}
for column in remaining_columns.columns:
    unique_values[column] = remaining_columns[column].unique()

# Print the unique values for each column
for column, values in unique_values.items():
    print(f"Unique values for column '{column}': {values}")

Unique values for column 'Component': ['Coolest room']
Unique values for column 'Condition': ['Regular' 'Irregular' 'High']
Unique values for column 'Usage Intensity': ['High' 'Moderate' 'Low']
Unique values for column 'Manufacturer': ['IBM' 'Aftermarket']
Unique values for column 'Weather Conditions': ['Hot/Dry' 'Cold/Wet' 'Moderate hot/humid']
Unique values for column 'Road Conditions': ['Rough' 'Smooth' 'Polished']

from sklearn.preprocessing import OneHotEncoder

# Get unique values for each column
unique_values = {}
for column in remaining_columns.columns:
    unique_values[column] = remaining_columns[column].unique()

# Print the unique values for each column
for column, values in unique_values.items():
    print(f"Unique values for column '{column}': {values}")

Unique values for column 'Component': ['Coolest room']
Unique values for column 'Condition': ['Regular' 'Irregular' 'High']
Unique values for column 'Usage Intensity': ['High' 'Moderate' 'Low']
Unique values for column 'Manufacturer': ['IBM' 'Aftermarket']
Unique values for column 'Weather Conditions': ['Hot/Dry' 'Cold/Wet' 'Moderate hot/humid']
Unique values for column 'Road Conditions': ['Rough' 'Smooth' 'Polished']

from sklearn.preprocessing import OneHotEncoder

# One-hot encode categorical columns
encoder = OneHotEncoder(handle_unknown="ignore")
encoded_data = encoder.fit_transform(remaining_columns[remaining_columns.columns]).toarray()

# Convert encoded data to a DataFrame
encoded_df = pd.DataFrame(encoded_data)

# Print the encoded DataFrame
print(encoded_df.head(10))
print(encoded_df)

0      2703.000  0.000000  0.000000  0.000000  0.000000
1      3333.000  0.000000  0.000000  0.000000  0.000000

```

OneHotEncoder:

This code uses scikit-learn's OneHotEncoder to generate a one-hot encoder. One-hot encoding converts text labels into numerical data for machine learning algorithms. The encoder ignores unknown categories instead of producing an error using the `handle_unknown="ignore"` option.

Data Encoding (`encoder.fit_transform`):

Line `encoded_data = encoder.fit_transform(remaining_columns[remaining_columns.columns]).toarray()` is performed by `fit_transform(remaining_columns)`. `Fit_transform` combines two stages. `fit()` finds unique categories in `remaining_columns`.

convert() encodes remaining_columns with learnt categories. Output is in encoded_data. Pd.DataFrame encoded: Line encoded_df = pd.DataFrame(encoded_data) is created by DataFrame(encoded_data). The one-hot encoded data from encoded_data is added to encoded_df.

The image shows two screenshots of a Jupyter Notebook interface. The top screenshot displays code for encoding categorical data using OneHotEncoder. The code includes imports, a function to handle categorical columns, fitting the encoder, transforming the data, and printing the resulting DataFrame. The output shows a DataFrame with numerical values for the first few rows. The bottom screenshot shows code for splitting the data into training and testing sets using train_test_split. It includes imports, the split function call, and printing the shapes of the resulting arrays. The output shows the shapes for X_train, X_test, y_train, and y_test.

```

from sklearn.preprocessing import OneHotEncoder

# One-hot encode categorical columns
encoder = OneHotEncoder(handle_unknown='ignore')
encoded_data = encoder.fit_transform(remaining_columns[remaining_columns.columns])

# Convert encoded data to a dataframe
encoded_df = pd.DataFrame(encoded_data)

# Print the encoded dataframe
print(encoded_df.head(10))
print(encoded_df)

0  (0, 23\11.000) (0, 33\11.000) (0, 73\11.000) ...
1  (0, 23\11.000) (0, 33\11.000) (0, 43\11.000) ...
2  (0, 33\11.000) (0, 23\11.000) (0, 73\11.000) ...
3  (0, 43\11.000) (0, 23\11.000) (0, 73\11.000) ...
4  (0, 23\11.000) (0, 33\11.000) (0, 43\11.000) ...
5  (0, 23\11.000) (0, 43\11.000) (0, 73\11.000) ...
6  (0, 13\11.000) (0, 33\11.000) (0, 73\11.000) ...
7  (0, 23\11.000) (0, 43\11.000) (0, 43\11.000) ...
8  (0, 23\11.000) (0, 33\11.000) (0, 73\11.000) ...
9  (0, 13\11.000) (0, 43\11.000) (0, 43\11.000) ...

...

996  (0, 23\11.000) (0, 43\11.000) (0, 73\11.000) ...
997  (0, 23\11.000) (0, 33\11.000) (0, 43\11.000) ...
998  (0, 23\11.000) (0, 23\11.000) (0, 73\11.000) ...
999  (0, 23\11.000) (0, 33\11.000) (0, 43\11.000) ...

[100 rows x 3 columns]

from sklearn.model_selection import train_test_split

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(remaining_columns, target_column, test_size=0.2, random_state=42)

# Print shapes of train and test sets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)

X_train shape: (344, 6)
X_test shape: (66, 6)
y_train shape: (344,)
y_test shape: (66,)

```



```

+ Code + Test Copy to Drive
from sklearn.model_selection import train_test_split

# split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(remaining_columns, target_column, test_size=0.2, random_state=42)

# Print shapes of train and test sets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)

X_train shape: (200, 6)
X_test shape: (50, 6)
y_train shape: (200,)
y_test shape: (50,)

# Import necessary libraries
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# Train Random Forest Regressor
random_forest_model = RandomForestRegressor(n_estimators=100, random_state=42)
random_forest_model.fit(X_train, y_train)

# Train XGBoost Regressor
xgboost_model = XGBRegressor(n_estimators=100, random_state=42)
xgboost_model.fit(X_train, y_train)

# Train Polynomial Regressor (degree 2)

```

Pd is the first line imported pandas library. Pandas is a popular Python data processor.

Read CSV (pd.read_ CSV):

Print df.shape:

The following two lines display the shapes of the two DataFrames, with row and column counts from the.shape property. The first print command displays DataFrame df dimensions. Second print command displays df_filtered DataFrame dimensions.

This method filters YouTube trending video data from a CSV file to keep videos with over 100,000 views and generates the original and filtered DataFrame.

```

+ Code + Test Copy to Drive
# Import necessary libraries
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# Train Random Forest Regressor
random_forest_model = RandomForestRegressor(n_estimators=100, random_state=42)
random_forest_model.fit(X_train, y_train)

# Train XGBoost Regressor
xgboost_model = XGBRegressor(n_estimators=100, random_state=42)
xgboost_model.fit(X_train, y_train)

# Train Polynomial Regressor (degree 2)
poly_features = PolynomialFeatures(degree=2)
X_train_poly = poly_features.fit_transform(X_train)
X_test_poly = poly_features.transform(X_test)
linear_model = LinearRegression()
linear_model.fit(X_train_poly, y_train)

# Print model names and training completion messages
print("Random Forest Regressor trained successfully.")
print("XGBoost Regressor trained successfully.")
print("Polynomial Regressor trained successfully.")

```

Figure 12 : Code

Tesseract is a robust open-source Optical Character Recognition (OCR) engine that is capable of extracting text from images. It is widely favored for several OCR assignments because of its:

- Tesseract's open-source nature enables universal accessibility and customization due to its free and open availability.
- Tesseract has commendable accuracy, particularly when dealing with unambiguous and properly structured text.
- Language versatility: This software has extensive support for multiple languages, allowing it to be used in a variety of situations.
- Tesseract is an optical character recognition (OCR) system.

Preprocessing of Images:

Prior to extracting text, Tesseract frequently necessitates image pre-processing. This may entail converting the image to grayscale, implementing noise reduction techniques, or modifying the contrast.

Optical Character Recognition (OCR):

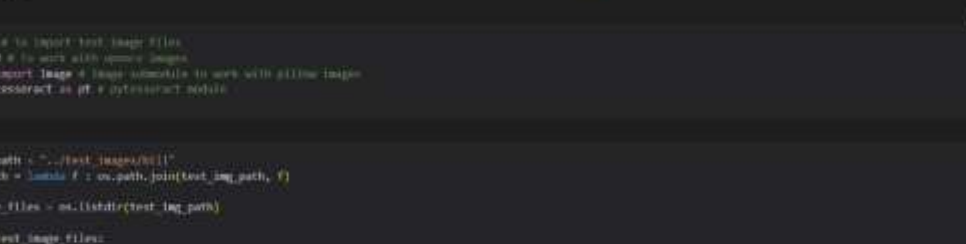
Tesseract uses pattern recognition methodologies to accurately detect and distinguish individual characters within the given image. The system utilizes a process of comparing small image portions, often known as sub-images, to its internal character database in order to identify the most accurate match.

Text Assembly:

Tesseract performs an analysis of the placements of individual characters and subsequently combines them to form words and lines of text.

You can add the "pytesseract" library and give it the alias "pt" by using the "import" term. There is the pytesseract package in this line. It is a wrapper for the Tesseract OCR engine. The optical character recognition (OCR) tool Tesseract can be downloaded for free and is used to extract text from photos [23].

The second item in the list "test_image_files" is given to the variable "image_path" as its value. The path to the second picture in the list of test_image_files is given to the image_path variable by this line.



The screenshot shows a Jupyter Notebook with the following code:

```

import cv2
import os
from PIL import Image
import pytorchdract as pt

test_img_path = ".../test_images/0011"
create_path = lambda f : os.path.join(test_img_path, f)

test_image_files = os.listdir(test_img_path)

for f in test_image_files:
    print(f)

def show_image(img_path, size=(600, 600)):
    image = cv2.imread(img_path)
    image = cv2.resize(image, size)

    cv2.imshow("image", image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

image_path = test_image_files[1]
path = create_path(image_path)

image = image.open(path)
text = pt.image_to_string(image, config="--num 3 --pre 67")

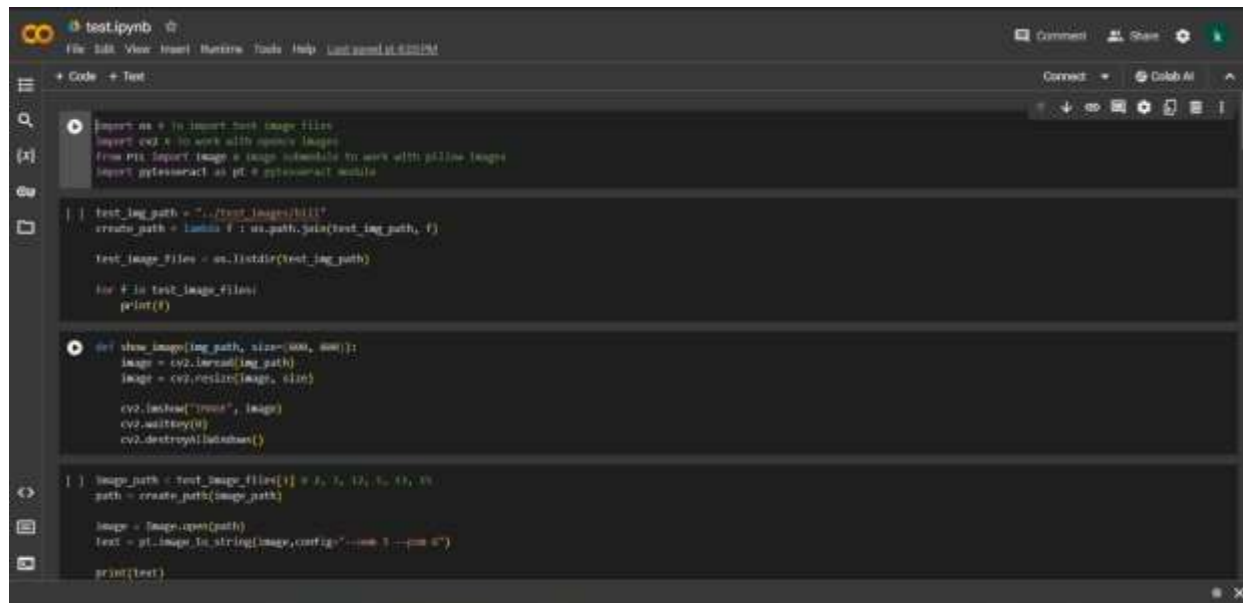
```

The interface includes a top bar with file management icons, a left sidebar with a file explorer, and a right sidebar with a 'Select Kernel' button. The main area displays the code cells, and the bottom status bar shows the current line and column.

```
verify.py -t  
File Edit View Insert Runtime Tools Help Last saved at 4:50 PM  
  
+ Code + Test  
  
[ ] # remove flattening and columns and rows  
import sys as s to import host image files  
import cv2 as c to work with opencv images  
from PIL import Image as Image submodule to work with pillow images  
  
# from verify-import Client  
  
# get your keys here: https://hub.verify.cc/magic/  
client_id = "wHsbGmSdCHTSMdtThQwmsCvrrYkzshuf"  
client_secret = "KqPwhlRvdorDtpoUwdsYhsatfStbaSAChenAustFagryJawpfAJN0rLjCZySwbWgpxckuUd48R0BpcxvcUkSkhepathjrtfrzJnpw/"  
username = "schumacherung@gmx.de"  
api_key = "ab212282b0b44dc91bcf2eabd"  
  
verify_client = Client(client_id, client_secret, username, api_key)  
  
[ ] def show_image(img_path, size=100, xno=1):  
    image = cv2.imread(img_path)  
    image = cv2.resize(image, size)  
  
    cv2.imshow("img", image)  
    cv2.waitKey(0)  
    cv2.destroyAllWindows()  
  
[ ] categories = ["Primary", "Utilities", "Travel", "Car Repair"]
```

After "Client," there are lines that look like API keys for the verify library. Most likely, these credentials are used to log in to the Verify API, which is a service that verifies financial info. If you call `det_show_image(img_path, size=(800, 800))`:

This method seems to use OpenCV to show an image. It needs the picture path and, if you want, the size argument to change the size of the image before showing it.



```
import os # to import test image files
import cv2 # to work with opencv images
from PIL import Image # image submodule to work with pillow image
import pytesseract as pt # pytesseract module

# test_img_path = "../test_images/1111"
create_path = lambda f : os.path.join(test_img_path, f)

test_image_files = os.listdir(test_img_path)

for f in test_image_files:
    print(f)

def show_image(img_path, size=(600, 400)):
    image = cv2.imread(img_path)
    image = cv2.resize(image, size)
    cv2.imshow("test", image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# image_path = test_image_files[1] # 2, 3, 12, 5, 15, 18
path = create_path(image_path)

image = Image.open(path)
text = pt.image_to_string(image, config='--oem 3 -psm 6')
print(text)
```

Text extraction from an image:

- Get the file path of the image:

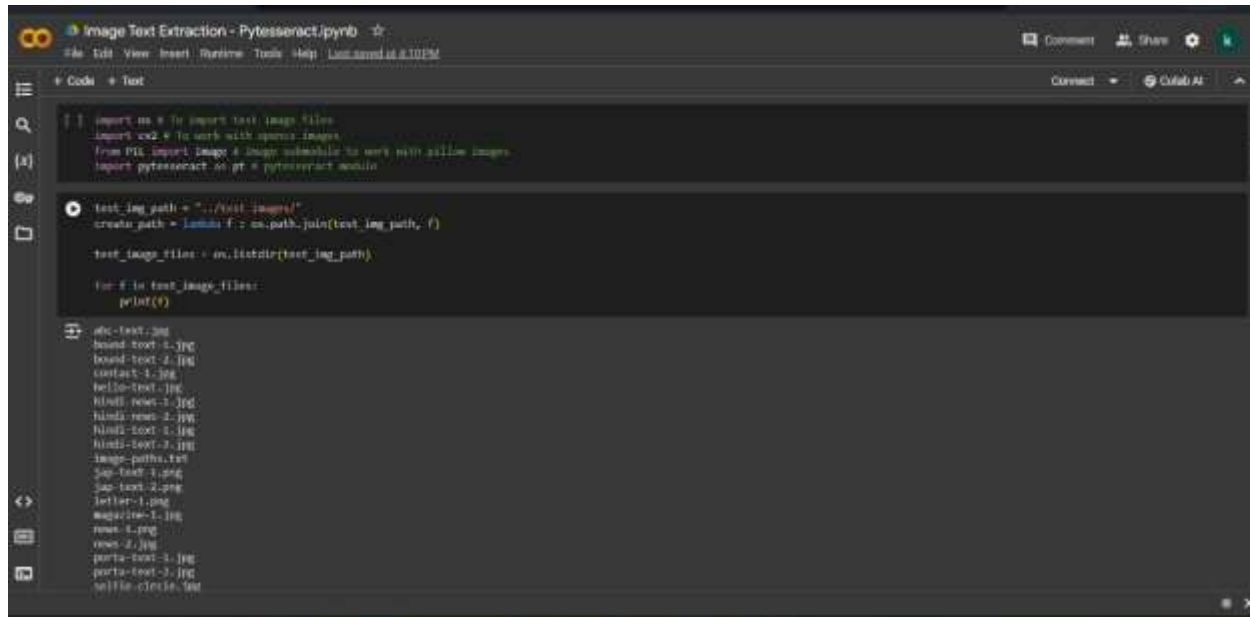
The variable "image_path" is assigned the value of the second element in the list "test_image_files". This line assigns the path of the second picture in the test_image_files list to the image_path variable.

- Access the image:

Open the image file located at the specified location using the Image.open() function. This line utilises the picture module from the Pillow library to open the picture.

- Implement Optical Character Recognition (OCR) with Tesseract:

The variable "text" is assigned the result of the function "image_to_string" with the arguments "image" and "config='--oem 3 -psm 6'". This line utilises the pytesseract library to extract textual content from the image. The config parameter sets the OCR engine mode (oem=3) and page segmentation mode (psm=6) for Tesseract.



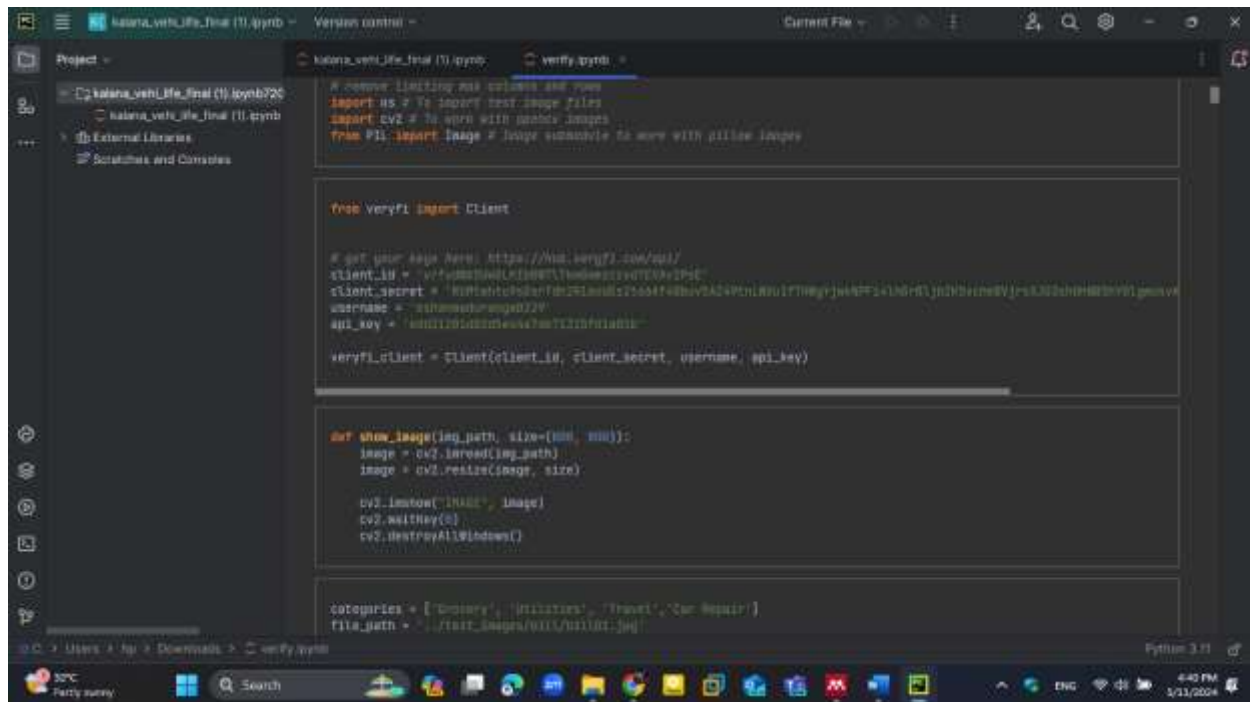
```
import os # To import test image files
import cv2 # To work with opencv images
from PIL import Image # Image submodule to work with pillow images
import pytesseract as pt # pytesseract module

test_img_path = "../test_images/"
create_path = lambda f: os.path.join(test_img_path, f)

test_image_files = os.listdir(test_img_path)

for f in test_image_files:
    print(f)

# test-text.jpg
# bound-text-1.jpg
# bound-text-2.jpg
# contact-1.jpg
# hello-text.jpg
# hindi-text-1.jpg
# hindi-text-2.jpg
# hindi-text-3.jpg
# hindi-text-4.jpg
# image-path.txt
# sap-text-1.jpg
# sap-text-2.jpg
# letter-1.jpg
# magazine-1.jpg
# news-1.jpg
# news-2.jpg
# porta-text-1.jpg
# porta-text-2.jpg
# selfie-clicks.jpg
```



