Intelligent Car Diagnosis Application using Natural Language Processing and Decision Trees Algorithms

by

Louay Mustafa Mohamed Rashad ElGarhy

A dissertation submitted in partial fulfillment of the requirements for the degree of

Bachelor of computer science (SE programme) in the

Faculty of Computer Science of the

October University for Modern Sciences and Arts (MSA), EGYPT

Graduation Projects advisor:

Dr. Moataz Samy

(July 2023)

Abstract

Diagnosis of car fault is a difficult task as it requires huge knowledge and experience to diagnose. For this reason, car owners frequently asking for skilled mechanic for diagnosing their car if there is a fault. However, most of car faults doesn’t need a mechanic to visit and can be handled by the car owner. A Smart car diagnoses system that helps the car owner to diagnose and fix car faults will be importance. Diagnoses systems are frequently needed in such fields as medical for providing diagnose from a database and car fault diagnosis that can be used by car owners. The purpose of this research is to implement an intelligence system for car fault diagnosis for car owners. The data was absorbed by several existing sites and projects, and the results of this intelligence system show that it can be used for car fault diagnosis by car owners. This will enable car owners to be able to find the car problem and fix it or to visits the nearest mechanic if the car owner can’t fix it manually.

**Acknowledgments**

I would like to own my gratitude to my supervisor, Dr. Moataz Samy, who was always in help and get the opportunity to present my Graduation project, and mentored me in the right path. In addition, my huge thanks to Prof. Islam ElShaarawy, for his continuous support and his time he has paid to let me throughout the graduation project and always giving solution in anytime. He was always willing to help in anytime during campus or even online. I also wish to express my thanks Dr. Ahmed Farouk for giving us the mentality of problem solving which made me master in it inside programming and in other fields as well. Moreover, My huge thanks for Prof. Ayman Ezzat and Prof. Hesham Mansour for their continuous help for the whole committee. Finally, I want to say a heartfelt thank you to everyone who helped me finish my graduation project.

# Contents

Contents

Abstract ……………………………………………………………………………………………………………………………………ii

Acknowledgments……………………………………………………………………………………………………………………iii

[Contents 4](#_Toc128933299)

[List of Tables 7](#_Toc128933300)

[List of Figures 8](#_Toc128933301)

[Introduction 9](#_Toc128933302)

[1.1 Introduction guide lines 9](#_Toc128933303)

[Chapter 2 - Background 10](#_Toc128933304)

[2.1 General Guidelines 10](#_Toc128933305)

[2.2 Example 11](#_Toc128933306)

[Specification - (SRS) 12](#_Toc128933307)

[Introduction 12](#_Toc128933308)

[3.1 Scope of this document 12](#_Toc128933309)

[3.2 Overview 13](#_Toc128933310)

[General Description 14](#_Toc128933311)

[3.3 Product Functions 14](#_Toc128933312)

[3.4 Similar System Information 14](#_Toc128933313)

[3.5 User Characteristics 14](#_Toc128933314)

[3.6 User Problem Statement 14](#_Toc128933315)

[3.7 User Objectives 14](#_Toc128933316)

[3.8 General Constraints 14](#_Toc128933317)

[Functional Requirements 15](#_Toc128933318)

[Interface Requirements 15](#_Toc128933319)

[3.9 User Interfaces 16](#_Toc128933320)

[3.9.1 GUI 16](#_Toc128933321)

[3.10 Hardware Interfaces 16](#_Toc128933322)

[Performance Requirements 17](#_Toc128933323)

[Design Constraints 17](#_Toc128933324)

[3.11 Standards Compliance 17](#_Toc128933325)

[Other non-functional attributes 17](#_Toc128933326)

[3.12 Security 17](#_Toc128933327)

[3.14 Reliability 17](#_Toc128933328)

[3.16 Portability 17](#_Toc128933329)

[3.18 Re-usability 17](#_Toc128933330)

[3.20 Resource Utilization 18](#_Toc128933331)

[Preliminary Object-Oriented Domain Analysis 18](#_Toc128933332)

[Operational Scenarios 19](#_Toc128933333)

[Software Design 20](#_Toc128933334)

[Introduction 20](#_Toc128933335)

[4.1 User Interfaces 20](#_Toc128933336)

[4.2 Scope 20](#_Toc128933337)

[4.3 Overview 21](#_Toc128933338)

[4.4 Definitions and Acronyms 21](#_Toc128933339)