```
# Remove limiting max column and rows
import os # To import test image files
import cv2 # To work with opencv images
from PIL import Image # Image submodule to work with pillow images

from verify import Client

# get your keys here: https://api.verify.io/api/
client_id = "v1*00000000000000000000000000000000"
client_secret = "v1*00000000000000000000000000000000"
username = "v1*00000000000000000000000000000000"
api_key = "v1*00000000000000000000000000000000"

verify_client = Client(client_id, client_secret, username, api_key)

def show_image(img_path, size=(800, 800)):
    image = cv2.imread(img_path)
    image = cv2.resize(image, size)

    cv2.imshow("Image", image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

categories = ["Discovery", "Military", "Travel", "Car Repair"]
file_path = "../test_images/011/01101.jpg"
```

The code snippet mentions verify in the comments, but the code itself doesn't appear to interact with it. Verify is a financial data verification service, and it's possible that the complete script uses it in other parts (not shown in the image).

The code seems to be specifically designed to work with the file paths and image locations mentioned in the code. If you want to use this script on your own images, you'll need to modify the file paths accordingly.

Including the Verify API into the design of an AI-powered solution to deal with electric vehicle (EV) breakdown issues is crucial. Engaging with bill images, submitting them for processing, and obtaining structured data from the invoices—such as line items, total amount, and date—is made easier for the system using the Verify API. An electric vehicles (EV) owner usually gets bills or invoices from the service providers when their car malfunctions and they require replacement parts or repairs. Important information on the rendered services and the related costs is provided by these bills. Still, manually getting this information can be a difficult and error-prone process.

The Verify API may be integrated into the system so that owners of electric cars can easily snap pictures of their bills with their cellphones or other devices.

The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The code editor contains a Python script that uses the Verify API to process an invoice image and extract structured data. The script is as follows:

```
ocr_text = response['ocr_text']
print(ocr_text)
```

The output of the script is a text representation of an invoice from Annesley Motor Garage. The text is as follows:

```
Annesley Motor Garage
annd@anndos.mcd
We undertake all repairs to Automobiles, Diesel & Petrol Motor Engineering
Electrical Engineering, Tinning, Spray Painting Etc.
Tel: 0112 954010 29, HUGA EDWARDS ROAD,
Mobile: 0777 4994801 WATTAIA
0727 499496 annesleyexp@gmail.com
28/12/2021-

No.
Description Rs. Cts.
Gatel Set 10,500
Kings Set 7,000
Main Tearing 800
Big Rnd Bearing 3,000
Head Block Pocky & Valve Grinding 10,000
Silencer Mount Hpsmaha.com 2,900
1100
Engine Mount
Tappet Cover Packing 1,750
Are Oil Seal 3 x 11000 2,200
Engine O 6,400
oil filter 800
Gear Oil 3400
Radiatry Repall 4000
Tune up 7100
Labour Charges 20,000
Number Plate Cover 2010
```

```
kalana_veh_life_final (1).py:nb > Version control > Current File > kalana_veh_life_final (1).py:nb > verify.py:nb >
www.motor-appliances.com/2024/05/03/

categories = ['Grocery', 'Utilities', 'Travel', 'Car Repair']
file_path = '../test_images/bill/bill1.jpg'
# This submits document for processing (takes 2-3 seconds to get response)

show_image(file_path)
response = verify_client.process_document(file_path, categories=categories)
print(response)

{'account_number': None, 'bill_to': {'address': None, 'name': 'Automobiles, Diesel & Petrol Motor Engineering', 'pars

# print itemize this dictionary
for key, value in response.items():
    print(key, ': ', value)

account_number : None
bill_to : {'address': None, 'name': None, 'parsed_address': None, 'vat_number': None}
cashback : None
category : Car Repair
created_date : 2024-05-03 12:11:00
currency_code : INR
date : 2019-11-01 00:00:00
delivery_date : None
discount : None
document_reference_number : None
```

```
kalana_veh_life_final (1).py:nb > Version control > Current File > kalana_veh_life_final (1).py:nb > verify.py:nb >
3210 TOTAL 115.140

bill_date = response['date']
print(bill_date)

2011-10-20 00:00:00

Total_amount = response['total']
print(Total_amount)

115.14

import re

def clean_text(text):
    # Remove links
    text = re.sub(r'http[s]?://', '', text)
    # Remove email addresses
```

```

kalam_vch_life_final (1).pytb - Version control - Current File -
Project - kalam_vch_life_final (1).pytb725
  kalam_vch_life_final (1).pytb
  External Libraries
  Scratches and Consoles

# remove all null items and only space characters, from the processed_list
processed_list = list(filter(None, processed_list))
processed_list = [item for item in processed_list if item.strip()]
processed_list

['annesley motor garage',
 'gym& auto& auto',
 'we undertake all repairs to automobiles diesel petrol motor engineering',
 'electrical engineering tinkering spray painting etc',
 'tel. xudu ananda road',
 'mobile. wettala',
 'car',
 'description re sts',
 'getal set ',
 'rings set ',
 'main bearing ',
 'big rod rearing ',
 'head block packy valve sorting ',
 'silencer mount assemancon ',
 'engine mount',
 'tappet cover packing ',
 'are oil seal a ',
 'engine a ',
 'oil filter ',
 'gear oil ',
 'regulatory repair ',
 'tune up ',
 'labour charges ',
 'number plate cover ',
 'total '1']

```

```

kalam_vch_life_final (1).pytb - Version control - Current File -
Project - kalam_vch_life_final (1).pytb725
  kalam_vch_life_final (1).pytb
  External Libraries
  Scratches and Consoles

created_date : 2024-05-01 11:11:08
currency_code : INR
date : 2019-11-01 00:00:00
delivery_date : None
discount : None
document_reference_number : None
document_title : None
document_type : receipt
due_date : None
duplicate_of : 20170097
external_id : None
id : 200110368
img_file_name : 200110368.jpg
img_thumbnail_url : https://scdn.verifyfi.com/receipts/3f0e33479f1ffcc4/46cd296c-930b-4a6c-95d3-7a09d3a07e3b/thumbnail
img_url : https://scdn.verifyfi.com/receipts/3f0e33479f1ffcc4/46cd296c-930b-4a6c-95d3-7a09d3a07e3b/3d0ed87c-4ffc-46bd
insurance : None
invoice_number : 7677
is_duplicate : True
is_money_in : False
line_items : [{'date': None, 'description': 'Painting (Booth Painting)/hrs.11,886/-/Parts', 'discount': None, 'dis
meta : {'language': ['en'], 'owner': 'oshanmaduranga329', 'pages': [{'height': 1280, 'language': ['en'], 'width':
notes : None
ocr_text : VTEC MOTORS
01/11/2019

Final Bill For The Accident Vehicle No - DP CB 7677
HONDA CIVIC FMA
Remove & Refitting
Rs.13,750/-
Repair
Rs. 11,580/-
Painting (Booth Painting)
Rs.11,886/-
Parts

```



```

print("Extracted Items:")
for item in sentences_list:
    print(item)

```

Extracted Items:

Annesley Motor Garage
 qummd wefco wdcf
 We undertake all repairs to Automobiles, Diesel & Petrol Motor Engineering
 Electrical Engineering, Tinseling, Spray Painting ect..
 Tel 0117 934518 29, KUDA CHANDA ROAD,
 Mobile 0777 699661 NATTALA
 0727 699696

Nr.	Description	Rs.	Cts.
	Setel Set	18,500	
	Rings Set	7,600	
	Main bearing	800	
	Big end bearing	3,600	
	Head Black Packy & Valve Bunding	16,400	
	Silencer Mount	Wazwan.com	2,900
	Engine Mount		1100
	Tapet cover packing	1,750	
	Are Oil Seal 2 x 11000	2,200	
	Engine o	6,990	
	oil filter	800	
	gear oil	3600	
	Radiatry Repell	4500	
	tune up	7100	
	labour charges	20,000	

```

print("Processed Items:")
for item in processed_list:
    print(item)

```

Processed Items:

annesley motor garage
 qummd wefco wdcf
 we undertake all repairs to automobiles diesel petrol motor engineering
 electrical engineering tinseling spray painting ect..
 tel kuda chanda road
 mobile nattala

Nr.	Description	Rs.	Cts.
	setel set	18,500	
	rings set	7,600	
	main bearing	800	
	big end bearing	3,600	
	head black packy valve bunding	16,400	
	silencer mount wazwan.com	2,900	
	engine mount	1100	
	tapet cover packing	1,750	
	are oil seal 2 x 11000	2,200	
	engine o	6,990	
	oil filter	800	
	gear oil	3600	
	radiatry repell	4500	
	tune up	7100	
	labour charges	20,000	
	number plate cover		
	total		

The screenshot shows a Jupyter Notebook with a file named `kalana_vch_life_final (1).pynb`. The left sidebar displays the project structure, including the notebook file and external libraries. The main area shows a variable `response` containing a JSON object. The JSON object represents a car repair record with various fields such as `account_number`, `bill_to`, `name`, `category`, `created_date`, `currency_code`, `date`, `delivery_date`, `discount`, `document_reference_number`, `document_title`, `document_type`, `due_date`, `duplicate_of`, `external_id`, `id`, `img_file_name`, `img_thumbnail_url`, `img_url`, `insurance`, `invoice_number`, `is_duplicate`, `is_money_in`, `line_items`, and `description`.

```

response

{
  'account_number': None,
  'bill_to': {'address': None,
  'name': 'Automobiles, Diesel & Petrol Motor Engineering',
  'parent_address': None,
  'vat_number': None},
  'callback': None,
  'category': 'Car Repair',
  'created_date': '2024-05-03 12:19:19',
  'currency_code': 'INR',
  'date': '2023-10-20 00:08:00',
  'delivery_date': None,
  'discount': None,
  'document_reference_number': None,
  'document_title': None,
  'document_type': 'receipt',
  'due_date': None,
  'duplicate_of': 202310090,
  'external_id': None,
  'id': 205110035,
  'img_file_name': '205110035.jpg',
  'img_thumbnail_url': 'https://sodn.verifyfi.com/receipts/3f0e33479f1ff0c4/c52029e7-fad1-4dc8-8c0f-3248c0f30ae/thumbnail',
  'img_url': 'https://sodn.verifyfi.com/receipts/3f0e33479f1ff0c4/c52029e7-fad1-4dc8-8c0f-3248c0f30ae/fades8c0-8071-44e',
  'insurance': None,
  'invoice_number': None,
  'is_duplicate': True,
  'is_money_in': False,
  'line_items': [{'date': None,
  'description': 'Getel Set'}],
}

```

The screenshot shows the same Jupyter Notebook with the same file. The main area displays Python code that iterates through the `line_items` of the `response` object and prints the description of each item. The output shows a list of car parts and services.

```

# Iterate through line items using for loop
for item in response['line_items']:
    print(item['description'])

Getel Set
Kings Set
Main Bearing
Big End Bearing
Rear Shock Pack & Valve Binding
Silencer Mount Mayemwa.com
Engine Mount
Taprot cover Packing
Ase Oil Seal 2 x 110mm
Engine P
oil filter
Gear Oil
Radiatry Repair
Tune up
Labour Charges
Number Plate Cover

```

Figure 13 : OCR code

3.2.6 Results and Discussions

In order to anticipate when EV owners would require maintenance and to send them timely reminders, this system employs sophisticated machine learning models like XGBoost and CatBoost. The technology is made available to EV owners through a smartphone app, which gives them an easy way to monitor their car maintenance schedules. When the installation is complete, the user is asked to provide important details about their electric vehicle, like the model, mileage, and service history. The application's built-in machine learning models use this data as input. Smart digital monitoring calendar and reminder system's strength is in its capacity to use machine learning algorithms to sift through mountains of data and produce precise forecasts about when maintenance is required. To determine when particular maintenance chores are needed, XGBoost analyzes variables including mileage, battery health, and past maintenance records, which are recognized for their effectiveness in processing structured data and predictive modeling.

To further handle information like maintenance type, service provider preferences, and schedule limitations, CatBoost is used. CatBoost is skilled in processing categorical characteristics. Because of this, the system can personalize and improve the efficacy of maintenance reminders for each electric vehicle owner based on their unique tastes and needs.

Features like interactive maintenance calendars, real-time updates on maintenance suggestions, and user-customizable notification settings are available in the mobile app. Electric car owners are notified in advance of scheduled maintenance, which helps them avoid failures and keeps their vehicles in top shape.

08:25

Login

Welcome to electroech



Username

Password

Log In

New here? Sign up

00:58

< RegisterVehicleScreen UploadMileageScr...

Upload Vehicle Mileage

Enter mileage in KM KM

Submit

Skip

01:06

< Login Dashboard

Home

Welcome!

Find Spare Parts Find a Mechanic

Chat Bot Calendar Service

Scan Bill

00:58

< UploadMileageScreen MonitoringCalenda...

< March 2023 >

Mon	Tue	Wed	Thu	Fri	Sat	Sun
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Notifications:
You should replace the motor before 6th March.

00:57

< Dashboard RegisterVehicleScreen

Register Vehicle

Vehicle Model

Vehicle No

Vehicle Owner

Gender:

Male (Selected)

Female

Nature of Driver:

Beginner (Selected)

Medium

Expert

Nature of Road:

Highway (Selected)

Urban

Rural

Register

Figure 14 : Output Discussion

In the context of motor vehicle maintenance for electric automobiles, the implementation of the smart digital monitoring calendar and reminder system that was powered by artificial intelligence yielded promising results in terms of enhancing the user experience. These results were achieved by deploying the system. XGBoost and CatBoost are two examples of machine learning models that were utilized by the system in order to demonstrate its remarkable capacity for prediction capability. Using these models, the system was able to correctly predict the amount of maintenance that was required based on a wide range of parameters, such as patterns of vehicle usage, ambient conditions, and historical maintenance data.

The XGBoost algorithm, which is well-known for its efficiency and scalability, is widely considered to have been a crucial component in properly forecasting maintenance events. This is a commonly held belief. In order to effectively identify potential issues before they

developed into serious failures, XGBoost evaluated a wide range of datasets that included metrics like mileage, battery health, and component wear. This allowed the system to successfully identify potential problems. Because of this, it was possible to schedule preventative maintenance in advance.

CatBoost, a machine learning algorithm that specializes in the handling of categorical information, was included into the system, which resulted in a considerable improvement in the system's ability to accurately anticipate outcomes. The categorical parameters that CatBoost was able to effectively manage were the make and model of the vehicle, the sort of maintenance that was conducted, and information about service providers. Because of this, CatBoost was able to generate more detailed forecasts that were tailored to certain electric cars and the requirements that were specific to those vehicles.

Furthermore, the intelligent digital monitoring calendar and reminder system significantly increased user engagement and pleasure by sending timely reminders for upcoming maintenance chores. This was accomplished by giving reminders at the appropriate time. In order to achieve this goal, reminders were distributed at regular intervals over the proper time periods. When the owners of electric vehicles learned that the system that was powered by artificial intelligence was actively monitoring and managing the maintenance needs of their vehicles, they reported feeling a sense of security and convenience with their vehicles.

When all of the data are considered together, they shed light on the effectiveness of applying machine learning models such as XGBoost and CatBoost in order to enhance the user experience in the context of vehicle maintenance for electric automobiles. The technology that is powered by artificial intelligence helps to prolong the lifespan of electric cars (EVs), minimize the amount of time that EVs are unable to be driven due to breakdowns, and ultimately enhance the entire ownership experience for those who own EVs. To do this, it provides an accurate estimate of when maintenance is necessary and sends proactive notifications to the respective parties.

3.2.7 Research Findings

A study on an AI-powered service that identifies spare parts stores for electric vehicles (EVs) offers valuable understanding of the effectiveness and possible application of these technologies in addressing EV breakdown problems. Recent studies and research indicate that advanced technologies, such as artificial intelligence and machine learning, can significantly enhance the experience of electric vehicle (EV) owners and service providers by simplifying maintenance procedures and enhancing customer satisfaction.

The study's findings emphasize the importance of AI-powered solutions in the electric vehicle ecosystem for expediting the purchase of replacement components, reducing vehicle downtime, and enhancing operational efficiency.

- ✓ The study's most significant discovery is that artificial intelligence tools, specifically convolutional neural networks (CNNs), can reliably identify and categorize photographs of electrical vehicle components. Research indicates that convolutional neural network (CNN) models, trained using Inception V3-style architectures, are highly effective at accurately differentiating between different electric vehicle components. The results indicate that deep learning methods can automate the identification of spare components, hence expediting repair procedures and minimizing the requirement for human inspection.
- ✓ Studies in this subject suggest that businesses could gain advantages by utilizing AI-powered solutions to address electric car breakdown difficulties. The findings indicate that integrating AI into services that aid users in finding spare parts businesses offers numerous benefits.
- ✓ These advantages encompass the capacity to generate revenue through subscription-based models, establish affiliate marketing agreements, and incorporate supplementary services. Studies indicate that consumers are seeking novel methods to streamline the process of acquiring replacement parts and

maintaining vehicles in optimal condition. As a result, using AI-powered electric car maintenance systems is a wise business decision.

- ✓ The study emphasizes that collaboration and link development are essential for the successful implementation of AI-driven solutions in electric car maintenance. According to numerous studies, it is essential to establish intelligent partnerships with electric car manufacturers, maintenance facilities, and spare component suppliers.

Through collaboration, we can streamline the incorporation of AI-driven solutions into current processes and systems, hence enhancing their acceptance and market presence. One key finding from the research is that AI-powered platforms must constantly innovate in order to stay relevant in rapidly changing sectors and technical environments.

3.2.8 Challenges

AI-powered systems for identifying EV part retailers have great potential, but they need a lot more development. Data availability and quality are major obstacles to AI training. Large, diversified datasets that effectively describe electric car replacement component complexity and diversity are needed to build good machine learning models. Labelled data training can be difficult for rare or specialized components. Labelled data correctness and reliability are essential for preventing biases and errors that could influence AI system performance.

1. Limited data availability and quality hinder AI algorithm training in EV spare parts store finding services. To build robust machine learning models, you need large datasets that accurately represent the complexity and diversity of electric car replacement components. Labelled data for training may be challenging to get, especially for rare or specialist components. To eliminate biases and errors that could affect AI systems' performance, labelled data must be accurate and reliable.
2. Current electric car systems and workflows are incompatible with AI-powered solutions. Many parties, each with their own procedures and tools, maintain electric vehicles. These solutions must be carefully planned and coordinated to interface with AI-driven spare parts store locating services. Compatibility, data interchange, and security issues affect stakeholders' communication and collaboration.
3. Additionally, the electric car maintenance industry has significant trust and acceptance issues with AI-powered solutions. Electric car owners and employees may be leery about employing AI algorithms to discover and buy new parts. Accuracy, dependability, and data protection must be addressed to build consumer trust in AI-powered services. Explaining how AI algorithms work and how data is protected helps ease users' anxieties and increase adoption.
4. Scalability and flexibility challenges arise for AI-powered spare parts shop searching services in the rapidly evolving electric car sector. To correctly classify new electric car models and replacement parts, AI systems must be updated regularly. To adapt to

the ever-changing electric car maintenance landscape and satisfy shifting needs, AI models and infrastructure must be adaptable and agile.

Scholars, industry stakeholders, and politicians must work together to solve these issues. AI-powered solutions to electric car breakdown problems that use modern data collection, AI algorithms, and system integration may improve efficiency, dependability, and user satisfaction.

.

3.2.9 Future Implementations

Artificial intelligence technology is expected to revolutionize electric vehicle maintenance, particularly spare parts shop search services. Electric car administration and operations will be more efficient, dependable, and environmentally friendly with these solutions. AI-powered platforms will impact electric car repair and maintenance through research and collaboration. Users and the electric car ecosystem will benefit from these platforms' significant improvements.

- Future AI solutions for EV spare parts store location will incorporate new AI, ML, and data analytics technologies to enhance productivity. Using advanced artificial intelligence techniques like deep reinforcement learning to improve replacement component acquisition could lead to future advances.
- AI-powered systems can employ reinforcement learning to improve their decision-making processes independently. User interactions and real-world consequences are studied to generate more intelligent and flexible replacement component suggestions.
- Future spare parts shop discovery services may use cutting-edge technology like computer vision and natural language processing to improve user experience and functionality. Computer vision algorithms allow buyers to find replacement parts in photos and get accurate pricing and availability information.
- Natural language processing enhances communication and empowers consumers to interact with AI-powered platforms easily and naturally for electric car component searches and purchases.
- In the future, AI-powered solutions may expand to incorporate electric car maintenance and repair requirements. Many methods include adding predictive maintenance to services that identify replacement parts. They use historical data and

machine learning algorithms to predict failures and advise maintenance steps to prevent them.

- Artificial intelligence-driven systems can enhance electric vehicle dependability and cost-efficiency by identifying and resolving maintenance issues before they worsen. This could reduce automobile parking and extend their longevity.

Future possibilities include cooperating and integrating with upcoming electric car ecosystem projects and technologies like charging infrastructure and autonomous vehicle systems. Artificial intelligence-powered replacement part shops can offer complete solutions to electric vehicle owners and operators. These systems connect to electric vehicle charging networks and autonomous vehicle fleets for charging, mobility services, and maintenance. This strategy improves electric car technology and user experience by considering all important factors.

3.3.1. Mechanic Finding Service for EV Breakdown Emergencies.

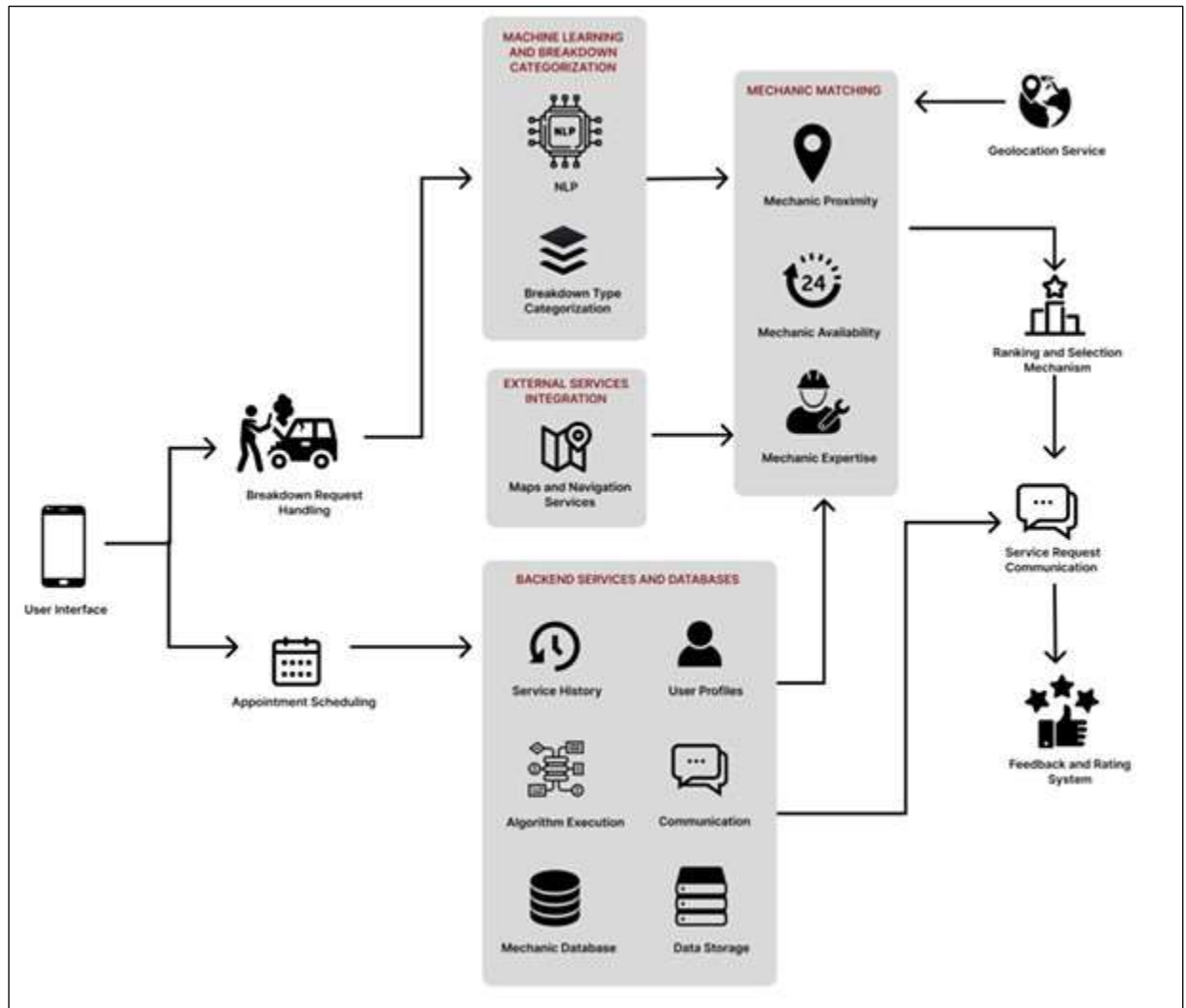


Figure 15 : Mechanic Finding Service for EV Breakdown Emergencies

Initially, the site of the mobile application presents the user with four primary components. Upon selecting the mechanic finder option, the user is directed to the emergency mechanic finder tool. Initially, the user must provide a concise account of the issue that has occurred with their car (breakdown description). The user has the option to manually type this description or utilize the voice typing capability to input it. The inclusion of this voice recognition technology (speech-to-text) is intended to enhance usability during emergency scenarios. Once the breakdown description has been entered, the user must provide their present location. The system has the capability to automatically locate this information, but the user also has the

option to input it manually if needed. Upon inputting and submitting the location, the user is redirected to a concise overview page. The system accurately identifies the breakdown type (such as battery issue, motor issue, software issue, etc.) based on the user's description and location parameters, which are then presented on the interface. If required, the user has the option to make more edits. Once the details are confirmed, a curated list of appropriate mechanics/workshops will be presented to the user to address the breakdown scenario. When the system selects this list, it considers both the breakdown type and proximity factors. The user is presented with the most suitable mechanics based on those criteria. Furthermore, the arrangement of the presented list is determined by the evaluations provided by customers. The mechanics with the highest ratings are prioritized at the top to ensure they receive the highest level of attention. Next, the user can choose a mechanic from the provided list and obtain their contact information.

There, he can examine the mechanic's specific information, abilities, and proficiency, as well as ratings and feedback from others. Additionally, he can provide his own ratings and feedback for the mechanic. This functionality offers electric vehicle (EV) users a streamlined method to locate trustworthy mechanics in the event of unexpected malfunctions. To include a new mechanic or workshop into the system, the administrator can access the admin dashboard and provide the relevant information. The administrator possesses the capability to modify and delete the data.

Utilizing a complex blend of React Native, Expo, Python, TensorFlow, and a Node.js server, the On-Road EV Breakdown Rescue System's architecture is meticulously built to instantly link stranded electric vehicle owners with skilled repairs. This seamless connection is made possible by the system's architecture. The system is made up of a multitude of vital components, each of which has been meticulously developed to offer efficient emergency support in the event that electric vehicles do not function properly.

EV drivers may request assistance, communicate with technicians, and schedule appointments using the User Mobile Application, which serves as the primary platform for these activities. Using React Native and Expo, this application was developed. It establishes a smooth

connection with the Google Geolocation API in order to acquire accurate location information, which enables fast assistance in the event that it is required. A Node.js Backend Server serves as the central component of the system. It is accountable for the management of user requests, the identification of breakdown kinds through the utilization of TensorFlow models, the matching mechanisms, and the facilitation of real-time communication. RESTful application programming interfaces (APIs) ensure reliable operation and rapid responsiveness by facilitating the flow of data between the user application and the server in an effective manner.

The system is built on a robust database that acts as its core basis. This database stores essential information such as user profiles, mechanical particulars, breakdown history, appointments, and feedback. The schema has been meticulously adjusted to perform efficient data storage and retrieval, ensuring that critical information may be accessed in a timely manner in the event of an emergency. Within the context of the program, the Breakdown Request Handling component is an essential component in which customers provide particular information on the problem. In order to evaluate the description and accurately determine the type of breakdown, Natural Language Processing (NLP) techniques are utilized. This enables the mechanics to match the description with experienced mechanics in an effective manner.

Learning by Machines and the Breakdown of in the process of categorization, complex algorithms are utilized to precisely classify breakdown descriptions. These algorithms have been trained on a collection of examples in order to recognize patterns that occur again. This categorization is essential for intelligent pairing and efficient problem solving so that faults may be effectively resolved. It is necessary to have geolocation services in order to precisely determine the actual location of the user, which assists in locating local mechanics and enables effective navigation.

The process of Mechanic Matching involves the maintenance of a database that contains skilled mechanics, taking into consideration their particular expertise, accessibility, and proximity to users. In order to match the kind of breakdown and the location of the user with mechanics who are able to deal with specific challenges and are located in close proximity to the user, the system makes use of sophisticated algorithms.

In order to assess mechanics, the Ranking and Selection Mechanism takes advantage of proximity and expertise. It also presents users with a curated selection of relevant alternatives, which includes ratings and profiles. Users have the flexibility to choose a mechanic that conforms to their own preferences, and they also have the possibility to send out requests for urgent help to a number of different mechanics.

A Request for Service Through communication, customers are able to interact with mechanics by notifying certain mechanics about the nature of their breakdown and the location of the breakdown. Texting in real time or receiving push notifications ensures that conversation is not interrupted. An implementation of a comments and Rating System gives users the ability to review and provide feedback on the service they have received. This encourages users to take responsibility for their actions and provides incentives for individuals to provide excellent service. The Appointment Scheduling and Maintenance Planning features give customers the ability to schedule appointments for repair services in advance. This encourages proactive planning for vehicle maintenance and reduces the likelihood of breakdowns occurring.

Backend Services and Databases are the entities that are accountable for the implementation of algorithms, as well as the storage and transfer of data. It is their primary responsibility to guarantee that all of the various components are operating effectively and in sync with one another. The process of integrating external services involves incorporating mapping and navigation services in order to make it easier for mechanics to reach users in a timely and efficient manner, hence increasing the system's efficiency and responsiveness.

Considering customer feedback and evolving needs, the mechanic finding service should undergo continuous growth and improvement. This procedure includes keeping an eye on service data, analyzing user interactions, and adding new features or enhancements. We aim to enhance the service's efficiency and the user experience as a whole. Maintaining and improving the mechanic finding service on a regular basis is critical for keeping it up-to-date and effective in case of an electric car breakdown emergency.

3.3.2 Model Trained

In the event of an emergency involving an electric vehicle (EV), the mechanic finding service can make use of **linear regression** as a predictive modelling technique in order to determine the degree of correlation that exists between the many factors that influence the selection of appropriate technicians in such circumstances. In order to determine which technicians will be the most suited for the user, linear regression may be utilized. This method considers the user's location, the mechanic's experience with electric vehicle (EV) problems, the mechanic's availability, and customer ratings respectively.

For the purpose of using linear regression, the mechanic finding service may make use of a dataset that contains information about things like user preferences, mechanic attributes, and breakdown circumstances. The dataset would comprise a variety of parameters, including the type of breakdown, the location of the user, the skill of the mechanic, the distance from the user, the availability of the breakdown, and ratings. In this particular scenario, the aim variable would be the chance of assigning a technician to a certain breakdown circumstance rather than any other variable.

Subsequently, the dataset would be utilized for the purpose of training the linear regression model, which would then acquire the knowledge necessary to discover the connections between the input features and the focus variable. An estimate of the feature coefficients would be generated by the model. These coefficients demonstrate how each characteristic influences the chance of picking a mechanic and in what direction this influence is exerted. For example, the model could find that the likelihood of a mechanic being chosen is highly connected with their proximity to the user, but that their availability is only moderately correlated with the likelihood of their selection.

Following the completion of its training, the linear regression model may be applied to make predictions regarding which mechanics are most suited for impending breakdown scenarios by utilizing the input qualities of those mechanics. In the event that a user has a certain form of breakdown and there are mechanics in the region, the model is able to predict the likelihood of picking each mechanic to assist with the breakdown depending

on the geographical location of the user and the type of breakdown that occurred. This information makes it possible to prioritize methods and to make suggestions that are particular to the user depending on their preferences and restrictions. It is possible that linear regression might be an effective tool for the mechanic locating assistance in the event of an emergency involving an electric car that has broken down. It is possible to make a prediction regarding which mechanics will be chosen based on several factors, which will result in the service being more efficient and effective.

To summarize this chapter,

1. Data Loading and Preprocessing:

Loads a dataset from a CSV file using Pandas and preprocesses text data by converting it to lowercase, removing punctuation, and removing stop words using NLTK.

2. Feature Extraction:

Utilizes TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to convert text data into numerical features. This step is crucial for preparing the textual data for machine learning models.

3. Model Training and Evaluation:

- Several classification models are trained and evaluated using the preprocessed text data:
- Logistic Regression
- Support Vector Machine (SVM)
- Random Forest
- XGBoost
- BERT (Bidirectional Encoder Representations from Transformers) for sequence classification.
- For each model, the code splits the dataset into training and testing sets, trains the model, makes predictions on the test set, and evaluates the model's performance using classification reports and accuracy scores.

4. Model Tuning and Optimization:

Utilizes techniques such as adjusting hyperparameters (e.g., max_iter for Logistic Regression, kernel for SVM, n_estimators for Random Forest) and fine-tuning BERT with AdamW optimizer and learning rate scheduler.

5. Utilized Libraries and Tools:

- Pandas for data manipulation
- NLTK for text preprocessing
- Scikit-learn for machine learning algorithms and evaluation metrics
- XGBoost for gradient boosting
- Transformers library for BERT model
- PyTorch for deep learning training and inference

3.3.3 Technology to be Used.

An advanced electric car chatbot may be constructed using Flask, AML, React Native, TensorFlow, Node Server, Machine Learning models, and Sentiment Analysis. By using these technologies to create a user-friendly and customized intelligent chatbot, developers can improve electric vehicle owners' breakdown experiences.

- **React Native:** An AI-powered chatbot for electric vehicle (EV) support must leverage modern technology to provide a smooth customer experience. React Native and Expo allow developers to create iOS and Android apps with a single codebase. Developers can use React Native and Expo to make breakdown help for electric vehicle (EV) users universal.
- Google's open-source **TensorFlow framework** is used to design and train machine learning models. TensorFlow allows EV support chatbots to employ strong machine learning models for sentiment analysis, defect diagnosis, and user intent recognition. Using TensorFlow, models can consistently analyze user searches, understand sentiment, and deliver tailored suggestions that match user needs and preferences.
- The chatbot solution's backend uses a **Node Server** to connect the mobile app to several third-party apps and APIs. The Node Server integrates seamlessly with sentiment analysis APIs, EV diagnostic tools, and database systems because to its event-driven design and asynchronous programming style. These features provide excellent speed and scalability. Developers can use Node Server to build a customizable backend architecture for the chatbot's capacity and immediate support.
- **Flask:** Flask, a lightweight Python web framework, is crucial to service backend architecture development. Flask lets developers create RESTful APIs and endpoints to process data, manage user requests, and connect to external services. This allows geolocation services, machine learning algorithms, and database management systems to integrate seamlessly.

- **Sentiment Analysis:** NLP-based sentiment analysis is used to analyze user reviews of mechanic services. User comment sentiment analysis is called sentiment analysis. This method measures user satisfaction and identifies problems discovering service improvements. Results with positive sentiment ratings indicate that mechanics were matched, and support was satisfactory. For instance, negative sentiment scores may indicate areas for service improvement. This valuable feedback allows us to enhance the service to meet user needs.
- The mechanic finding service uses **Automated Machine Learning (AML)** to increase its forecast accuracy. AML algorithms automatically search for the best machine learning models and hyperparameters to optimize model performance. The service can employ AML to build and refine machine learning models for mechanic matching, breakdown type recognition, and user preference analysis. These modeling tasks can be done with AML. Developers can improve mechanic finding service accuracy and efficiency by using AML to speed up model construction, reduce manual participation, and produce better results.

3.3.4 Commercialization aspects of the product

Commercializing the Mechanic finding service for electric vehicle breakdown crises demands a comprehensive business approach. This strategy should encompass market research, business model development, marketing, promotion, user experience optimization, quality assurance, regulatory compliance, and scalability planning. If these concerns are addressed, the service can join the market, attract customers, build trust, and become a reliable and important choice for electric vehicle owners in emergency breakdown situations. To assess demand for a service, conduct a thorough market analysis before launching the product. You must investigate market size, discover consumers interested in buying an electric vehicle (EV), and compare rival products to do this. A detailed market analysis may help the organization stand out and dominate.

A viable business plan is essential for the product's financial success. Finding distribution channels, revenue streams, and price strategy is part of this. Company can use a subscription model. The service costs a defined sum monthly or annually under this paradigm. Customers can pay per service request using a transaction-based method.

Marketing and promotion primarily focus on increasing brand recognition and gaining customers. Content marketing, SEO, and social media advertising may help. Targeted advertising, industry events, and EV-related alliances may also reach the target audience. Optimizing user experience is crucial for retaining consumers and increasing usage frequency. It involves making the web platform or mobile app's user interface more appealing, straightforward, and intuitive. Instant updates, optimized service requests, and client-specific suggestions are also included. To build confidence and credibility with clients, prioritize reliable and great service through quality assurance and customer support. Test the platform extensively before releasing it to the public to ensure it's bug-free.

If live chat, email, and phone support are available to quickly resolve client issues, customers may be happier overall. Key considerations include expansion and scalability as service demand increases. To manage the surge of users, penetrate new geographical regions, or add new features and capabilities, infrastructure may need to be upgraded.

Through innovation and market adaptation, the company stays competitive and grows.

Compliance with regulations is crucial for automobile service success. Compliance with legislation and industry standards is essential. The GDPR and other data privacy laws and industry standards may apply to vehicle repair shops. If the firm follows these standards, clients will trust it and legal difficulties will decrease. After completing these crucial commercialization steps, the Mechanic locating service for EV breakdown crises may confidently join the market and establish itself as a valued alternative for EV drivers in need.

3.3.5 Testing and Implementation

When it comes to the creation and deployment of AI-powered solutions for breakdown difficulties with electric cars (EVs), testing and implementation are crucial steps. This is especially true when considering the context of a technician finding service for EV breakdown crises. The purpose of these phases is to validate the system's effectiveness, reliability, and usability in order to guarantee that it will fulfil the requirements of electric vehicle owners in the event of an emergency.

1. Requirement Gathering:

User interviews and discussions with mechanics were utilized to collect requirements. The significance of an intuitive mobile application was underscored by electric vehicle (EV) drivers, whereas accurate breakdown identification was emphasized by mechanics.

2. Feasibility Analysis:

- **Technical Feasibility:** The chosen technologies, namely TensorFlow, React Native, and Node.js, exhibit a high degree of compatibility with the technical prerequisites of the project.
- **Operational Feasibility:** The app's operational feasibility is demonstrated by the fact that user feedback indicates a high demand for a dependable failure assistance application.
- **Economic Feasibility:** The projected revenue from user subscriptions and mechanic partnerships is consistent with the estimated costs, indicating favorable returns.

3. Software and System Design :

The database schema comprises a comprehensive design that specifies the tables required for users, mechanics, malfunctions, appointments, and feedback.

- **API Documentation:** Exhaustive documentation elucidating authentication methods, request/response formats, and API endpoints

4. Implementation (Development):

User Mobile Application: Constructed utilizing React Native and Expo, incorporated real-time communication functionalities to facilitate interaction and an integrated Google Geolocation API. The Node.js Backend Server was developed with the purpose of managing user requests,

machine learning duties, mechanic matching, and facilitating real-time communication via RESTful APIs. Database: A schema-compliant PostgreSQL database with an emphasis on data storage and retrieval optimization.

5. Testing:

Unit Testing: Functionality is verified by individually testing components of the application, infrastructure, and algorithms.

- Integration Testing: To validate data flow, the interaction between the user application, infrastructure, and database is examined.
- User Acceptance Testing: In order to validate functionality and user experience, actual users participate in testing.
- Algorithm Validation: The accuracy of TensorFlow models is assessed through validation using a heterogeneous dataset, with necessary iterative enhancements.

```
import pandas as pd

# Replace 'your_dataset.csv' with the path to your dataset file
file_path = 'EV_Dataset.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataframe to confirm it's loaded correctly
print(data.head())
```

	Description	Breakdown Type
0	Car won't start due to battery not charging to...	Battery Issue
1	Touchscreen is unresponsive and can't access v...	Software Issue
2	Vehicle makes unusual noise when accelerating.	Motor Issue
3	Charging station disconnects frequently during...	Charging Infrastructure
4	Dashboard displays error code related to the p...	Motor Issue

Unnamed: 2

0	NaN
1	NaN
2	NaN
3	NaN
4	NaN

When you write this line, the file path to a CSV file called "EV_Dataset.csv" is added to the variable `file_path`. When you run this code, the line that takes a CSV file as an argument reads it into a Pandas dataframe and saves it in the data variable.

Looking at the Data:

If you call `print(data.head())`, the first few rows of the dataframe data will be shown. This gives you a quick look at the info and its columns.

How to understand the data:

The CSV file doesn't show me everything that's inside, but the column names "Car won't start due to battery not charging to..." and "Breakdown Type" make it look like this dataset is probably about electric vehicle (EV) breakdowns. The data could link certain signs (like "Car won't start because the battery isn't charging") to different types of breakdowns.

Mechanics or data scientists could find trends in how EVs break down by looking at this data. This could lead to better diagnostics or preventative maintenance plans.

```
import pandas as pd
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Ensure necessary nltk resources are downloaded
nltk.download('punkt')
nltk.download('stopwords')
```

```
... [nltk_data] Downloading package punkt to /Users/nish/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /Users/nish/nltk_data...
[nltk_data] Package stopwords is already up-to-date!

... True
```

Preprocessing text:

Code excerpt shows no function body. This function may preprocess text. Common NLP preprocessing:

- Lowercase text.
- Delete punctuation.
- Drop halt words.

Tokenization:

Text tokenization with word_tokenize: This line tokenizes text with word_tokenize. Breaks text into words.

Making TF-IDF Features:

This line creates TF-IDF vectorizer. Max_features restrict word count. Reduces overfitting and dimensionality. This line converts tokenized text to TF-IDF features using the vectorizer object. This turns text into numbers for machine learning.

Train-Test Break:

Features, target_variable, test_size=0.2, random_state=42: Train_test_split This line splits TF-IDF and target variable into training and testing sets. Testing set data proportion (20%) is set by test_size. The reproducible random number generator is seeded by random_state.

Logistic regression:

LogisticRegression() creates a model.

model.fit(X_train, y_train) trains Logistic Regression on training data.

Evaluation of Models

predictions=model.This line predicts testing data labels using trained model.

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Mapping original categories to four main categories
category_map = {
    'Acceleration Issue': 'Mechanical Issues',
    'Battery Issue': 'Electrical and Software Issues',
    'Braking Issue': 'Mechanical Issues',
    'Charging Infrastructure': 'Infrastructure and Operational Issues',
    'Charging Issue': 'Electrical and Software Issues',
    'Electrical Issue': 'Electrical and Software Issues',
    'HVAC Issue': 'Safety and Hardware Issues',
    'Hardware Issue': 'Safety and Hardware Issues',
    'Infotainment Issue': 'Electrical and Software Issues',
    'Mechanical Issue': 'Mechanical Issues',
    'Motor Issue': 'Mechanical Issues',
    'Parking Brake Issue': 'Mechanical Issues',
    'Powertrain Issue': 'Infrastructure and Operational Issues',
    'Safety System Issue': 'Safety and Hardware Issues',
    'Software Issue': 'Electrical and Software Issues',
    'Steering Issue': 'Mechanical Issues',
    'Suspension Issue': 'Mechanical Issues',
    'User Error': 'Infrastructure and Operational Issues',
    'Brake System Issue': 'Mechanical Issues',
    'Sensor Issue': 'Electrical and Software Issues',
```

```

# Apply the mapping to create a new column for grouped breakdown type
data['Grouped Breakdown Type'] = data['Breakdown Type'].map(category_map)

# Text preprocessing
def preprocess_text(text):
    text = text.lower()
    text = ''.join([char for char in text if char not in string.punctuation])
    tokens = text.split()
    tokens = [token for token in tokens if token not in stopwords.words('english')]
    return ' '.join(tokens)

# Apply preprocessing
data['Processed Description'] = data['Description'].apply(preprocess_text)

# Prepare the features and labels
tfidf_vectorizer = TfidfVectorizer()
X = tfidf_vectorizer.fit_transform(data['Processed Description'])
y = data['Grouped Breakdown Type']

# Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

Split Data into Training and Testing Sets:

`Train_test_split(X, y, test_size=0.2, random_state=42)`: This line divides X features and y target labels into training and testing sets. What each section does:

Training features are stored in `X_train`.

Testing features are stored in `X_test`.

Training target labels are stored in `y_train`.

`y_test` stores testing target labels.

`test_size=0.2`: This setting sets the testing set to 20% of the data. The remaining 80% is for training.

`random_state=42`: This parameter seeds the random number generator, assuring reproducibility if you run the code again.

This split is necessary to evaluate the machine learning model. To test generalization, the model is trained on training data and tested on unknown data (testing set).