[System Overview 21](#_Toc128933340)

[System Architecture 22](#_Toc128933341)

[4.5 Architectural Design 22](#_Toc128933342)

[4.5.1 Data Collection 22](#_Toc128933348)

[4.5.2 Data Preprocessing: 22](#_Toc128933349)

[4.5.4 Deployment: 22](#_Toc128933350)

[4.5.5 Maintenance: 22](#_Toc128933351)

[4.6 Decomposition Description 23](#_Toc128933352)

[4.6.1 User Interface Subsystem 23](#_Toc128933353)

[Data Design 24](#_Toc128933354)

[4.7 Data Description 24](#_Toc128933355)

[4.7.1 Data collection 24](#_Toc128933356)

[4.7.2 Cleaning Data 26](#_Toc128933357)

[4.7.3 Data transformation 26](#_Toc128933358)

[4.7.4 Machine learning tools - classification 27](#_Toc128933359)

[4.7.5 Decision tree 28](#_Toc128933360)

[4.8 Data Dictionary 31](#_Toc128933361)

[Dataset Design 31](#_Toc128933362)

[Human Interface Design 31](#_Toc128933363)

[4.9 Overview of User Interface 31](#_Toc128933364)

[4.10 Screen Images 33](#_Toc128933365)

[33](#_Toc128933366)

[4.11 Screen Objects and Actions 33](#_Toc128933367)

[Bibliography 34](#_Toc128933368)

# List of Tables

Table 4.1: Before and after OneHotEncoding 27

Table 4.2: MODULES OF THE DEVELOPED SYSTEM AND THEIR DESCRIPTION 31

Table 4.3: The Dataset view used for training of the Intelligent Car Diagnosis Application 31

# List of Figures

[Fig 3.1: Working of the Intelligent Car Diagnosis Application. 15](#_Toc128930782)

[Fig 3.2: Working of the Intelligent Car Diagnosis Application from end-to-end processes. 15](#_Toc128930782)

[Fig 4.1: Architectural Design. 19](#_Toc128930782)

[Fig 4.2: Dataflow Diagram of the Intelligent Car Diagnosis Application. 20](#_Toc128930782)

[Fig 4.3: Processes Model for Intelligent Car Diagnosis Application. 21](#_Toc128930782)

[Fig 4.4: A Standard Data Analysis Pipeline. 21](#_Toc128930782)

[Fig 4.5: Decision tree 25](#_Toc128930782)

Fig. 4.6: (a) A small hierarchy. Class label names reflect the position in the hierarchy, e.g., ‘2.1’ is a subclass of ‘2’. (b) The set of classes {1,2,2.2}, indicated in bold in the hierarchy, and represented as a vector 26.

Fig 4.7: Real time work flow and interface of the Intelligent Car Diagnosis Application with human 28

Fig 4.8: Real time work flow and interface of the Intelligent Car Diagnosis Application with human 28

Fig 4.9: Flowchart for User Module 29

**Chapter 1**

# Introduction

The ability to learn is one of the most fundamental attributes of intelligent behavior [1], and to learn. The procedure of diagnosing a car problem is challenging and needs a large amount of knowledge, and to diagnose a car fault found by the owner it would be difficult. car owners need professional car mechanic. Nonetheless, some errors are simple and won't need the assistance of expert mechanics. It would be vital to have an automated system that can assist car owners in identifying and repairing typical car issues [2].

## Introduction guide lines

The purpose of this project is to create a chatbot interface for a car diagnostic application using decision tree algorithm to be trained by historical symptoms which leads to diagnose the car faults, and it will be integrating Google Maps API, the app will let the users save their time as well as the application already specified their problem.

This project aims to provide a user-friendly and accessible tool for auto problem diagnosis without the need for specialized knowledge to car owners and auto experts.

# Chapter 2 - Background

## General Guidelines

This research focus on a cross-platform application with open-source languages for implementing the user interface and Machine Learning (ML) Algorithms. Rule-based decision trees, Expert systems and more have been used for car fault diagnosis [2-8]. In addition, decision trees have also been proposed for car fault diagnosis. Expert systems are reliable, accurate and cost effective [2-3]. The knowledge base and techniques used for data by Expert Systems in car problem detection, however, also need to be continually upgraded due to the continuously changing specifications. This research purpose is to improve the expert systems diagnosis accuracy by ML techniques and cross platform application which makes the user more accessible to the system. The system proposed targets anyone have a car and highlights common issues in car faults, which makes it different from current systems that primarily target car mechanics and focus on specific car components like the gearbox and brake systems. has the following contributions, though: Qualitative approaches were used to identify the typical issues with particular vehicles, and a proposed expert system can assist drivers and unskilled mechanics in identifying such issues.