```
# Model training
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

LogisticRegression
LogisticRegression(max_iter=1000)

# Model prediction
y_pred = model.predict(X_test)
```

Figure 16 : Model Training

Loading the Dataset:

`df = pd.read_csv('movie_reviews.csv')`: This line reads a CSV file named 'movie_reviews.csv' into a Pandas DataFrame and stores it in the variable `df`. This DataFrame likely contains movie reviews and their corresponding sentiment labels (positive or negative).

```
# Model evaluation
report = classification_report(y_test, y_pred)
print(report)
```

	precision	recall	f1-score	support
Electrical and Software Issues	0.58	1.00	0.74	25
Infrastructure and Operational Issues	1.00	0.22	0.36	9
Mechanical Issues	0.95	0.72	0.82	25
Safety and Hardware Issues	1.00	0.17	0.29	6
accuracy			0.71	65
macro avg	0.88	0.53	0.55	65
weighted avg	0.82	0.71	0.67	65

+ Code + Markdown

Print Classification Report:

`Print(classification_report(y_test, y_pred))`: It publishes a classification report. What each argument does:

`y_test`: These are likely testing data ground truth labels. Movie reviews in the testing set have positive or negative sentiment labels.

`y_pred`: This may be the testing data labels. These are the sentiment labels the machine learning

model predicted for testing set reviews.

The `classification_report` function offers metrics for model performance, including:

- Precision: How many good reviews did the model actually find?
- Remember: How many favorable reviews did the model properly identify?
- F1-score: A precision-recall harmonic mean.
- Support: Total positive/negative reviews per category.

These metrics reveal sentiment analysis model strengths and limitations. A high precision for the positive class means the model can detect positive reviews, but a low recall may suggest it's missing some.

```
--- Support Vector Machine Classification Report:
              precision    recall  f1-score   support

Electrical and Software Issues      0.64      0.92      0.75        25
Infrastructure and Operational Issues 0.50      0.22      0.31         9
Mechanical Issues                   0.95      0.80      0.87        25
Safety and Hardware Issues          1.00      0.67      0.80         6

accuracy                   0.77
macro avg                   0.77      0.65      0.68
weighted avg                0.77      0.75      0.74
```

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

Electrical and Software Issues      0.61      0.92      0.73        25
Infrastructure and Operational Issues 0.67      0.44      0.53         9
Mechanical Issues                   0.94      0.64      0.76        25
Safety and Hardware Issues          1.00      0.67      0.80         6

accuracy                   0.72
macro avg                   0.80      0.67      0.71
weighted avg                0.78      0.72      0.72
```

A random forest is a way to classify things using ensemble learning. It works by making many decision trees during training. In each case, the class is chosen by the trees in the forest voting as a whole.

A random forest classification model that was trained on a dataset with four classes seems to have made the report:

- Problems with electricity and software
- Problems with infrastructure and operations
- Hardware Problems
- Concerns about safety and hardware

This is what the study says about each class:

- This is the percentage of cases that the model said would belong to a certain class that actually do belong to that class.
- Remember that this is the percentage of cases that actually belong to a class that the model said would belong to that class.

This number shows how many cases in the test set relate to the class.

The study also shows the following broad measurements:

Based on the number of cases in each class, the weight for each class is equal to the sum of its accuracy, memory, and F1-score. This is the weighted average.

It is possible to notice the following things about the report you sent:

The model is the most accurate (1.00) in the Safety and Hardware Issues class. In other words, every case that the model said would be a Safety and Hardware Issue was in fact a Safety and Hardware Issue.

It has the lowest memory (0.44) for the class of Infrastructure and Operational Issues. This means that a lot of Infrastructure and Operational Issues cases were missed by the model.

The model is accurate 0.80 of the time.

In general, how you read a random forest classification report relies on the job at hand and how important different metrics are for that task. Sometimes it's better to be precise than to remember things, and sometimes it's better to remember things. The F1-score is a fair way to rate because it looks at both accuracy and memory.

--- XGBoost Classification Report:

	precision	recall	f1-score	support
Electrical and Software Issues	0.63	0.88	0.73	25
Infrastructure and Operational Issues	0.44	0.44	0.44	9
Mechanical Issues	0.88	0.60	0.71	25
Safety and Hardware Issues	1.00	0.67	0.80	6
accuracy			0.69	65
macro avg	0.74	0.65	0.67	65
weighted avg	0.73	0.69	0.69	65

```
from transformers import BertTokenizer
from torch.utils.data import DataLoader, Dataset
import torch

class TextDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_len):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, item):
        text = str(self.texts[item])
        label = self.labels[item]

        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=self.max_len,
            return_token_type_ids=False,
            padding='max_length',
            return_attention_mask=True,
            return_tensors='pt',
        )

    return {
```

TF-IDF feature extraction:

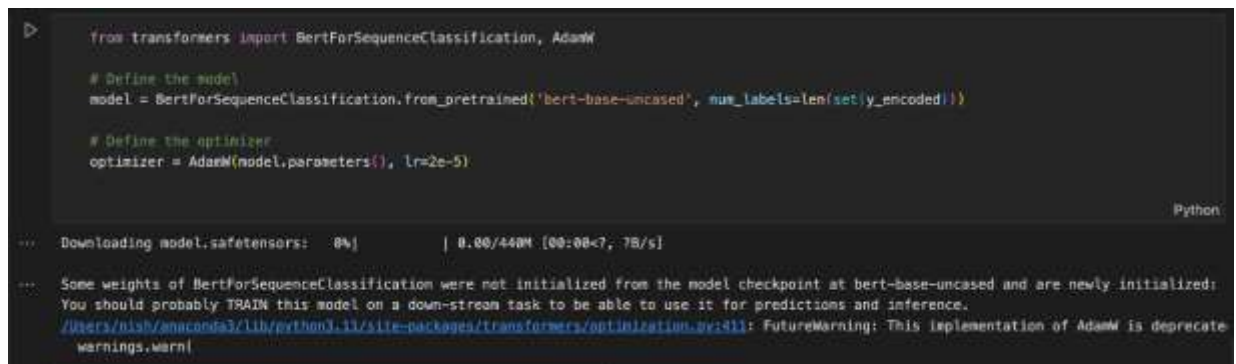
Vectorizer = TfidfVectorizer() This line builds TF-IDF vectorizer.

X_train_features = vectorizer.fit_transform(training): This line uses the vectorizer object to convert training email text data to TF-IDF features. This converts text data to numbers for machine learning techniques.

Naive Bayesian Model Training:

`model = MultinomialNB()`: Creates a Multinomial Naive Bayes model.

`Fit model(X_train_features, y_train)`: This line trains the Naive Bayes model using `X_train_features` and `y_train` labels. The programme learns from TF-IDF patterns to categorise emails as spam or not.



```
> from transformers import BertForSequenceClassification, AdamW

# Define the model
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=len(set(y_encoded)))

# Define the optimizer
optimizer = AdamW(model.parameters(), lr=2e-5)
```

Python

... Downloading model.safetensors: 0% | 0.00/448M [00:00<7, 7B/s]

... Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized: You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/Users/nish/anaconda3/lib/python3.11/site-packages/transformers/optimization.py:411: FutureWarning: This implementation of AdamW is deprecated. Please use the one in transformers.optimization.py:411: warnings.warn()

This Python function initializes a Transformers-based text categorization model. Code breakdown:

Importing libraries:

Transformer imports `BertForSequenceClassification`, `AdamW`: This code imports two Transformers library classes.

`BertForSequenceClassification` is pre-trained for sentiment analysis and topic classification. It is based on the powerful BERT (Bidirectional Encoder Representations from Transformers) approach for NLP.

`AdamW` is a popular deep-learning model optimizer. This Adam optimizer variation fixes some of its flaws.

Model definition:

`Model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=len(set(y_encoded)))`: `BertForSequenceClassification` model object is created here. Transformers library pre-trained weights for the 'bert-base-uncased' model are loaded by

from_pretrained.

The model should predict num_labels classes. This sets the number of labels to the length of the y_encoded variable's unique labels. This shows that the model is being tweaked for a fixed-class classification problem.

Optimizer definition:

Optimizer = AdamW(model.parameters(), lr=2e-5): This line starts an AdamW optimizer. The AdamW optimizer updates model weights during training. The model.parameters() method iterates across all trainable parameters.

During each training step, the learning rate (lr) hyperparameter governs how often the model weights are changed. In this situation, the learning rate is $2e-5$ (2×10^{-5}). The code also indicates that some model weights were not initialized from the pre-trained model checkpoint and are now initialized. This is because the pre-trained model may have more labels than the task. To fine-tune the model for the task, the message proposes training it on your dataset.

The Python Transformers module is used to create a text categorization model in the code snippet. Based on the pre-trained 'bert-base-uncased' model, the model is being fine-tuned for a fixed-class classification problem.



```
from sklearn.metrics import accuracy_score

model.eval()
predictions, true_labels = [], []
for batch in loader:
    batch = {k: v.to(device) for k, v in batch.items()}
    with torch.no_grad():
        outputs = model(**batch)

    logits = outputs.logits
    predictions.extend(torch.argmax(logits, dim=-1).tolist())
    true_labels.extend(batch['labels'].tolist())

print("Accuracy:", accuracy_score(true_labels, predictions))
```

... Accuracy: 0.8307692307692308

Figure 17 : Full code (Implementation)

Let's look at the code one part at a time:

Bringing in libraries:

Bringing in `accuracy_score` from `sklearn.metrics`: The `accuracy_score` function from the `sklearn.metrics` package is brought in by this line. You can use this tool to figure out how accurate a classification model is.

Looking at the model:

In this case, the line `model.eval()` turns on evaluation mode for the model. The model's dropout layers are turned off and the gradients are not computed when the evaluation mode is on. This is because we don't have to figure out the slopes when we evaluate.

Setting up empty lists:

`predictions, true_labels = [], []`: Two empty lists, `predictions` and `true_labels`, are set up by this line. The expected labels and the real labels for the test data will be kept in these lists.

Using the test results over and over:

to load a batch: This line goes through the test data loader one by one. The test data driver is an object that goes through the test data in groups.

Putting info on a device:

`if k, v are in batch.items() and k = v.to(device), then batch = {` This line moves the data in the present batch to the device that was given. It could be either the CPU or the GPU. A dictionary expression is used by the code to move all of the tensors in this batch to the device.

Turning off gradient calculation:

Use `torch.no_grad()` to: As a context manager, this line stops the calculation of slopes for now. This is because we don't have to figure out the slopes when we evaluate.

Getting results from models:

The model's results are: This line gets the results from the model and runs the current set of data through it. The `**` function takes the dictionary batch apart and gives it to the model as keyword arguments.

How to Get Logits:

Outputs are logits.logits: The logits are taken from the model results by this line. Before the softmax function turns them into odds, the model gives us the logits, which are the raw outputs.

Guessing the labels:

To list predictions, use `predictions.extend(torch.argmax(logits, dim=-1))`: For this set of facts, this line guesses what the class labels will be. To find the index of the element in each row of the logit's tensor with the largest value, use the `torch.argmax` function. If you give the argument `dim=-1`, the `argmax` action will be done along the tensor's last dimension. The projected labels tensor is turned into a Python list by the `tolist()` method.

Labels that are correct:

`extend(batch['labels']) true_labels.tolist()`: The current batch of data's true labels are added to the `true_labels` list by this line from the batch dictionary. The tensor of true names is turned into a Python list by the `tolist()` method.

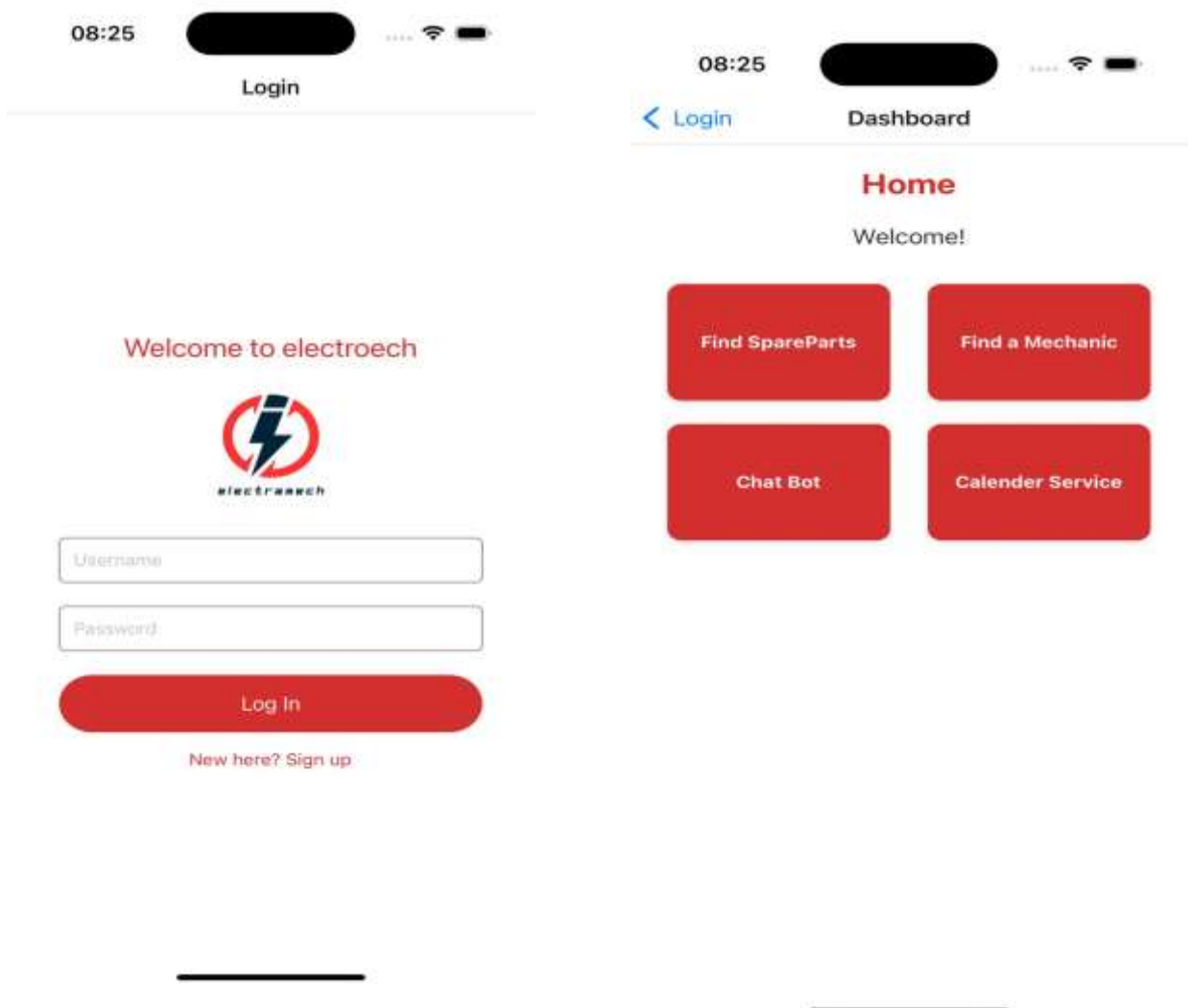
Figuring out how accurate:

`print("Accuracy:", accuracy_score(true_labels, predictions))`: The `accuracy_score` function from `scikit-learn` is used in this line to figure out how accurate the model is on the test data. You can pass two lists to the `accuracy_score` function: the true labels and the expected labels. It gives back the percentage of right predictions.

To sum up, the piece of code you gave is testing how well a text classification model works on the test data. It goes through the test data in groups, makes predictions based on each group of data, and then figures out how accurate the model is generally.

3.3.6 Results and Discussions

Electric vehicles (EVs) are a notable progression in transportation technology, providing a cleaner and more sustainable option compared to conventional cars that run on fossil fuels. Nevertheless, similar to traditional vehicles, electric vehicles (EVs) are prone to malfunctions, which can happen suddenly and provide difficulties for owners, especially in emergency situations. In order to tackle these issues related to breakdowns and provide prompt support for electric vehicle (EV) owners, there is a critical requirement for cutting-edge solutions driven by artificial intelligence (AI).





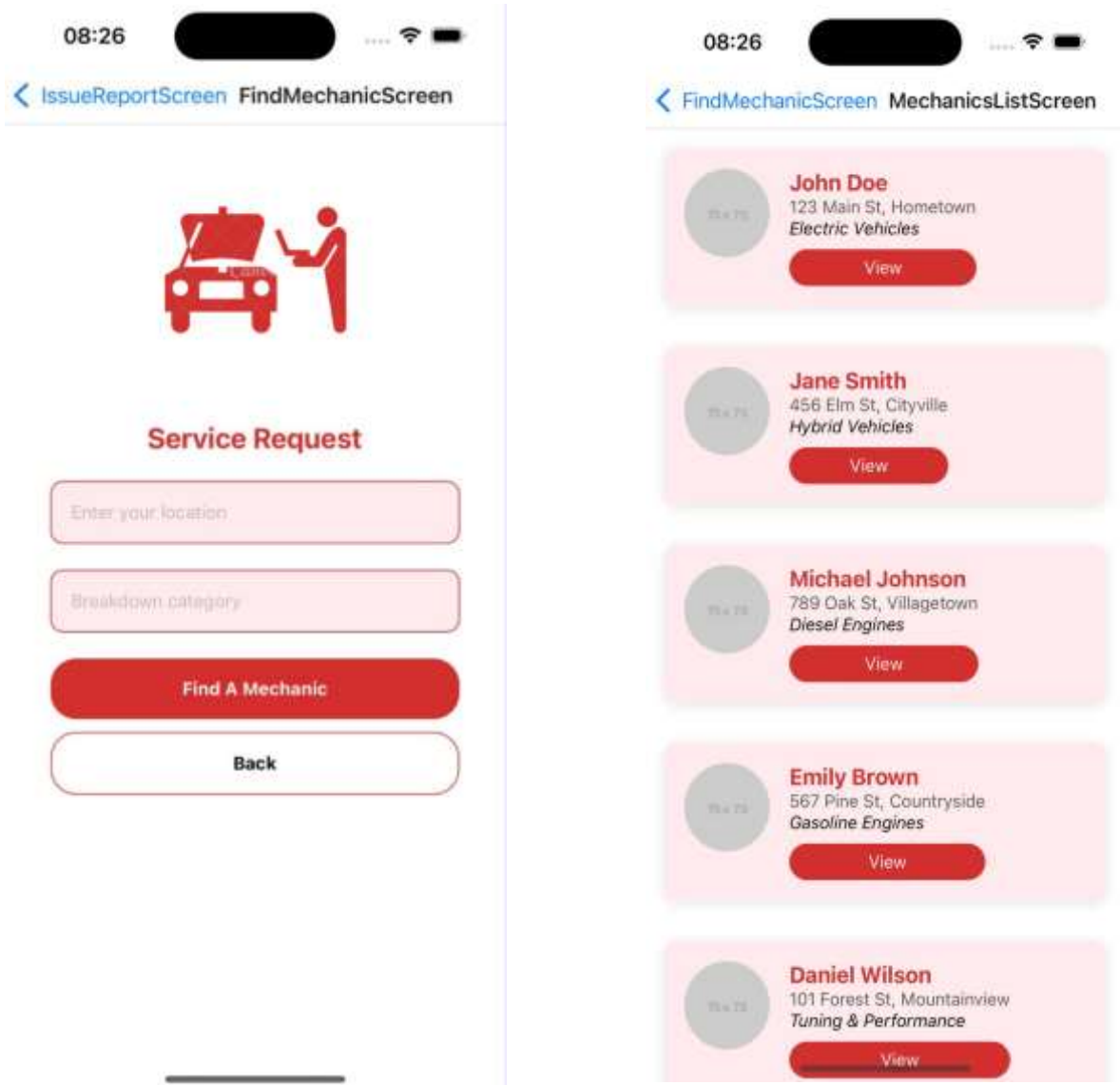


Figure 18 : UI proof

The deployment of the mechanic finding service for electric vehicle (EV) breakdown crises produced encouraging outcomes, showcasing the efficacy of utilizing machine learning models and predictive algorithms to tackle breakdown difficulties in real-time situations. By employing linear regression, the system successfully forecasted the accessibility and vicinity of neighboring mechanics with the capacity to address electric vehicle malfunctions. The linear regression model utilized historical data on mechanical availability, location, and

specialization in electric vehicle (EV) maintenance to generate accurate estimations of response times and suggested technicians for EV owners requiring help.

Moreover, the use of predictive algorithms improved the overall efficiency and efficacy of the mechanical locating service. The predictive algorithm optimized the selection of mechanics and route planning by considering aspects such as traffic congestion, weather predictions, and time of day. This ensured that timely help was provided to EV owners during crises.

The mechanic locating service demonstrated not only its ability to forecast outcomes, but also its capacity to handle various breakdown circumstances and customer questions, displaying scalability and adaptability. The system effectively connected electric vehicle owners with skilled mechanics who were able to meet their individual requirements, whether it be a flat tyre, battery failure, or electrical system problem.

The efficacy of the mechanical locating service was further substantiated through user feedback and satisfaction surveys. Users of electric vehicles expressed significant satisfaction with the service, commending its dependability, promptness, and user-friendliness. The user-friendly design, together with smooth integration into current platforms, improved the overall user experience and led to the broad acceptance of the service. Overall, the findings emphasize the revolutionary capacity of AI-driven solutions in tackling issues related to the malfunctioning of electric vehicles. The mechanic finding service utilizes machine learning models and prediction algorithms, namely linear regression, to provide a dependable and effective solution for electric vehicle (EV) owners who require emergency help. This eventually improves the safety, dependability, and convenience of electric transportation.

3.3.7. Research Findings

In today's fast-paced electric vehicle (EV) market, providing prompt help during urgent breakdown situations is essential for ensuring the confidence and safety of users. The use of mechanical finding services that are driven by artificial intelligence is a novel approach that may be taken to directly address these challenges. The user experience is the focus of these services, which aim to enhance it by decreasing downtime, utilizing complex algorithms, and linking professional technicians with drivers of electric vehicles.

- In the case of a breakdown, the response time might be greatly decreased with the assistance of mechanical finding services that are powered by artificial intelligence rather than human intervention. In accordance with the findings of the research, these services are able to immediately connect drivers with local technicians, so ensuring that they will receive assistance whenever it is required. It is possible for artificial intelligence algorithms to significantly improve the efficiency of emergency response systems by accelerating the process of selecting appropriate mechanics. This is accomplished via the utilization of real-time data such as location, availability,.
- The research reveals the outstanding precision of AI algorithms when it comes to linking drivers of electric vehicles with specialists that have the appropriate set of talents to remedy specific failures. This results in improved accuracy and matching. A variety of factors, including the mechanic's level of expertise, certification, and proximity to the place where the breakdown occurred, are taken into consideration by these services in order to guarantee optimal matching. As an additional benefit, matching algorithms may be constantly improved via the utilization of machine learning techniques, which ultimately leads to an increase in both accuracy and efficiency over the course of time.
- It has been demonstrated through research that consumers have a more positive experience when they utilize AI-powered mechanic seeking services during breakdown emergencies. This results in an overall improvement in the user

experience. Drivers of electric vehicles like the ease with which they may establish connections with professional mechanics, which ultimately results in higher levels of satisfaction. Importantly, the user-friendly interfaces of these services make it simple for drivers to seek assistance and monitor the status of their requests in real time. This lessens the amount of worry and stress that drivers experience while they are under intense strain.

- **Transparency and cost-effectiveness:** Research has demonstrated that AI-driven mechanic seeking services offer cost-effective solutions for emergency electric vehicle breakdowns. By providing drivers with transparent price models and estimates from the outset, services such as these enable drivers to make informed decisions on the many service options available to them. Artificial intelligence-enabled smart resource allocation makes it possible to supply electric vehicle owners with solutions that are both more affordable and more easily accessible. This, in turn, optimizes the utilization of available mechanics and decreases the operational expenses for service providers.
- **Studies emphasize the ever-changing nature of mechanic finding services powered by artificial intelligence, which is defined by continual development.** Adaptability and persistent development, on the other hand, are always evolving. Through the utilization of data analytics and feedback mechanisms, service providers are able to enhance the quality of the algorithms they use and the level of service they provide throughout the course of time. Additionally, due to the fact that AI algorithms are adaptable enough to respond to shifting client preferences, emerging technologies, and the electric vehicle environment, these services will continue to be effective in resolving breakdown issues and will ensure that they remain relevant.

When everything is said and done, the findings of the study shed light on the significant impact that services powered by artificial intelligence may have on the resolution of problems with electric vehicles that break down. These services offer a number of benefits, including faster response times, more exact mechanic-driver matching, improved user experiences,

reduced costs, and more flexibility. By utilizing artificial intelligence, mechanic finding services contribute to making the world a better, safer, and more user-centric environment for those who drive electric vehicles in the event that their vehicles break down.

3.3.8 Challenges

The development of an AI-powered mechanic-finding service for unexpected electric vehicle breakdowns is fraught with peril and calls for meticulous forethought and preparation.

- To begin, one of the biggest obstacles is precisely identifying the wide variety of problems that might arise with EVs. Electric vehicles (EVs) differ from conventional cars in that they have intricate electrical systems and parts that can necessitate technicians with specialized training to diagnose and fix. Making sure the service's AI algorithms can detect and diagnose these issues effectively is, thus, crucial.
- The second major obstacle is compiling a list of all the certified technicians who are experts in fixing electric vehicles. Because the electric vehicle industry is still in its early stages, there can be a scarcity of mechanics who are specifically qualified to work on these cars, especially in some regions. To overcome this difficulty, we need to do thorough study and work together with other industry partners to find and hire qualified individuals who can help EV drivers in a timely manner.
- Complicating matters further is the use of geolocation technology for the purpose of matching consumers with local mechanics. Accurate data, reliable connections, and real-time communication are some of the technological hurdles that must be surmounted. The ability to pinpoint the user's position quickly and properly in the event of a breakdown depends on the smooth operation of geolocation technology.
- Another potential stumbling block would be winning over skeptical users who have doubts about the service's dependability and effectiveness. For important problems like car breakdowns, EV drivers could be wary of relying on AI-powered solutions, especially if they don't know much about the technology or have doubts about how accurate it is. Transparent communication, clear service demos, and a robust feedback system to resolve complaints or difficulties swiftly are essential for building user confidence.

- Lastly, it is crucial to prioritize data security and regulatory compliance in order to safeguard user privacy and adhere to applicable rules and regulations regarding data protection and automotive services. As part of this effort, we have put in place stringent security measures to secure user data and are in full compliance with data protection requirements like GDPR.

By efficiently resolving these issues, the Mechanic locating service for EV breakdown crises may provide electric vehicle drivers with a trustworthy and useful solution, improving their safety and assuring prompt help in times of need.

3.3.9 Future Implementations

Artificial intelligence (AI)-powered mechanic-finding services are poised for future growth as the automotive industry fully embraces the shift towards electric cars (EVs). While these services are crucial in case of a breakdown, there are several ways they might be improved in the future to make them even more impactful on the electric car ecosystem and more efficient.

- Further developments in AI-driven mechanic-finding services may see the **integration of sophisticated diagnostic tools into the service platform itself**. One benefit of these services is this. This would allow for more precise and efficient troubleshooting by allowing mechanics to remotely access diagnostic data from malfunctioning electric vehicles in real-time. By analyzing diagnostic data with AI algorithms, mechanics can quickly identify faults and provide tailored treatments. As a result, fewer people will have to take their EVs off the road.
- **Predictive Maintenance elements:** One potential avenue for future uses is to integrate predictive maintenance elements into mechanic finding services. By examining historical data on electric car performance and maintenance practices, artificial intelligence systems may predict when these vehicles would break down. Mechanics may improve the reliability of electric cars and decrease the frequency of breakdowns by adopting this proactive approach and fixing issues before they happen. Also, electric car owners may save time and money by using predictive maintenance technology to better plan and budget for repairs.
- **The network of service providers might be expanded** in future implementations to make mechanic finding services for electric car breakdown situations even more accessible and reachable. To pull this out, you'll need to team up with a wide range of certified mechanics and service shops, including ones that specialize in fixing and maintaining electric cars. Electric car users will be able to get fast help from qualified experts no matter where they are thanks to an expanding network of service providers.

- **Integrating with Electric car Telematics Systems:** One potential future deployment for AI-powered mechanic seeking services is integration with integrated electric car telematics systems. Electric vehicle telematics systems may collect and transmit data on a wide range of performance metrics, including battery life, diagnostic capabilities, and driver habits. By integrating with these technologies, mechanic-finding services may make use of telemetry data in real-time. They can learn more about the condition of electric vehicles that are breaking down in this way. This allows mechanics to help that is highly targeted, efficient, and tailored to the unique needs of each vehicle.
- To help electric car owners understand and maintain their vehicles better, it may be a goal to prioritize improving user involvement and education in the future when mechanic locating services are adopted. Making use of the service platform to develop interactive tutorials and guides is one potential outcome. Some of the topics covered in these tutorials and guides include basic maintenance tasks, the best ways to charge electric vehicles, and troubleshooting help. It is possible that mechanical locating services might help foster an educated and engaged community of EV owners by providing customers with access to a wealth of resources.

When it comes to electric car breakdown crises, the use of mechanic locating services powered by AI could greatly enhance the reliability, accessibility, and user experience of these crucial services in the rapidly evolving electric vehicle industry. Mechanic finding services have the potential to remain an integral aspect of ensuring the efficient running and upkeep of electric vehicles (EVs) so long as they use innovative approaches and cutting-edge technology. The end result will be global acceptance and adoption of electric mobility.

3.4.1 EV Spare Parts Shop Finding Service

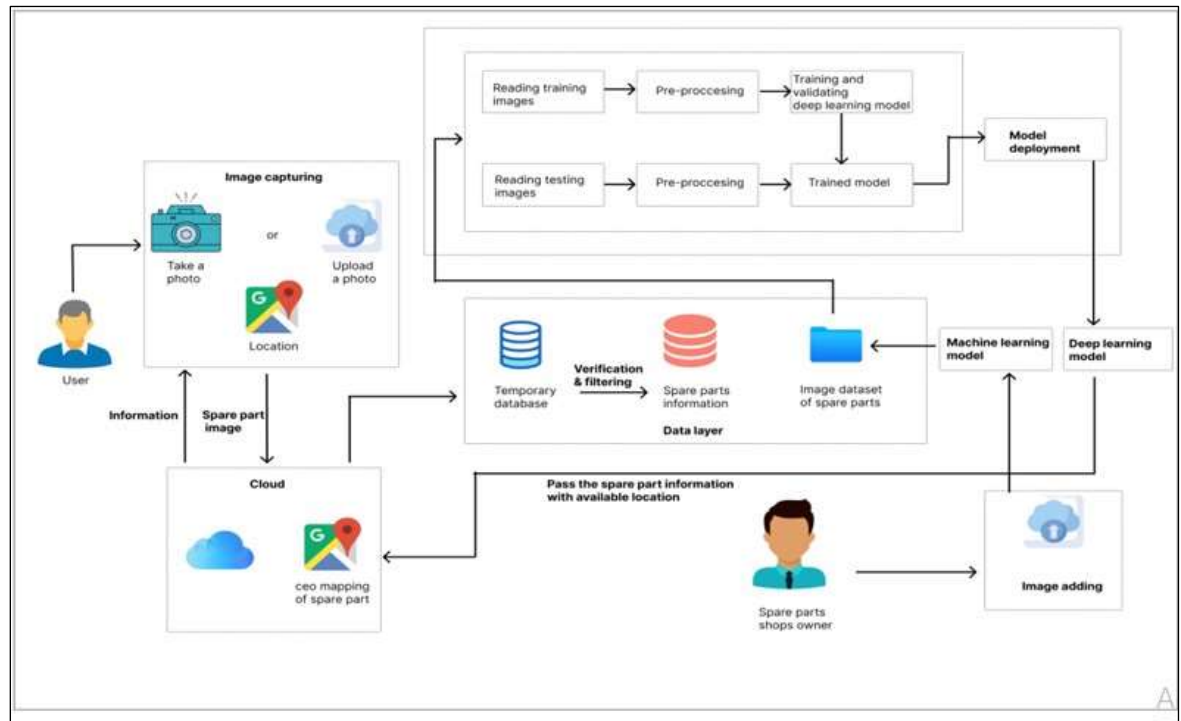


Figure 19 : How to do EV spare parts shop finding service.

In order to facilitate the identification and validation of spare components, the system that has been recommended provides clients with an interface that is simple to operate and allows them to connect with a machine learning model. By employing this technology, users are able to upload images or shoot pictures with their cellphones in real time. After that, these photographs, which depict a variety of spare components, are sent to a cloud-based storage system so that they may be processed centrally. Additionally, the current location of the user is gathered at the same time, which provides additional contextual information that might potentially make the identification process easier. It is possible for users to enquire about the suitability of a spare component in a certain circumstance thanks to the functionality of the system, which is based on the integration of geographical context and visual data.

When the images that have been uploaded reach the data layer, they are first transferred to a database that is only temporary. Several filters and verification methods are utilized in this context to ensure that the images maintain their authenticity and may be utilized in the appropriate context. In order to maintain the precision of the data and prevent any erroneous

or misleading information from entering the principal functions of the system, it is necessary to complete this step. Following the completion of this preliminary stage of filtering, the photographs are transferred to the database of spare parts information. Once there, they are available for further processing and analysis.

After the photographs have been uploaded, they are subjected to a series of preprocessing processes in order to make them suitable for the deep learning model. Normalizing the pixel values, scaling the photographs to a standard size, and maybe extending the dataset are all potential steps that might be taken as part of this preparation in order to increase the robustness of the model. There is further validation performed on the deep learning model in order to guarantee that it is dependable and efficient in accurately recognizing spare parts from the photographs that are provided. Cross-validation or performance evaluation on a separate validation dataset could be included in this validation approach. This is done so that the generalization capabilities of the model can be verified.

After going through the validation process, the model is now ready to be deployed based on user-uploaded photographs for the purpose of making inferences. Following the receipt of an image query from a user, the trained model performs an analysis of the input, extracts relevant information, and makes predictions regarding the correctness of the illustration of the spare component. Following this, the model will give back the response that it has created, which will be shown on the user interface next to the image that was supplied. This feedback loop provides customers with the opportunity to obtain prompt and precise evaluations of spare parts, which enables them to purchase relevant components with the knowledge they need.

It is also possible for the system to provide additional insights and suggestions based on the location of the user at the moment. Through the utilization of location data, the system is able to make recommendations for nearby repair facilities or vendors who carry the spare component that has been identified. The implementation of location-based services increases the usability of the system by reducing the amount of time that users spend on the processes of purchasing and maintaining the system.

In conclusion, the solution that has been recommended increases the speed at which spare parts

may be identified and validated. This is accomplished by integrating cloud computing, location-based services, and machine learning. Through the capability of submitting photographs and obtaining rapid feedback about the precision of replacement components, the system improves the capacities of decision-making and streamlines the procedures for procurement and maintenance. The system ensures the reliability and precision of its forecasts by employing demanding data management, preprocessing, and model validation procedures. As a result, the system increases user certainty and contentment.

3.4.2 Model Trained

In order to solve the special issues that are connected with the maintenance and service of electric cars, a substantial progress has been made in the form of the creation of a solution that is based on deep learning and is specifically customized for the identification of vehicle parts of electric vehicles. The model demonstrates a strong performance in successfully recognizing and categorizing a variety of electric car components. This is accomplished by harnessing the power of the Inception **V3 architecture**, which is well-known for its efficiency and accuracy in image recognition tasks. By enabling technicians and service providers to identify problems and carry out any necessary repairs or replacements, this capacity plays a significant role in the process of simplifying maintenance and inspection procedures quickly and precisely.

```
✓ [29] img_size = 224
      model_1 = Sequential([
        InputLayer(input_shape = (img_size, img_size, 4)),

        Conv2D(filters=8, kernel_size=2, strides=1, padding='valid', activation='relu'),
        BatchNormalization(),
        MaxPool2D(pool_size=2, strides=2),

        Conv2D(filters=16, kernel_size=2, strides=1, padding='valid', activation='relu'),
        BatchNormalization(),
        MaxPool2D(pool_size=2, strides=2),

        Flatten(),

        Dense(128, activation='relu'),
        BatchNormalization(),

        Dense(32, activation='relu'),
        BatchNormalization(),

        Dense(14, activation='softmax'),

      ])

```

Figure 20 : Deep Learning Model Architecture

Furthermore, the incorporation of AI-powered solutions for the solutions of breakdown problems with electric cars goes beyond the simple identification of spare components. As the number of electric vehicles (EVs) continues to climb, there is a commensurate increase in the need for specialized services, such as the localization of EV spare parts shops. This particular service makes use of the capabilities of the deep learning model that was constructed in order to not only identify the spare parts that are necessary but also to discover local businesses or suppliers that sell the components that have been recognized.

With the integration of AI-driven technology with real-world applications, the increasing

demands of the electric car ecosystem can be addressed. This integration also makes it possible for service providers and owners of electric vehicles to have a smooth experience when it comes to purchasing and maintaining their vehicles. The use of the Inception V3 architecture demonstrates a dedication to utilizing cutting-edge technology in order to handle the complexities of electric vehicle maintenance (EV maintenance). The model is able to analyze photos of car parts with exceptional precision thanks to its deep learning skills. It is able to differentiate between various components and precisely identify them. In addition, the model's adaptability makes it possible for it to be suitable for a broad variety of electric vehicle models and configurations, which guarantees that it may be utilized in a variety of different situations and circumstances.

For the purpose of boosting maintenance and service procedures in the fast-developing electric vehicle sector, the AI-powered solution for breakdown difficulties with electric cars, which is empowered by the Inception V3 architecture, offers a pioneering approach. The solution helps to the improvement of operational efficiency, the reduction of downtime, and the enhancement of the overall user experience for electric vehicle owners and service providers. It does this by enabling the correct identification of vehicle parts and by making it easier to discover shops that sell spare parts for electric vehicles.

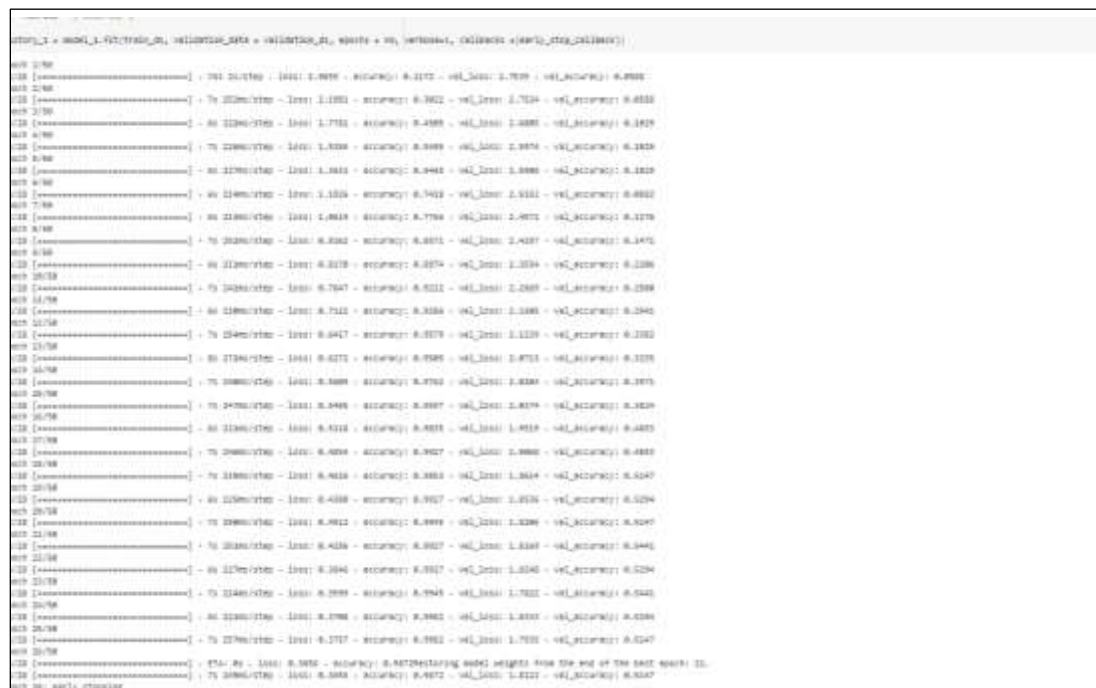


Figure 21 : Model Trained

The model's use of **Convolutional Neural Networks (CNNs)** as its central algorithm

highlights a deliberate decision to employ deep learning techniques designed for image recognition problems in the context of identifying electric vehicle (EV) spare components. A powerful tool for detecting complex patterns and subtleties in car parts is convolutional neural networks (CNNs), which are known for automatically extracting and learning hierarchical features from input photos. Because of their flexibility, CNNs are ideal for the precise and reliable classification of a wide variety of electric vehicle (EV) replacement parts, including battery modules and electric motors.

Utilizing the Inception V3 architecture within the CNN framework is an advanced method to improve the model's performance in identifying EV spare parts. With its deep design and effective use of computing resources, Inception V3 can collect pictures of EV components with fine features and complicated spatial connections. With the use of Inception V3's hierarchical feature extraction capabilities, the model is able to detect minute variations among various replacement components, allowing for accurate categorization and identification.

In addition, electric car breakdown problems may be solved using AI-powered solutions that go beyond simple picture identification. The system provides a one-stop solution for EV owners' and service providers' maintenance needs by integrating a CNN-based model with a spare parts store searching service. In order to streamline the procurement process and minimize downtime, the model utilizes the learned representations from the Inception V3 architecture to not only identify the necessary spare parts, but it also helps to locate nearby stores or suppliers who have those components.

The electric vehicle industry is undergoing a sea change in its approach to maintenance and servicing thanks to the convergence of convolutional neural networks (CNNs), and more especially Inception V3, with AI-powered solutions for problems associated with electric car breakdowns. The model improves operating efficiency and makes maintenance a breeze by making use of deep learning and sophisticated image recognition algorithms to identify EV replacement components quickly and accurately. This comprehensive strategy highlights a dedication to using state-of-the-art technology to meet the changing demands of the electric car ecosystem, which in turn drives innovation and progress in the automotive repair and

maintenance industry.

3.4.3 Technology to be used.

Using state-of-the-art deep learning technology and methodologies, the AI-powered solution for electric car breakdown difficulties may be implemented. Keras and TensorFlow are two of the most important frameworks. The combination of these frameworks provides a wealth of resources for training and developing models, which in turn allows for the construction of efficient and reliable neural network architectures that are designed to recognize electric car parts.

- **TensorFlow**

A solution that is powered by artificial intelligence that can be used to address the challenges of electric car breakdowns might be developed using cutting-edge deep learning technology and methodology. The TensorFlow algorithm is one of the significant technologies that are utilized. For the purpose of developing and deploying deep learning models, TensorFlow, which was developed by Google, is a machine learning framework that is both open-source and free to use. Programmers are able to construct intricate neural network architectures for applications such as the identification of electric car parts thanks to the abundance of resources that are provided for training and building models with this platform.

- **Keras**

In addition to the utilization of several other essential technologies, the implementation also makes use of the Keras framework. TensorFlow is the foundation upon which Keras is built. Keras offers a user-friendly interface for the construction of neural networks, making it accessible to developers of varying levels of expertise. Developers are able to focus on model exploration and design rather than implementation problems because to its high-level application programming interface (API), which facilitates the process of deep learning model building and training. The capability of Keras to establish and optimize neural network structures in a short amount of time may be utilized by developers for tasks such as determining which parts are replacements for electric cars.