## Example

A System was proposing a decision tree algorithm for diagnosing faults in an automobile engine. The algorithm uses historical data to identify the symptoms of the fault and predict the possible causes of the fault. The authors test the algorithm on a dataset of engine fault information and demonstrate its effectiveness in diagnosing engine faults with high accuracy [2][5][7][10].

Another system proposed an improved Long short term memory Neural Network for diagnosing faults in a car engine[9]. The system was test on a dataset of car engine fault information and demonstrate its effectiveness in diagnosing faults with high accuracy.

**Chapter 3**

# Specification - (SRS)

# Introduction

## Scope of this document

The suggested system has an easy-to-use chat interface that allows users to interact with it. The user has two options: input symptoms they are now experiencing (Smell, feel etc.) or symptoms where exactly the problem (Engine, Exhaust etc.), and when the user inputs his symptom, the chatbot will predict the diagnose through the symptoms.

To prepare the dataset for machine learning model training, the system transformed the categorical values into an appropriate structured numerical dataset. The System was trained with decision tree with training accuracy 100% and 98.3% testing overall, and Based on [8] which is similar system, The decision tree classifier has been the main classifier employed in the system.

NLP is used to do tokenization which involves breaking the sentence up into words, converting the collection of words into lower case and eliminating stop words Stop words are eliminated, and then stemming is used to break down the remaining words into their root form. On the corpus that has been analyzed, the system extracts keywords.

The algorithm determines if the keywords it has acquired match symptoms the user is describing or a car malfunction he is now experiencing. The defined as the fact diagnosis if the keyword terms correspond to user symptoms. For this, a trained machine learning model is employed. There is a cutoff point of four user symptoms for better diagnosis prediction. Because many diagnoses have similar symptoms, the prediction's accuracy will be lower if a user submits less than four symptoms. Thus, a four has been chosen as the barrier. The chatbot prompts the user to add more symptoms if it finds less symptoms than the target value.

NLP text pre-processing

NLP used to let the user have easy experience with the application. The user enters symptom to be pre-processed and extracting keywords from it. Pre-processing methods – tokenization[6], stemming[6], TF-IDF[6] and Cosine Similarity[6]

Is used to produce a suitable diagnose for the user symptoms.

1. *Tokenization:* Case-folding is the process of converting user-inputted text to lower case. Raw text from the user is tokenized into a Bag of words. Sequentially, the keywords that text contained will be separated. Each word may be handled and examined independently via tokenization. Once all punctuation has been removed, the last bag of words is gained.
2. *Stop words removal:* Stop words like "a," "an," "it" etc. are eliminated from the bag of words produced from the preceding pre-processing phase in order to extract relevant keywords. Stop words must be removed since they consume critical space and time during pre-processing.
3. *Stemming:* The root form of each word is then formed by deleting suffixes or prefixes from the bag of words through an iterative process. In NLP, this procedure is referred to as stemming. The created tokens serve as the stemmer's input.
4. *TF-IDF:* A measure of a term's frequency in a phrase is called term frequency (TF). Inverse Document Frequency (IDF) determines a term's relevance by calculating how uncommon it is across the document. Combining the TF and IDF values gives the term in the document its weight.

X = tf \* idf (1)

tf = number of times term occurred in a sentence

idf = inverse document frequency

The final vector is created by calculating the weight of each phrase from the user's input using the formula above. Similar to this, another vector is created following the same pre-processing and weighting.

1. *Sentence similarity:* The distance between two vectors created by two sentences indicates how similar they are to one another. The distance between the vectors is calculated using the cosine similarity approach. The user receives the appropriate text as a chatbot response if the cosine angle between the two sentence vectors is greater than 0, else the user is prompted to input the required symptoms.

## Overview

To diagnose typical car problems, the system will use a decision tree algorithm that has been trained using past symptoms, and it will integrate the Google Maps API to give consumers relevant details regarding their symptoms. The suggested system seeks to provide a user-friendly and accessible tool for car problem detection without requiring specialised knowledge. It is aimed at car owners and unskilled mechanics.