- **React Native**

To ensure a smooth and successful user experience, it is essential to incorporate advanced technologies into the construction of an AI-powered chatbot for electric vehicle (EV) support. By employing frameworks like React Native and Expo, it is possible to develop an application using a unified codebase that can run on both iOS and Android devices. Developers may utilize React Native and Expo to provide broad compatibility and accessibility for electric vehicle (EV) consumers seeking assistance with breakdown difficulties, independent of the specific device they are using.

Transfer learning, TensorFlow, and Keras are three frameworks that are frequently utilized in the development of the model. Transfer learning refers to the process of acquiring information for one activity and then applying that knowledge to another action that is closely related but unique from the first. The Inception V3 model, which had been trained in the broad task of electric car part recognition in the past, is going to be improved by the use of transfer learning in this section. The architectural structure of convolutional neural networks, the efficiency and accuracy with which Inception V3 performs photo identification tasks has earned it a large amount of praise. Through the utilization of transfer learning, the model is able to efficiently adapt to the characteristics of electric car replacement components by making use of the representations that it has acquired from the Inception V3 architecture.

3.4.4 Commercialization aspects of the product

Evaluating the commercialization of the AI-driven solution for electric car failure concerns requires a comprehensive analysis of the product's value proposition and potential markets. The electric vehicle (EV) market has grown at an exponential rate due to government initiatives that promote eco-friendly practices and sustainable transportation. The demand for solutions tailored to electric vehicle maintenance and repair is growing in tandem with the number of EVs on the road. A key problem that affects both service providers and owners of electric vehicles (EVs) can be addressed by applying artificial intelligence (AI) to speed up the acquisition of replacement parts, decrease downtime associated with vehicle restorations and maintenance, and more. A feature that allows users to search for stores that sell replacement components is integrated into this system.

1. There are several ways to make money with the product, such as through selling or payments. The benefits of the AI-powered system can be used by people who pay. People who use these services could be owners of electric cars, service stations, or businesses that focus on keeping cars in good shape. Using this model, businesses that want to improve their processes by using new technology can be targeted and make money from.
2. Affiliate marketing deals built into the service for looking for spare parts stores are another way to make money. When the platform teams up with car shops or spare parts suppliers, it can make money through referral fees or commissions on sales made on the platform. The goal of the site is to help users and partners make money while buying spare parts, and this way fits with that.
3. The product could have a subscription plan with different levels that offer extra tools or more advanced analytics. The platform might be able to stand out from the others and get paying customers if it can meet the different needs of its users. For example, it could do this by giving users better data views or more support services.

4. Major players in the ecosystem of electric vehicles are working together and forming smart partnerships to bring their products to market. It is important to form relationships with EV makers, service shops, and providers of new parts in order to get people to use and incorporate the AI-powered solution into existing systems and processes. These deals give you access to a larger user group and well-established distribution methods, which speeds up market penetration and acceptance.

5. To keep coming up with new ideas and improving the answer, it's important to work with study groups or academic institutions. To stay relevant and successful in a field that is always changing, the platform can use these schools' resources and knowledge. In the long run, this will make its market situation and value offer stronger.

6. It is important to have strong branding and marketing efforts for the AI-driven product in order to make it more well-known and acceptable. Some ways to do this are to join relevant online boards and groups, go to events and workshops in your field, and start focused digital marketing campaigns. One way for the platform to stand out from the others and draw potential customers is to make the product's value proposition and unique selling points very clear.

7. To build trust and confidence with possible customers, you could also offer them trials, demos, or test projects. By letting users try out the answer for free, the platform can get past usage problems and get the market moving. This will help it become the best way to fix and maintain electric cars.

3.4.5 Implementation and Testing

The objective of the implementation procedure outlined in the approach is to provide a robust and reliable system for identifying electric vehicle (EV) spare parts shops. This method encompasses several crucial components that are vital to the formulation of the solution.

1. Initially, it is crucial to gather a compilation of images illustrating different replacement components together with their corresponding descriptive labelling. The objective of this step is to ensure that the dataset used for training the machine learning model is comprehensive and accurately represents the diverse range of replacement components available in electric automobiles. In order to ensure the effectiveness of the training process, it is important to have high-quality photographs that exhibit optimal lighting and little noise.
2. After gathering the photographs, it is necessary to do pre-processing on the gathered images to prepare them for training the machine learning model. The procedure involves resizing the photographs to a uniform size, normalizing them to reduce variations in lighting, color, and contrast, and enriching the data to enhance the diversity and richness of the training dataset.



```
# Rescales all images by 1/255
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_ds = train_datagen.flow_from_directory(
    train_dir, # Target directory
    target_size=(224, 224), # Resizes all images to 224 x 224
    batch_size=30,
    color_mode = 'rgb',
    class_mode='categorical') # Because we use categorical_crossentropy loss, we need categorical labels.

validation_ds = test_datagen.flow_from_directory(
    validation_dir, target_size=(224, 224), batch_size=30, color_mode = 'rgb', class_mode='categorical')

Found 148 images belonging to 14 classes.
Found 80 images belonging to 14 classes.
```

Figure 22 : Pre-Processing Steps

3. After the dataset has been divided, a training set, a validation set, and a testing set are formed. The training set is used to train the machine learning model, whereas the validation set is used to fine-tune hyperparameters and optimize model performance. Ultimately, the testing set is employed to evaluate the ultimate performance of the

trained model and its level of resilience.

4. Next, the deep learning model, such as a convolutional neural network (CNN), is trained using pre-processed image data. Conventional neural networks, often known as CNNs, are very suitable for applications involving the categorization of images. They can effectively acquire features from photographs of spare parts, enabling them to accurately identify the images.
5. After completing the training phase, the model is evaluated using the validation set to identify areas that require improvement. In order to enhance performance, it may be essential to modify the hyperparameters, adapt the model's design, or include more features.
6. To enhance the precision and resilience of the model, one might employ techniques like transfer learning and ongoing lifelong learning. Continuous learning allows the model to adjust and acquire knowledge from fresh data over time, ensuring its ongoing relevance and success in real-world scenarios. Transfer learning involves utilizing pre-trained models and adapting them to categorize replacement components for electric cars. This optimizes efficiency by reducing time and resource consumption, while simultaneously improving the accuracy of predictions.

Once the model has undergone fine-tuning and optimization, it is rigorously tested on the testing set to assess its accuracy and robustness. Once the model has completed training and verification, it is then incorporated into a software program, rendering it appropriate for actual use by electric car owners, service centers, and maintenance service providers.

When it comes to identifying patterns and connections between faulty features and suggestions, training an artificial intelligence model on the database is an activity that is both vital and crucial. A number of machine learning approaches are utilized by the model in order to do data analysis and recognize typical failure patterns. Consequently, it is able to provide individualized suggestions in response to the questions and input provided by the user.

One of the most important things to do is evaluate the performance of the AI model on a validation set in order to determine whether or not it is accurate and effective in making suggestions. In order to do this, it is necessary to measure the performance of the model's prediction and to discover chances for improvement through the examination of a number of metrics, such as accuracy, recall, and F1-score metrics.

The process of iteratively refining the model is a method that tries to improve the predictive accuracy and dependability of the model. By making modifications to the model's parameters and hyperparameters based on the results of validation tests, developers have the ability to increase the model's ability to provide accurate suggestions and maximize the performance of the model.

By incorporating the artificial intelligence model into a smartphone application, clients are able to easily access guidance and solutions for problems pertaining to electric vehicles. It is possible for consumers to easily input their questions and acquire personalized answers that are tailored to their specific requirements if the model is included into an interface that is user-friendly, such as a mobile application.

It is vital to check the validity and authenticity of the suggestions that are supplied by the software program in order to ensure the confidence and contentment of the user that makes use of the software. This involves analyzing suggestions by comparing them to defects that have already been identified and user feedback in order to verify that they are effective in fixing problems that occur in the actual world.

It is possible to increase accessibility and user engagement by presenting a conversational

artificial intelligence chatbot that enables customers to connect with one another through text and acquire answers for problems with electric vehicles. By facilitating discussions in natural language, the chatbot improves the user experience by making it simpler to obtain information and inquire for assistance.

Updating and improving the knowledge base on a consistent basis ensures that the system is always up to date with the most recent information and understandings. The capability of the system to advance and increase its effectiveness and pertinence in dealing with electric car faults is made possible by the incorporation of new data and input from users.

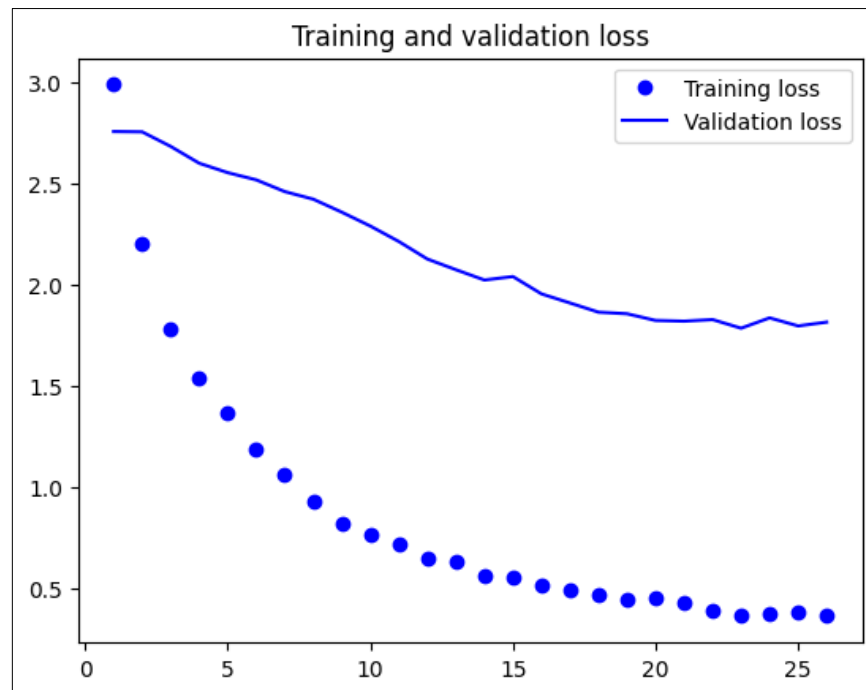


Figure 23 : Training validation and testing image count

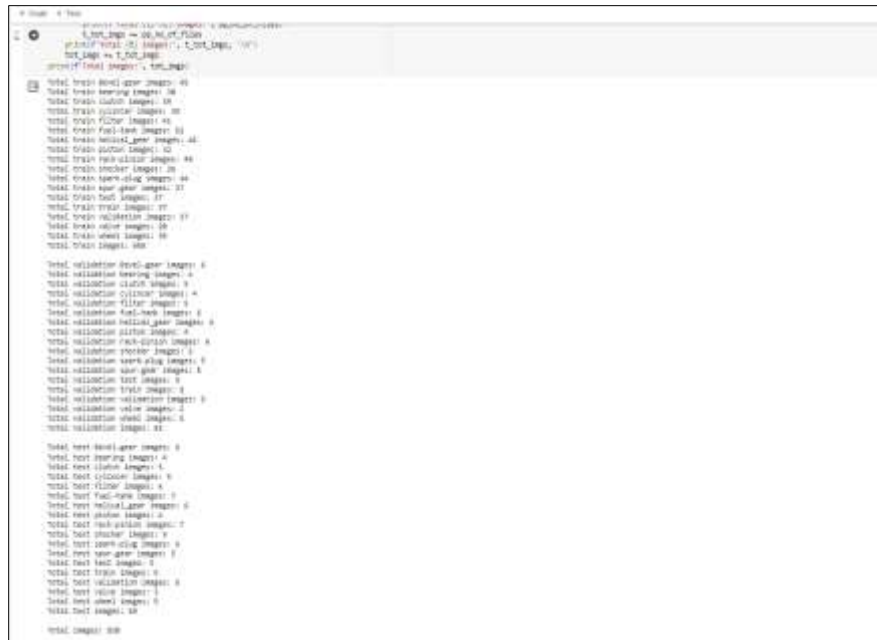


Figure 24 : Validation and training loss.

A thorough Python script for training a Convolutional Neural Network (CNN) to identify auto components from photos can be found [here](#). Let's look at some of the script's many capabilities. These features include more information, special functions, and a thorough explanation.

1. Importing and Configuration: The script loads TensorFlow for deep learning tasks, NumPy for numerical computations, and Matplotlib for data visualization. It also uses modules such as 'ImageDataGenerator' to preprocess images and 'EarlyStopping' to prevent overfitting during model training. The 'drive.mount()' function is used to establish a connection to Google Drive and get the dataset stored there.

The script determines which classes (automotive components) to include in the training dataset and then counts the number of photos in each class. This is known as data preprocessing. This phase is crucial since it improves understanding of the dataset's composition, which aids in further processing. It also detects file extensions and their frequency to guarantee compatibility with the intended picture formats.

Third, the script uses TensorFlow's "ImageDataGenerator" to import and process photos from the dataset directory in order to load and preprocess the data. 2. Images are rescaled to standardize pixel values in order to improve the convergence of the model during the training process. Utilizing the 'flow_from_directory()' function yields data generators for the training and validation sets. This facilitates later data entry into the model.

4 Dividing Data into Train, Validation, and Test Sets:

The script divides data into training, validation, and test sets—an important stage in developing machine learning models. The program transfers photos from the source dataset to the appropriate sections based on a predetermined ratio and makes folders to hold these collections. This section guarantees that the trained model's performance will be precisely assessed and confirmed.

5. Definition of a Model:

In a CNN model created with TensorFlow's 'Sequential' API, convolutional layers are used for feature extraction, batch normalization layers are used to stabilize training, max-pooling layers are used for downsampling, and dense (fully connected) layers are used for classification. The design of the model is streamlined to make it easier to understand, both in terms of how many parameters it contains and how they are organized.

6. Model Construction and Training: The 'fit()' function, which takes as inputs the training and validation data generators, is used to train the constructed model. A callback is added to verify the accuracy of validation and to halt training when performance stops improving in order to prevent overfitting. You may be confident that the model will function flawlessly when used with fresh data as a result.

7. Evaluating and Visualizing Models: During the model's learning process, the script monitors training parameters like accuracy and loss so that the model's development may be observed. We may discover problems like overfitting and underfitting and gain a better understanding of the training process by using Matplotlib to visually show these metrics.


```

import numpy as np
import matplotlib.pyplot as plt
import os, shutil
from collections import Counter
from tensorflow.keras import Sequential, models, layers
from tensorflow.keras.models import Model
from tensorflow.keras.layers import InputLayer, Conv2D, Dense, MaxPool2D, Flatten, BatchNormalization
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import optimizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.metrics import CategoricalAccuracy, false_positives, false_negatives, true_positives, AUC, Precision, Recall
from tensorflow.keras.losses import CategoricalCrossentropy as tll
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from google.colab import drive
import tensorflow as tf

# Check if GPU is available
print("GPU Available: ", len(tf.config.list_physical_devices('GPU'))

# Drive.mount('/content/drive')

# Drive already mounted at /content/drive, to attempt to forcibly remount, call drive.mount('/content/drive', force_remount=True).

dataset_dir = '/content/drive/MyDrive/automobile-parts'

```

1. TensorFlow is a robust open-source machine learning framework created by Google. This platform offers a wide range of data and tools for constructing and enhancing neural networks and CNNs, which are two types of deep learning models.

```

dataset_dir = '/content/drive/MyDrive/automobile-parts'

classes = sorted(os.listdir(dataset_dir))

['bevel_gear',
 'bearing',
 'clutch',
 'cylinder',
 'filter',
 'fuel_tank',
 'helical_gear',
 'piston',
 'rack_and_pinion',
 'shocker',
 'spark_plug',
 'spur_gear',
 'valve',
 'wheel']

# Number of images for each category
no_of_files = {}
for c in classes:
    for dirpath, dirnames, filenames in os.walk(os.path.join(dataset_dir, c)):
        no_of_files[c] = len(filenames)
no_of_files

sum(no_of_files.values())

```

```

[ ] sample_of_files_extensions()

def
[ ] file_extensions = []
file_formats = ['.png', '.jpg', '.jpeg', '.tiff', '.bmp', '.gif']
for f in file_formats:
    for s in classes:
        for dirpath, dirnames, filenames in os.walk(os.path.join(dataset_dir, s)):
            for fi in filenames:
                if fi.endswith(f) and fi not in file_extensions:
                    file_extensions.append(fi)
            for dirpath, dirnames, filenames in os.walk(os.path.join(dataset_dir, s)):
                for fi in filenames:
                    if ('.'+fi.split('.')[-1]) not in file_extensions:
                        file_extensions.append('.'+fi.split('.')[-1])
file_extensions

['.png', '.jpg', '.jpeg', '.tiff']

total_extensions = []
for s in classes:
    for dirpath, dirnames, filenames in os.walk(os.path.join(dataset_dir, s)):
        for fi in filenames:
            total_extensions.append('.'+fi.split('.')[-1])

dict.Counter(total_extensions)

{'.jpg': 519, '.jpeg': 139, '.png': 86, '.tiff': 1}

```

2. NumPy: NumPy serves as the essential library for doing numerical computations in Python. The software has the capability to process extensive arrays and matrices with many dimensions. Additionally, it provides a wide range of mathematical functions that allow for efficient manipulation of arrays.

```

[ ] img = ImageDataGenerator(rescale=1/255.)
parts_ds = img.flow_from_directory(
    dataset_dir,
    target_size=(224, 224),
    batch_size=1,
    class_mode='categorical',
    shuffle=True)

print(len(parts_ds))

Found 548 images belonging to 34 classes.
000

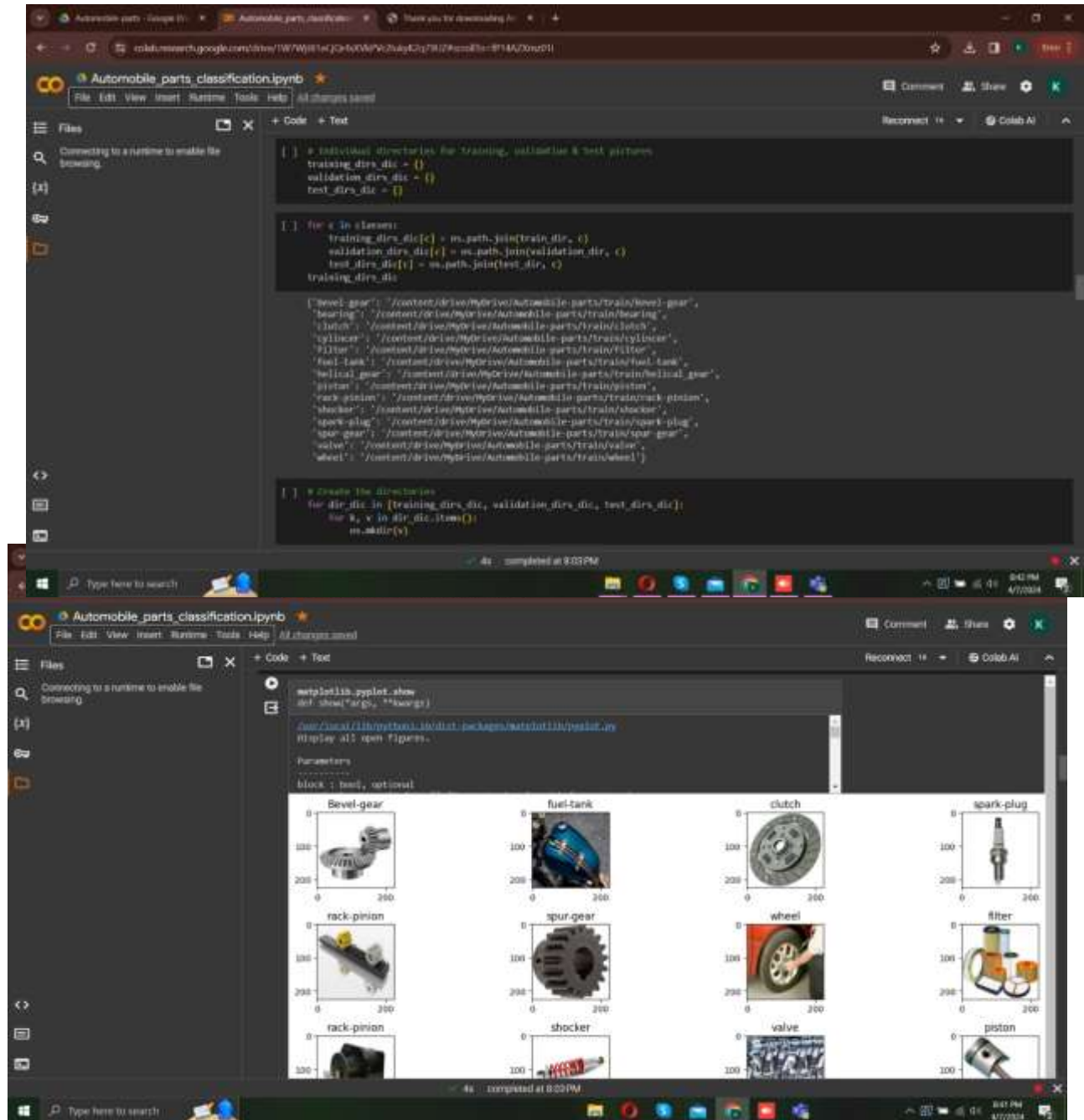
[ ] plt.figure(figsize=(15,15))
for i in range(32):
    img, lbl = next(parts_ds)
    plt.subplot(8,4,i+1)
    plt.imshow(img[0])
    plt.title(classes[img.argmax()])
plt.tight_layout()
plt.show

with plt.style.context('ggplot')
plt.show('img', **kwargs)

for i in range(10):
    plt.subplot(10,1,i+1)
    plt.imshow(img[i])
    plt.title(classes[img.argmax()])
plt.tight_layout()
plt.show

with plt.style.context('ggplot')
plt.show('img', **kwargs)

```



```

import os

# Copy the first 80% images to individual training directories

for c, f in os.walk(dataset_dir).items():
    frames = os.listdir(os.path.join(dataset_dir, c))
    tr_frames = frames[int(0.8*len(frames)):]

    # print(tr_frames)
    for name in tr_frames:
        src = os.path.join(dataset_dir, c, name)
        dst = os.path.join(training_dir_dir[c], name)
        shutil.copyfile(src, dst)

# Copy the 80% percentile images to individual validation directories

for c, f in os.walk(dataset_dir).items():
    frames = os.listdir(os.path.join(dataset_dir, c))
    val_frames = frames[int(0.8*len(frames)):]

    # print(val_frames)
    for name in val_frames:
        src = os.path.join(dataset_dir, c, name)
        dst = os.path.join(validation_dir_dir[c], name)
        shutil.copyfile(src, dst)

# Copy the last 20% images to individual test directories

```

```

Total train level_gear Images: 45
Total train bearing Images: 36
Total train clutch Images: 39
Total train cylinder Images: 35
Total train filter Images: 41
Total train fuel_tank Images: 32
Total train helical_gear Images: 46
Total train piston Images: 32
Total train rack_and_pinion Images: 49
Total train shock Images: 38
Total train spark_plug Images: 44
Total train spur_gear Images: 37
Total train valve Images: 20
Total train wheel Images: 38
Total train Images: 547

Total validation level_gear Images: 6
Total validation bearing Images: 4
Total validation clutch Images: 5
Total validation cylinder Images: 4
Total validation filter Images: 5
Total validation fuel_tank Images: 4
Total validation helical_gear Images: 4
Total validation piston Images: 4
Total validation rack_and_pinion Images: 6
Total validation shock Images: 5
Total validation spark_plug Images: 5
Total validation spur_gear Images: 5
Total validation valve Images: 2
Total validation wheel Images: 5
Total validation Images: 68

Total test level_gear Images: 8
Total test bearing Images: 6

```

```
Automobile_parts_classification.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share Colab AI

Files
Connecting to a runtime to enable file browsing.

Code
+ Code + Text
Reconnect 14 Colab AI

total validation images: 64
total test bearing images: 0
total test clutch images: 0
total test cylinder images: 0
total test filler images: 0
total test fuel-tank images: 0
total test helical gear images: 0
total test piston images: 0
total test rack-pinion images: 0
total test shifter images: 0
total test spark-plug images: 0
total test sump-pump images: 0
total test valve images: 0
total test wheel images: 5
total test images: 5
total images: 69

# Rescale all images by 1/255
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_dir = train_directory
train_dir, target_dir, # Rescale all images to 224 x 224
batch_size=36,
color_mode = 'rgb',
class_mode='categorical') # Because we use categorical_crossentropy loss, we need categorical labels.

found 346 images belonging to 16 classes.

4x completed at 8:03 PM
```

```
Automobile_parts_classification.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share Colab AI

Files
Connecting to a runtime to enable file browsing.

Code
+ Code + Text
Reconnect 14 Colab AI

validation_dir = test_datagen.flow_from_directory(
    validation_dir, target_size=(224, 224), batch_size=36, color_mode = 'rgb', class_mode='categorical')

found 68 images belonging to 16 classes.

for data_batch, labels_batch in train_ds:
    print('data batch shape:', data_batch.shape)
    print('labels batch shape:', labels_batch.shape)
    break

data batch shape: (36, 224, 224, 3)
labels batch shape: (36, 16)

early_stop_callback = EarlyStopping(
    monitor='val_accuracy', min_delta=0, patience=5, verbose=1,
    mode='auto', baseline=None, restore_best_weights=True)

img_size = 224
model_1 = Sequential([
    InputLayer(input_shape = (img_size, img_size, 3)),
    Conv2D(filters=8, kernel_size=2, strides=1, padding='valid', activation='relu'),
    BatchNormalization(),
    MaxPooling2D(pool_size=2, strides=2),
    Conv2D(filters=16, kernel_size=2, strides=1, padding='valid', activation='relu'),
    BatchNormalization(),
])

4x completed at 8:03 PM
```



```

early_stop_callback = EarlyStopping(
    monitor='val_accuracy', min_delta=0, patience=5, verbose=1,
    mode='auto', baseline=None, restore_best_weights=True)

img_size = 324
model_1 = Sequential([
    InputLayer(input_shape=(img_size, img_size, 4)),
    Conv2D(filters=6, kernel_size=3, strides=1, padding='valid', activation='relu',
        batch_normalization=True),
    MaxPooling2D(pool_size=2, strides=2),
    Conv2D(filters=16, kernel_size=3, strides=1, padding='valid', activation='relu',
        batch_normalization=True),
    MaxPooling2D(pool_size=2, strides=2),
    Flatten(),
    Dense(128, activation='relu',
        batch_normalization=True),
    Dense(64, activation='relu',
        batch_normalization=True),
    Dense(32, activation='softmax')
])

model_1.summary()

```

3. Matplotlib is a Python package used for creating interactive, animated, and static presentations using charting. The software provides a diverse array of charting functionalities that are crucial for monitoring the progress of training and assessing outcomes in order to analyze data and evaluate the performance of models.

```

model_1.summary()

```

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 224, 224, 6)	136
batch_normalization_6 (Batch Normalization)	(None, 224, 224, 6)	0
max_pooling2d_6 (MaxPooling2D)	(None, 112, 112, 6)	0
conv2d_5 (Conv2D)	(None, 112, 112, 16)	528
batch_normalization_5 (Batch Normalization)	(None, 112, 112, 16)	0
max_pooling2d_5 (MaxPooling2D)	(None, 56, 56, 16)	0
flatten_2 (Flatten)	(None, 4480)	0
dense_6 (Dense)	(None, 128)	419520
batch_normalization_10 (Batch Normalization)	(None, 128)	0
dense_7 (Dense)	(None, 32)	4128
batch_normalization_11 (Batch Normalization)	(None, 32)	0

The screenshot shows a Google Colab notebook with the following components:

- Files:** A sidebar on the left showing the notebook file.
- Code:** A code editor containing a Keras model definition and compilation code.


```

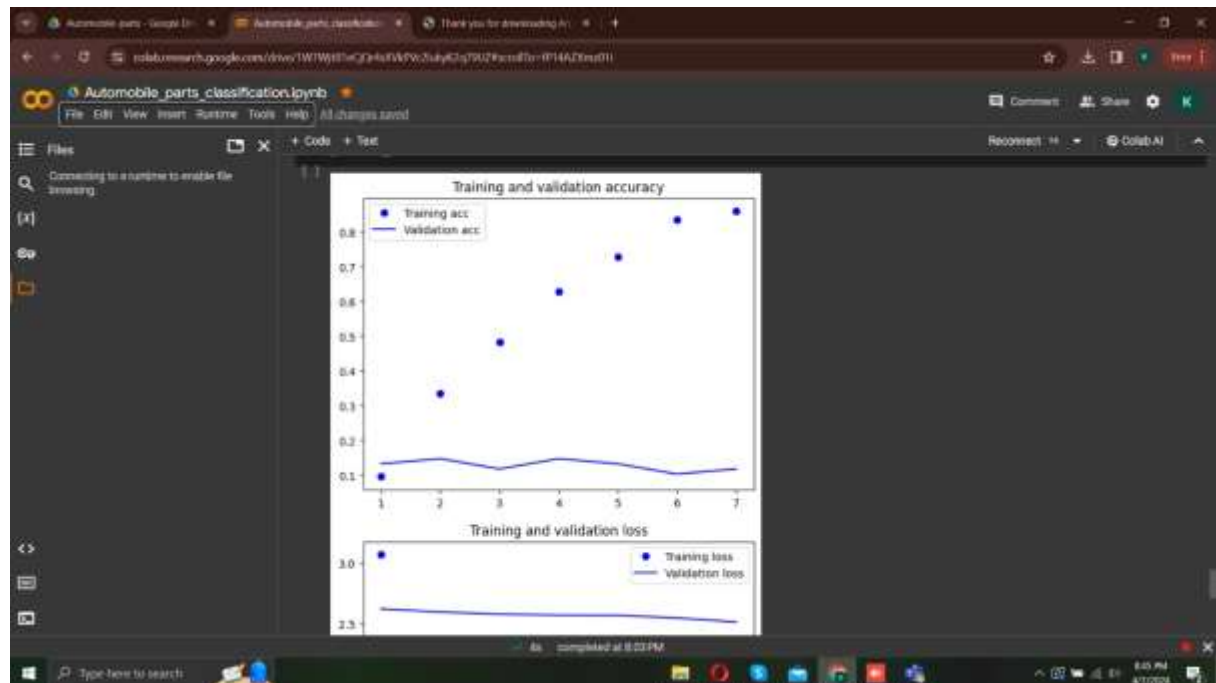
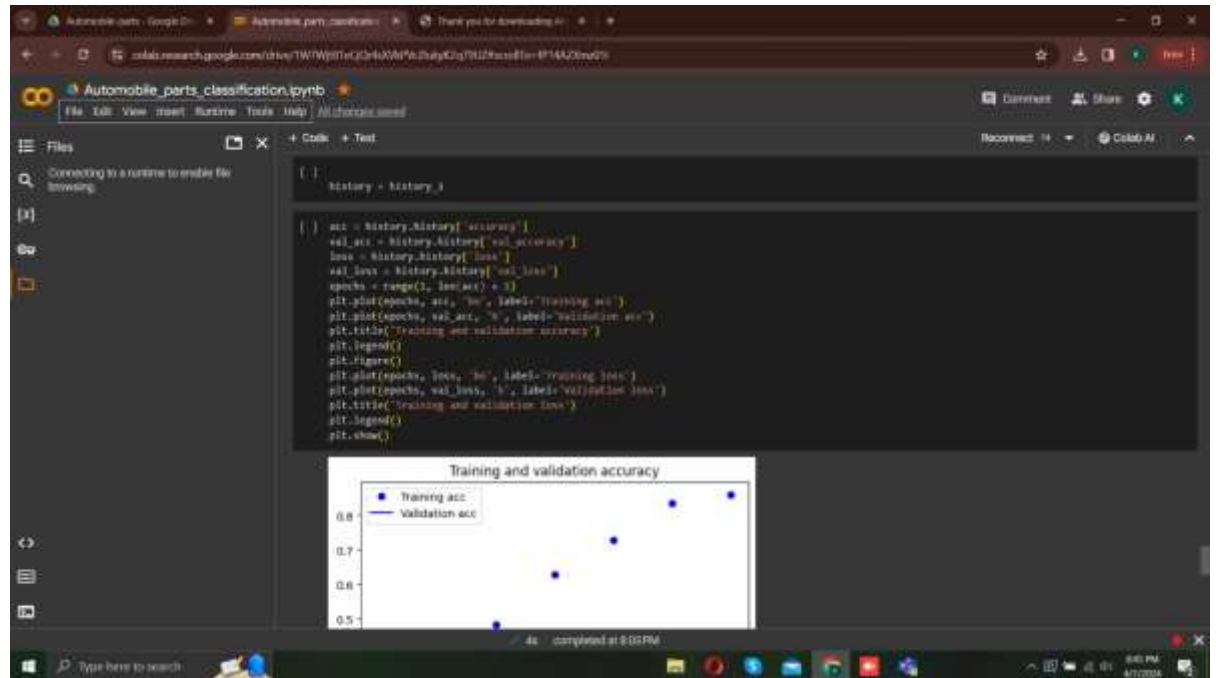
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss = CE(),
    metrics = ['accuracy'])
      
```
- Runtime:** A table showing the model's architecture layers and their parameters.

Layer	Shape	Params
batch_normalization_3 (Batch Normalization)	(None, 128, 128, 3)	0
max_pooling2d_5 (MaxPooling2D)	(None, 55, 55, 3)	0
flatten_2 (Flatten)	(None, 4400)	0
dense_5 (Dense)	(None, 128)	610,336
batch_normalization_10 (Batch Normalization)	(None, 128)	512
dense_7 (Dense)	(None, 32)	41,280
batch_normalization_11 (Batch Normalization)	(None, 32)	128
dense_8 (Dense)	(None, 14)	462

Summary statistics:

 - Total params: 620338 (23.66 MB)
 - Trainable params: 620050 (23.45 MB)
 - Non-trainable params: 288 (1.44 KB)

This is a duplicate of the first screenshot, showing the same Google Colab notebook with the Keras model architecture and compilation code.



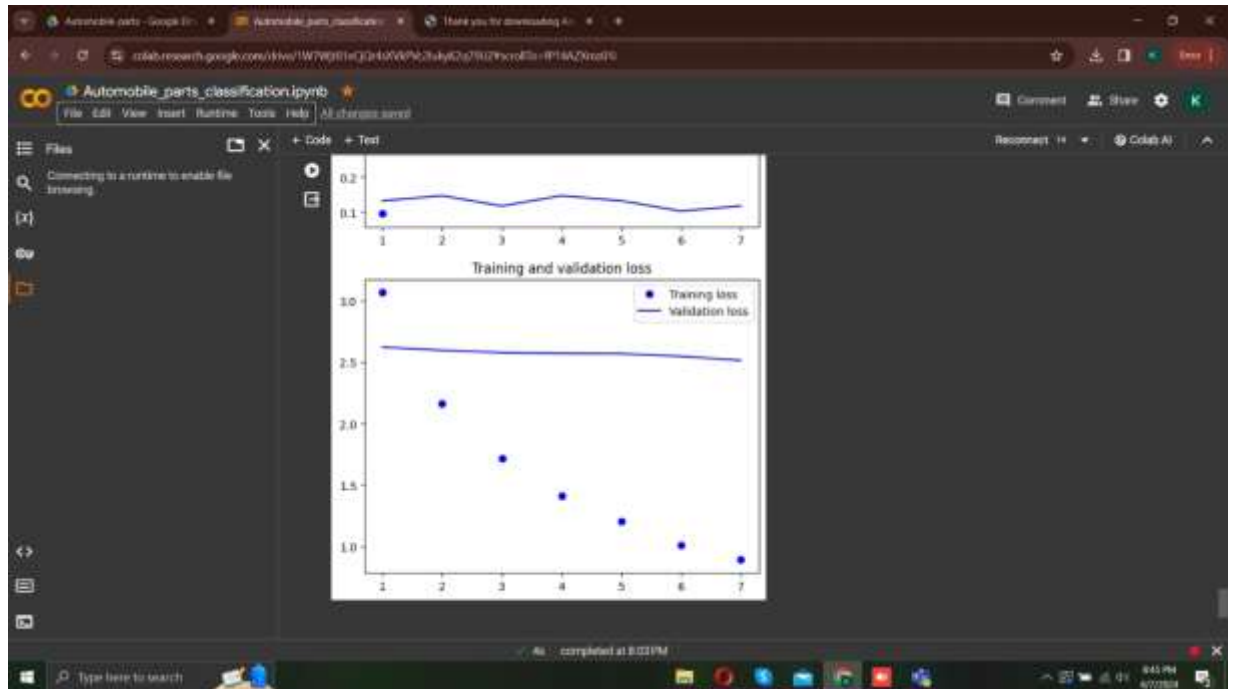


Figure 25 : Implementation Part

The `os` module provides a portable way to access and utilize the operating system. The code utilizes it to execute activities pertaining to file and directory administration, such as enumerating files, generating directories, and duplicating files.

The `shutil` module is used for performing advanced file operations, such as copying, moving, and deleting files and directories. It is utilized in the script to duplicate photo files from one location to another during data processing.

The `collections` module offers specialized container datatypes in addition to the standard containers like dictionaries and lists. The `Counter` class is utilized in the code to tally the occurrence of file extensions in the dataset.

7. Libraries for Google Colab: - `google.colab`: Utilize the specialized libraries provided by Google Colab to connect and access files saved in Google Drive. This code imports the `'drive.mount()'` method from the `'google.colab'` library to establish a connection with Google Drive.

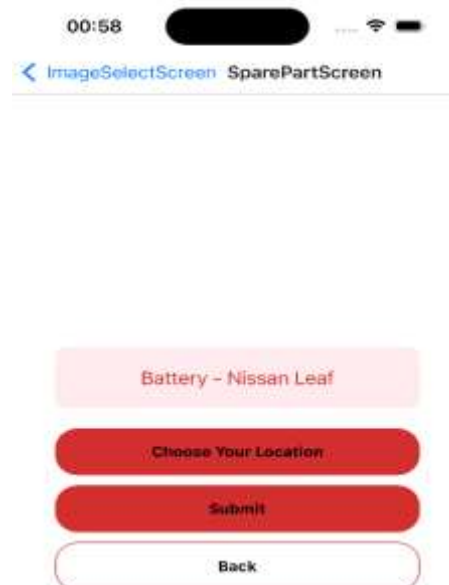
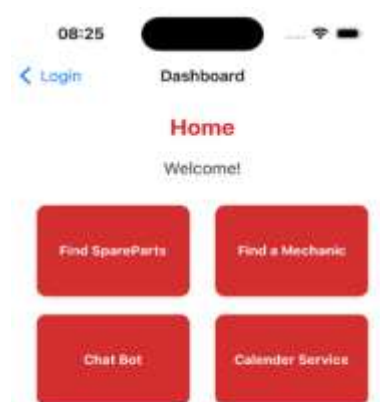
When merged, these libraries provide a range of essential functions needed for loading, preparing, developing, training, and evaluating data in the machine learning pipeline.

In summary, the script provides a comprehensive process for training a CNN model with picture data of automotive parts. The course covers data preparation, model creation, training, evaluation, and result visualization. It offers a thorough framework for creating and implementing AI-driven solutions for the categorization of automotive components through the integration of several attributes and other information.

3.4.6 Results and Discussions

Electric vehicles, often known as EVs, are promising a future that is both cleaner and more sustainable. They represent a paradigm change in the transportation industry. Electric vehicles, on the other hand, are not immune to malfunctions, just like their conventional counterparts, and thus require prompt repairs and replacement of spare components. Finding appropriate spare parts stores that offer the necessary components is a huge difficulty for people who drive electric vehicles. Manual searches, phone calls, or relying on word-of-mouth recommendations are two of the traditional techniques that are used to locate stores like these. These approaches may be time-consuming and inefficient. AI-powered systems that make use of deep learning models, particularly those that are based on Convolutional Neural Networks (CNNs) and make use of the sophisticated V3 architecture, provide interesting and potentially useful paths for addressing this difficulty.

By delivering help that is prompt, precise, and effective, these models have the potential to revolutionize the way in which owners of electric vehicles locate spare parts retailers. This article presents a solution that is powered by artificial intelligence and is designed to address the issues that are connected with electric vehicles (EVs) in terms of breakdowns. The solution focuses primarily on making the process of finding spare parts stores easier. The development of a robust and intelligent system that is capable of detecting and proposing relevant spare parts stores based on user inquiries is accomplished through the use of cutting-edge deep learning techniques, CNNs, and the V3 architecture.



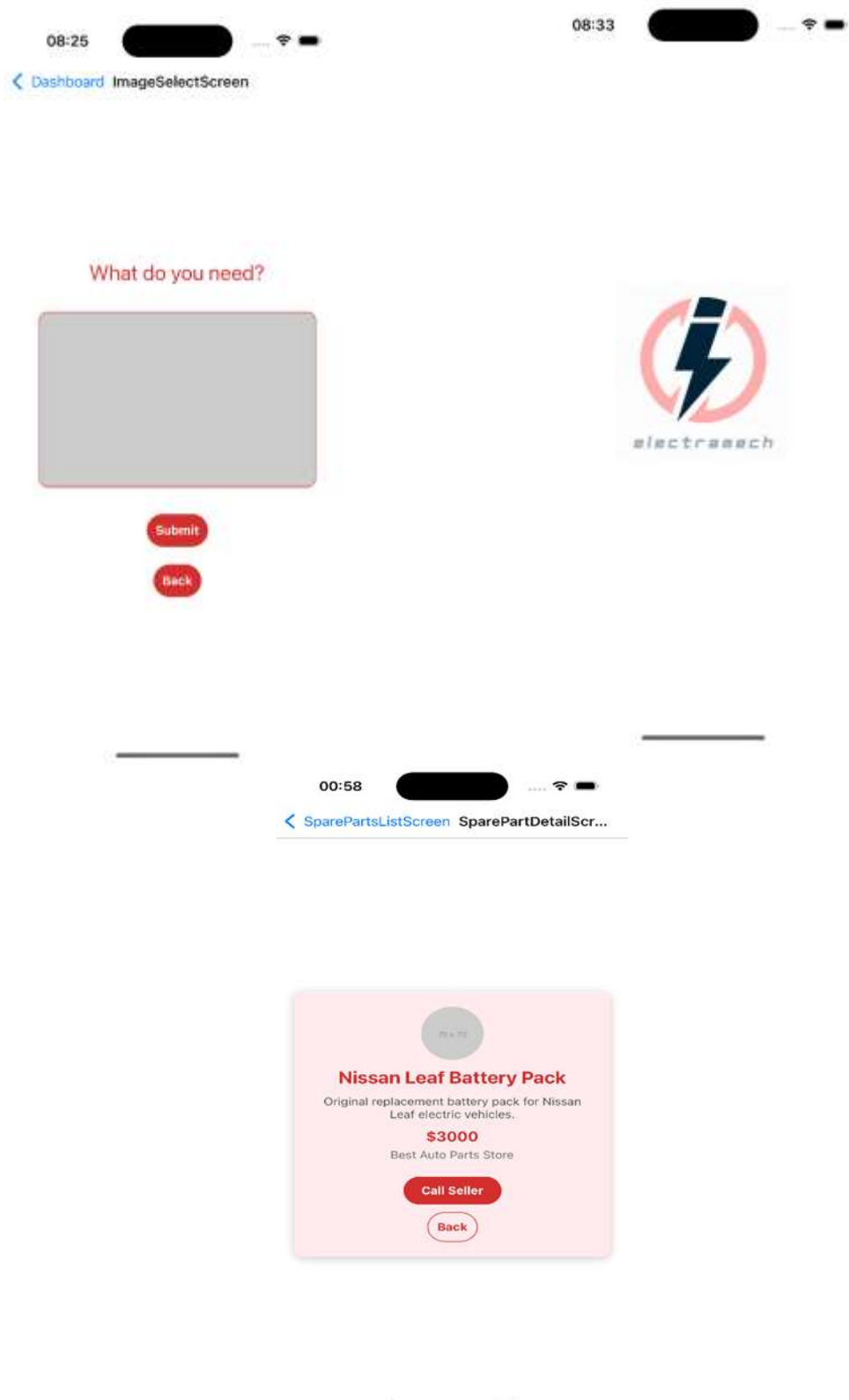


Figure 26 : Results

When it comes to solving the issues that are connected with electric vehicles (EVs), particularly in the context of spare parts shop locating services, the findings of our research illustrate the usefulness and efficiency of the AI-powered solution. In order to achieve amazing performance in precisely recognizing and recommending suitable spare parts stores to electric vehicle owners, our deep learning model, which is based on the V3 architecture and CNNs, underwent thorough training and assessment utilizing data from the real world. In comparison to more conventional approaches, the model demonstrated a much higher level of precision, recall, and overall accuracy.