In addition to describing the design and implementation of the suggested system, the document provides background information on the topic of car defect diagnostics and associated research. In order to produce relevant results for user queries, the system uses NLP text pre-processing techniques such tokenization, stemming, TF-IDF, and cosine similarity.

# General Description

## Product Functions

The chatbot must be trained well by the NLP model to understand the user input, The chatbot should be able to diagnose by using decision tree algorithm the problem based on input from the user , the diagnoses must to be accurate with its description and inspection list. The system should be light, speed, simple that anyone could use it simply

## Similar System Information

There are various system for car fault detection By using On-board diagnoses which takes the car fault codes and diagnose based on it, it is a very good approach and accurate as well but it demands hardware device and costs price[2][7][10],

Another Existing system using recurrent neural network to diagnose [9], comparing decision tree with recurrent neural network results, both models almost identical, but the decision tree is faster, lighter and quicker. Which is very useful to our chatbot to reply to the user in most optimized way, system using chatbot and diagnoses the car well like (MechanicPal) but it isn’t for free.

## User Characteristics

The user characteristics would be the car owners, mechanics, and automotive lovers who are interested to diagnose car faults, and to satisfy the users with their concerns, The expected expertise with software systems and the application domain may vary among the different user groups.

## User Problem Statement

The user will have difficulty identifying faults in car and the user will need accurate answers for all his symptoms. In addition, the user is not familiar with technical terms and jargon used by mechanics, which makes it difficult to describe the symptoms accurately. Moreover, users often don’t have time to visit the mechanic in person, which makes it challenging to get the necessary help in a timely manner. In the last, the user needs a system that can identify symptoms in plain language, provide accurate diagnoses, and can be accessed remotely through a user-friendly interface.

## User Objectives

The user objectives is to receive a quick and reliable diagnosis of car faults, access the diagnostic system remotely, through a user-friendly interface, without the need for in-person visits to a mechanic, system have confidence in the accuracy of the diagnostic system, and to trust that the system will provide reliable recommendations for repairs or next steps, and to save time and money by avoiding unnecessary visits to a mechanic, or by being able to quickly identify and address car faults before they become more serious issues.

## General Constraints

The car fault diagnosis developer will need to make sure on some constraints and requirements, as like speed, hardware platforms, user interface design, security and privacy, and data availability. The system should provide quick and accurate diagnoses based on historical symptoms and be optimized for a range of hardware platforms. The user interface should be intuitive and easy to use, accommodating users with varying levels of expertise. Security and privacy measures should be in place to protect user.

# Functional Requirements

Function Name: Natural Language Processing

Description: The chatbot should be able to prompt the user to provide information on the symptoms they are experiencing with their car.

Criticality: High

Technical issues: The chatbot needs to be able to understand and interpret user input to determine the appropriate questions to ask next.

Risks: There may be a risk of misinterpreting user input, leading to inaccurate diagnoses.

Dependencies with other requirements: This requirement is a pre-condition for Requirement #2. Pre-condition: The chatbot should be available and ready to receive input.

Post-condition: The user has provided information on their car's symptoms.

Function Name: Decision Tree.

Description: The chatbot should be able to diagnose the problem based on historical data.

Criticality: High.

Technical issues: The chatbot needs accurately match symptoms to diagnose the most likely issue.

Risks: Misdiagnosis could lead to the user taking incorrect actions to resolve the problem or even cause more damage to the car.

Dependencies with other requirements: This requirement depends on the previous requirement #1.

Pre-condition: The user has provided information on their car's symptoms.

Post-condition: The chatbot has diagnosed the problem.

Function Name: Provide Detailed Information.

Description: The chatbot should be able to provide the user with detailed information on the problem and how to resolve it.

Criticality: High

Technical issues: The chatbot needs to provide accurate information on the diagnosed problem and its resolution.

Risks: The chatbot may provide inaccurate or outdated information, leading to incorrect or ineffective resolution of the problem.

Dependencies with other requirements: This requirement is a pre-condition for Requirement #4. Pre-condition: The chatbot has diagnosed the problem.

Post-condition: The user has access to detailed information on the problem and how to resolve it.

# Interface Requirements

## User Interfaces

User Interface: the user interface should interact with the user to take inputs ” symptoms” through python implementation and return output “diagnoses” through flutter (see Fig. 1).



### GUI

The System GUI is simple to use to let the user get the most friendly interface to diagnose his car, as the user is fraustrated for his car and doesn’t need complicated application, the system provides an 3 pages (Login,Register and home page) to let the user get his diagnosis in quickest way possible.