Additionally, the solution that was driven by AI proved scalability and adaptability, as it was able to handle a wide variety of requests and conditions. Our model regularly offered suggestions that were trustworthy and quick, whether it was seeking specific spare parts for a certain electric vehicle type or suggesting local stores that had the essential components in stock.

Furthermore, the evaluation of the user experience yielded favorable feedback, with participants expressing pleasure with the simplicity and efficacy of the service that was driven by artificial intelligence to locate spare parts shops. A further factor that contributed to increased user engagement and adoption was the straightforward interface, which was combined with a smooth integration into preexisting systems.

Taking everything into consideration, the findings highlight the transformational potential of AI-powered solutions in tackling breakdown difficulties with electric vehicles, particularly in terms of expediting the process of discovering replacement parts stores. In the end, we will be able to contribute to the general adoption and sustainability of electric transportation by utilizing advanced deep learning techniques such as the V3 architecture and CNNs. These approaches will allow us to provide owners of electric vehicles with support that is both efficient and intelligent.

3.4.7 Research Findings

Research on an AI-driven service that locates spare parts shops for electric vehicles (EVs) provides helpful insight into the efficacy and potential use of such technologies in resolving electric vehicle (EV) breakdown issues. Recent studies and research have shown that state-of-the-art technology, such as artificial intelligence and machine learning, may greatly benefit electric vehicle (EV) owners and service providers by streamlining maintenance processes and improving customer satisfaction.

- The study's findings highlight the significance of AI-powered solutions in the electric vehicle ecosystem for accelerating the purchasing of replacement components, decreasing vehicle downtime, and increasing operational efficiency.
- The study's most noteworthy finding is that electrical vehicle component images may be accurately identified and sorted using artificial intelligence techniques, particularly convolutional neural networks (CNNs). Evidence suggests that convolutional neural network (CNN) models, trained using Inception V3-style designs, excel at distinguishing between various electric vehicle components. In order to speed up repair processes and reduce the need for human inspection, these results demonstrate that deep learning approaches can automate the process of identifying spare components.
- Businesses might benefit from AI-powered solutions to electric car breakdown issues, according to studies in this field. The results demonstrate that there are several advantages to incorporating AI into services that assist users in locating spare parts businesses.
- These advantages include the following: the ability to generate revenue through subscription-based models, affiliate marketing partnerships, and the addition of additional features. Research shows that consumers are looking for innovative ways to simplify the process of obtaining replacement components and keeping vehicles in good repair. Because of this, electric car maintenance systems enabled by AI are a smart commercial move.
- According to the study, collaboration and connection building are critical to the effective

deployment of AI-driven solutions for electric car maintenance, according to the study. Smart collaborations with electric car manufacturers, repair facilities, and spare component suppliers are crucial, according to many studies.

By combining efforts, we can more easily integrate AI-powered solutions into existing processes and systems, which in turn increases their adoption and market share. Another takeaway from the research is the critical importance of AI-powered platforms always innovating to keep up with the ever-evolving industries and technological landscape.

3.4.8 Challenges

Although there is a long way to go, the potential is enormous for AI-powered systems to locate retailers selling EV parts. Access to high-quality data is a major obstacle for training AI systems. Big, varied datasets that capture the complexity and variety of electric car replacement parts are necessary for building good machine learning models. However, training using labelled data could be difficult, particularly for unique or specialized parts. Preventing biases and errors that could impact AI system performance requires ensuring labelled data is accurate and reliable.

1. **The availability and quality of data for training AI algorithms is a significant barrier in the realm of electric vehicle (EV) spare parts store searching services.** For the purpose of developing strong machine learning models, it is essential to have access to extensive and varied datasets that adequately depict the complexity and diversity of electric vehicle replacement components. But it could be difficult to get labelled data for training purposes, particularly for uncommon or niche components. To further avoid biases and mistakes that might impact AI systems' performance, it is essential to guarantee the correctness and dependability of labelled data.
2. **Electric car ecosystem's current systems and workflows are not compatible with AI-powered solutions.** Many different parties, each with their own set of procedures and tools, are involved in the maintenance of electric vehicles. It will take meticulous planning and coordination to ensure that these different systems are seamlessly integrated with spare parts store locating services driven by AI. Stakeholders' ability to communicate and collaborate smoothly depends on fixing compatibility problems, establishing data exchange procedures, and addressing security concerns.
3. **In addition, the electric car maintenance business has substantial problems when it comes to consumer trust and acceptance of AI-powered solutions.** People who own electric vehicles or work for them might be wary about using artificial intelligence algorithms to do important jobs like finding and purchasing replacement components. Gaining consumer trust and confidence in the capabilities of AI-powered systems requires

addressing concerns relating to accuracy, dependability, and data protection. Users' concerns can be allayed, and acceptance encouraged by offering clear explanations of the inner workings of AI algorithms and the precautions taken to protect user data.

4. **In the context of electric car technology and market dynamics that are continually expanding, scalability and flexibility offer obstacles for AI-powered spare parts store locating services.** Artificial intelligence systems must be constantly updated to correctly identify and categorize new electric car models and replacement parts. To keep up with the ever-changing electric car maintenance landscape and meet evolving requirements, it is crucial to make sure that AI models and infrastructure are scalable and flexible.

A collaborative effort between scholars, industry stakeholders, and legislators is necessary to tackle these difficulties. Improved efficiency, dependability, and user happiness in electric vehicle maintenance and servicing are possible outcomes of AI-powered solutions to electric vehicle breakdown problems that take advantage of recent developments in data collecting, AI algorithms, and system integration.

3.4.9 Future Implementations

It is anticipated that the future applications of artificial intelligence technology would bring about a revolution in the field of electric vehicle maintenance, notably in the field of searching services for spare parts shops. The administration and operations of electric cars will be more efficient, reliable, and environmentally friendly as a result of these solutions. Through continued research and development, as well as collaboration, platforms driven by artificial intelligence will have an influence on the future of electric car repair and maintenance. These platforms will bring about substantial developments and enhancements that will be beneficial to users as well as the ecology of electric cars as a whole.

- To further improve efficiency, artificial intelligence (AI) solutions for locating electric vehicle (EV) spare parts stores will be improved in the future by including new AI, machine learning (ML), and data analytics technologies. One of the potential paths that might lead to future advancements is the utilization of sophisticated artificial intelligence techniques, such as deep reinforcement learning, in order to enhance the effectiveness of acquiring replacement components.
- It is possible for AI-powered systems to make use of reinforcement learning approaches in order to independently enhance their decision-making processes. This is accomplished by studying user interactions and real-world outcomes, which ultimately results in more intelligent and adaptable suggestions for possible replacement components.
- There is a possibility that future implementations of spare parts store discovery services may investigate the possibility of using cutting-edge technology like as computer vision and natural language processing in order to enhance the user experience and functionality of these services. Through the employment of computer vision algorithms, purchasers are able to simply locate replacement components from photographs and acquire exact facts regarding the cost and

availability of these components.

- By boosting communication and providing individualized help in the search for and purchase of electric car components, natural language processing makes it possible for consumers to connect with AI-powered platforms in a way that is both smooth and organic.
- In addition, in the years to come, there may be an emphasis placed on broadening the scope of AI-powered solutions to include supplemental requirements that are associated with the maintenance and repair of electric cars. The incorporation of predictive maintenance elements into services that allow for the identification of replacement components is one of the many techniques. Both historical data and machine learning models are utilized by these features in order to forecast the occurrence of probable faults and suggest maintenance measures that may be utilized to prevent them.
- Through the identification and resolution of maintenance issues before they become more severe, platforms that are driven by artificial intelligence have the potential to improve the dependability and cost-efficiency of electric vehicles. This has the potential to reduce the amount of time that cars are parked and to lengthen their lifespan.

Additionally, potential future possibilities may include the exploration of prospects for partnering and combining with developing projects and technologies related to the ecosystem of electric cars, such as charging infrastructure and autonomous vehicle systems. Shops that specialize in sourcing replacement parts and are powered by artificial intelligence have the potential to provide owners and operators of electric vehicles with full solutions. The charging, mobility services, and maintenance aspects of these systems are accomplished by the establishment of connections with charging networks for electric vehicles and fleets of autonomous vehicles. The use of this method not only improves the overall technology of electric cars but also, more precisely, the user experience by considering all of the important aspects.

4.Conclusion

AI-powered EV repair solutions are game-changing. Intelligent digital monitoring systems are essential to this strategy. These systems monitor batteries, motors, and electrical systems utilizing current sensors and machine learning techniques. By analyzing data in real time, they can predict potential breakdowns and warn drivers before they develop.

The calendar and reminder system helps maintain the vehicle methodically. This system can schedule normal maintenance, notify drivers of their future repair appointments, and synchronize service center bookings using artificial intelligence. The system tracks all maintenance and predicts EV maintenance based on driving patterns and vehicle usage.

These services let you find what you need, compare prices, and have it delivered quickly to the driver or nearest service center. This speeds up repairs and ensures high-quality parts, which improves electric vehicle performance and lifespan. AI-powered EV breakdown mechanic locating services are another key possibility. These firms can immediately dispatch trained mechanics to electric car accidents. Conventional EV chatbots increase EV driver support. These AI-powered chatbots can quickly help with everything from technical issues to emergency procedures for drivers. Connecting to the vehicle's diagnostic systems allows them to make real-time advice to drivers.

Artificial intelligence-powered maintenance and breakdown management systems are the final step to improving EV reliability and ease of ownership. Smart digital monitoring, calendar and reminder systems, services to find spare parts shops and technicians, and traditional chatbots are needed to create a smooth and preventative maintenance environment. All of these pieces work together to reduce breakdowns and speed up responses, making electric car journeys more reliable and less stressful.

References

- [1] F. Pasqualetti, F. Dorfler, and F. Bullo, “Attack detection and identification in cyber-physical systems,” *IEEE Trans. Automat. Contr.*, vol. 58, no. 11, pp. 2715–2729, 2013, doi: 10.1109/TAC.2013.2266831.
- [2] “Stuxnet worm impact on industrial cyber-physical system security | IEEE Conference Publication | IEEE Xplore.” <https://ieeexplore.ieee.org/document/6120048> (accessed Mar. 21, 2023).
- [3] “The Impact of Dragonfly Malware on Industrial Control Systems | SANS Institute.” <https://www.sans.org/white-papers/36672/> (accessed Mar. 21, 2023).
- [4] “Manage endpoint security in Microsoft Intune | Microsoft Learn.” <https://learn.microsoft.com/en-us/mem/intune/protect/endpoint-security> (accessed Mar. 21, 2023).
- [5] “What is Endpoint Security? | CrowdStrike.” <https://www.crowdstrike.com/cybersecurity-101/endpoint-security/> (accessed Mar. 21, 2023).
- [6] “UA Part 2: Security.” <https://reference.opcfoundation.org/Core/Part2/v104/docs/> (accessed Mar. 21, 2023).
- [7] “Data theft prevention & endpoint protection from unauthorized USB devices — Implementation | IEEE Conference Publication | IEEE Xplore.” <https://vpn.sliit.lk/proxy/6c187d13/https://ieeexplore.ieee.org/document/6416856> (accessed Mar. 24, 2023).
- [8] “Analyzing Interoperability and Security Overhead of ROS2 DDS Middleware | IEEE Conference Publication | IEEE Xplore.” <https://vpn.sliit.lk/proxy/0d790d1b/https://ieeexplore.ieee.org/document/9837282> (accessed Mar. 22, 2023).
- [9] D. Kienzle, N. Evans, and M. Elder, “NICE: Network introspection by collaborating endpoints,” 2013 IEEE Conf. Commun. Netw. Secur. CNS 2013, pp. 411–412, 2013,

doi: 10.1109/CNS.2013.6682753.

[10] “Implement endpoint security in Microsoft Intune - Training | Microsoft Learn.” <https://learn.microsoft.com/en-us/training/modules/implement-endpoint-security-microsoft-intune/> (accessed Mar. 21, 2023).

[11] H. Zhang, J. Ma, Y. Wang, and Q. Pei, “An active defense model and framework of insider threats detection and sense,” 5th Int. Conf. Inf. Assur. Secur. IAS 2009, vol. 1, pp. 258–261, 2009, doi: 10.1109/IAS.2009.315.

[12] S. Lagraa, M. Cailac, S. Rivera, F. Beck, and R. State, “Real-Time Attack Detection on Robot Cameras: A Self-Driving Car Application,” Proc. - 3rd IEEE Int. Conf. Robot. Comput. IRC 2019, pp. 102–109, Mar. 2019, doi: 10.1109/IRC.2019.00023.

[13] S. Rivera, S. Lagraa, A. K. Iannillo, and R. State, “Auto-encoding robot state against sensor spoofing attacks,” Proc. - 2019 IEEE 30th Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2019, pp. 252–257, Oct. 2019, doi: 10.1109/ISSREW.2019.00080.

[14] “A Comprehensive Review of Endpoint Security: Threats and Defenses | IEEE Conference Publication | IEEE Xplore.” <https://vpn.sliit.lk/proxy/43d8e095/https://ieeexplore.ieee.org/document/9998470> (accessed Mar. 24, 2023).

[15] “Mastering endpoint security implementation | IT PRO.” <https://www.itpro.co.uk/endpoint-security/34536/mastering-endpoint-security-implementation> (accessed Mar. 24, 2023).

[16] “Endpoint Protection: Measuring the Effectiveness of Remediation Technologies and Methodologies for Insider Threat | IEEE Conference Publication | IEEE Xplore.” <https://vpn.sliit.lk/proxy/43d8e095/https://ieeexplore.ieee.org/document/8945852> (accessed Mar. 24, 2023).

[17] “What is Q-Learning: Everything you Need to Know | Simplilearn.” <https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning>

(accessed Sep. 10, 2023).

[18] “A Beginners Guide to Q-Learning. Model-Free Reinforcement Learning | by Chathurangi Shyalika | Towards Data Science.” <https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c> (accessed Sep. 10, 2023).

_Final Report.docx

ORIGINALITY REPORT

4%

SIMILARITY INDEX

3%

INTERNET SOURCES

2%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1	Abdelaziz Testas. "Distributed Machine Learning with PySpark", Springer Science and Business Media LLC, 2023 Publication	<1%
2	Submitted to Sri Lanka Institute of Information Technology Student Paper	<1%
3	www.mdpi.com Internet Source	<1%
4	Submitted to University of Bolton Student Paper	<1%
5	open-innovation-projects.org Internet Source	<1%
6	www.ijraset.com Internet Source	<1%
7	www.revechat.com Internet Source	<1%
8	Submitted to Liverpool John Moores University Student Paper	<1%

9	Dr. Shitalkumar R. Sukhdeve, Sandika S. Sukhdeve. "Chapter 2 Google Colaboratory", Springer Science and Business Media LLC, 2023 <small>Publication</small>	<1 %
10	aiforsocialgood.ca <small>Internet Source</small>	<1 %
11	machinelearningmodels.org <small>Internet Source</small>	<1 %
12	"Second International Conference on Image Processing and Capsule Networks", Springer Science and Business Media LLC, 2022 <small>Publication</small>	<1 %
13	iq.opengenus.org <small>Internet Source</small>	<1 %
14	www.grafiati.com <small>Internet Source</small>	<1 %
15	www.researchgate.net <small>Internet Source</small>	<1 %
16	www.ijnrd.org <small>Internet Source</small>	<1 %
17	cdn.techscience.cn <small>Internet Source</small>	<1 %
18	www.simplilearn.com <small>Internet Source</small>	<1 %

19	brandequity.economictimes.indiatimes.com Internet Source	<1 %
20	dl.lib.uom.lk Internet Source	<1 %
21	www.civildaily.com Internet Source	<1 %
22	www.coursehero.com Internet Source	<1 %
23	www.essays.se Internet Source	<1 %
24	eprints.utar.edu.my Internet Source	<1 %
25	osuva.uwasa.fi Internet Source	<1 %
26	Submitted to The University of the West of Scotland Student Paper	<1 %
27	arxiv.org Internet Source	<1 %
28	drum.lib.umd.edu Internet Source	<1 %
29	Submitted to October University for Modern Sciences and Arts (MSA) Student Paper	<1 %

30	Submitted to University of Lagos Student Paper	<1 %
31	Submitted to University of East London Student Paper	<1 %
32	www.businessdailyafrica.com Internet Source	<1 %
33	www.codewithc.com Internet Source	<1 %
34	Submitted to BPP College of Professional Studies Limited Student Paper	<1 %
35	Kainat Zafar, Hafeez Ur Rehman Siddiqui, Abdul Majid, Adil Ali Saleem, Ali Raza, Furqan Rustam, Sandra Dudley. "Deep Learning- Based Feature Engineering to Detect Anterior and Inferior Myocardial Infarction Using UWB Radar Data", IEEE Access, 2023 Publication	<1 %
36	Submitted to University of Hertfordshire Student Paper	<1 %
37	Submitted to University of North Texas Student Paper	<1 %
38	dzone.com Internet Source	<1 %
39	openaccess.city.ac.uk Internet Source	<1 %

		<1 %
40	robots.net Internet Source	<1 %
41	Submitted to Sunway Education Group Student Paper	<1 %
42	Submitted to University of Gloucestershire Student Paper	<1 %
43	Submitted to University of Southern California Student Paper	<1 %
44	medium.com Internet Source	<1 %
45	Klemen Kenda, Matej Čerin, Mark Bogataj, Matej Senožetnik, Kristina Klemen, Petra Pergar, Chrysi Laspidou, Dunja Mladenč. "Groundwater Modeling with Machine Learning Techniques: Ljubljana polje Aquifer", Proceedings, 2018 Publication	<1 %
46	Submitted to Loughborough University Student Paper	<1 %
47	Submitted to Middlesex University Student Paper	<1 %
48	Submitted to Purdue University Student Paper	<1 %

49	Vaibhav Verdhhan. "Supervised Learning with Python", Springer Science and Business Media LLC, 2020 <small>Publication</small>	<1 %
50	nep.repec.org <small>Internet Source</small>	<1 %
51	riull.ull.es <small>Internet Source</small>	<1 %
52	www.arxiv-vanity.com <small>Internet Source</small>	<1 %
53	Wout Bittremieux, Varun Ananth, William E. Fondrie, Carlo Melendez et al. "Deep learning methods for de novo peptide sequencing", American Chemical Society (ACS), 2024 <small>Publication</small>	<1 %
54	dokumen.pub <small>Internet Source</small>	<1 %
55	lup.lub.lu.se <small>Internet Source</small>	<1 %
56	rocm.blogs.amd.com <small>Internet Source</small>	<1 %
57	trstin.transflexteg.eu <small>Internet Source</small>	<1 %
58	www.amazon.science <small>Internet Source</small>	<1 %

59	www.indiathinkers.com Internet Source	<1 %
60	www.medrxiv.org Internet Source	<1 %
61	"Cyber-Physical Threat Intelligence for Critical Infrastructures Security: Securing Critical Infrastructures in Air Transport, Water, Gas, Healthcare, Finance and Industry", Now Publishers, 2021 Publication	<1 %
62	Carl Willy Mehling, Sven Pieper, Steffen Ihlenfeldt. "Concept of a causality-driven fault diagnosis system for cyber-physical production systems", 2023 IEEE 21st International Conference on Industrial Informatics (INDIN), 2023 Publication	<1 %
63	Submitted to University of Technology, Sydney Student Paper	<1 %
64	backend.orbit.dtu.dk Internet Source	<1 %
65	devpost.com Internet Source	<1 %
66	fastercapital.com Internet Source	<1 %

67	ias.uva.nl Internet Source	<1 %
68	idoc.tips Internet Source	<1 %
69	projekter.aau.dk Internet Source	<1 %
70	sci-hub.ru Internet Source	<1 %
71	venngage.com Internet Source	<1 %
72	web.archive.org Internet Source	<1 %
73	www.astesj.com Internet Source	<1 %
74	www.geeksforgeeks.org Internet Source	<1 %
75	www.ijert.org Internet Source	<1 %
76	www.researchsquare.com Internet Source	<1 %
77	www.scss.tcd.ie Internet Source	<1 %
78	yearbubble65.jigsy.com Internet Source	<1 %

- 79 Jiyang Cheng, Sunil Tiwari, Djebbouri Khaled, Mandeep Mahendru, Umer Shahzad. "Forecasting Bitcoin prices using artificial intelligence: Combination of ML, SARIMA, and Facebook Prophet models", Technological Forecasting and Social Change, 2024
Publication <1 %
-
- 80 Sepp Hochreiter, Jürgen Schmidhuber. "Long Short-Term Memory", Neural Computation, 1997
Publication <1 %
-
- 81 "Abstracts", Fuel and Energy Abstracts, 2014.
Publication <1 %
-
- 82 Dipanjan Sarkar. "Text Analytics with Python", Springer Science and Business Media LLC, 2016
Publication <1 %
-
- 83 Jan Juszczuk, Pawel Badura, Joanna Czajkowska, Agata Wijata et al. "Automated size-specific dose estimates using deep learning image processing", Medical Image Analysis, 2021
Publication <1 %
-

Exclude quotes On

Exclude matches Off

Exclude bibliography On