* Welcome screen with introduction to the chatbot and instructions for use
* Text input field for the user to enter their symptoms
* Follow-up questions from the chatbot to clarify the user's symptoms
* Diagnosis screen with the chatbot's recommended solution to the user's car problem
* Home screen with options to diagnose car problems, locate nearby mechanics, and access saved car information
* Diagnose car problems screen with text input field for symptoms and chatbot interface
* Locate nearby mechanics screen with Google Maps API integration

## Hardware Interfaces

Python, Android Studio for implementing Flutter, Visual Studio code to implement the models by using python and Jupyter Notebook Extension.

# Performance Requirements

The model built in 0.7sec.

# Design Constraints

* The System should be developed using open-source languages and tools which is frequently updated.
* The system shall adhere to the overall design of the car diagnostic application.
* The system should be delivered within the given project timeline.

## Standards Compliance

Data collection.

# Other non-functional attributes

## Security

Intelligent Car Diagnosis Application must to secure the user data from illegal access or manipulation, the chatbot system should guarantee secure communication between the client and the server.

* 1. **Binary Compatibility**

The system is compatible with different operating systems by using flutter, dart for mobiles application and python as a back end.

## Reliability

The chatbot is available 24/7 with minimum downtime to provide uninterrupted service to the users.

* 1. **Maintainability**

Easy to maintain and update to ensure efficient system management and timely bug fixes.

## Portability

The system can be used from several Operating Systems (IOS,Android etc.).

* 1. **Extensibility**

The system is very flexible and open-sourced that allowing for future updates and enhancements as required. This includes modular code design, use of scalable architecture, and the ability to add new features and functionalities without disrupting the existing system.

## Re-usability

The proposed system must be created to enhance the software code's use across various projects and modules. By sticking to coding best practises and software engineering principles like abstraction, modularity, and encapsulation, the code's reusability will be made possible. The system must also offer a set of APIs that other projects can apply to access the functionality it offers. These APIs must be well-documented and simple to use in order for other developers to integrate them into their applications w

* 1. **Application Affinity/Compatibility**

compatible with other applications and services to ensure smooth integration is very important.

## Resource Utilization

To ensure high performance and lower the chance of system slowdowns or crashes, the system should be built to make effective use of system resources like CPU and memory.

* 1. **Serviceability**

The system should be designed to be serviceable, allowing for quick and easy maintenance, repair, and updating of the system.

# Preliminary Object-Oriented Domain Analysis

**User:** represents a user of the system, who interacts with the chatbot to diagnose their car's problem.

**Symptom:** represents a symptom that the user can input into the chatbot, which will be used to diagnose the car's problem.

**CarProblem:** represents a car problem that the chatbot can diagnose based on the symptoms provided by the user.

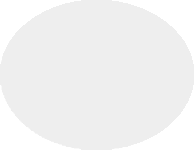
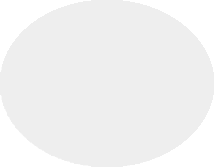
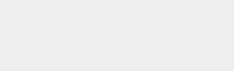
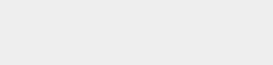
**DecisionNode**: represents a node in the decision tree that the chatbot uses to diagnose the car's problem.

**Chatbot**: represents the chatbot system, which includes the decision tree, symptom input, and car problem diagnosis.

**Application:** represents the Flutter application that contains the chatbot system and the GoogleMapsAPI.

# Operational Scenarios

Fig 3.: Working of the Intelligent Car Diagnosis Application.



User

User opened the app

GUI User Interface

iRepair application

Machine Learning

Model Building

Response to User

Query Processing

Learning Data

Chatbot Interface

Fig 3.2: Working of the Intelligent Car Diagnosis Application from end-to-end processes

**Chapter 4**

# Software Design

# Introduction

## User Interfaces

The purpose of this Intelligent Car Diagnosis Application Software Design Document is to outline the architecture and system design of the Car Diagnosis Chatbot flutter, Dart and python, and intended functionality, interfaces, and interactions with the end-users.

## Scope

The Intelligent Car Diagnosis Application we are developing is a friendly flutter mobile application, it contains chatbot which helps the user to diagnose their the car problems. With the use of the Natural Language processing Algorithm (NLP). It will allow users to input various symptoms they are experiencing with their car, and the NLP Module will use decision tree algorithms to determine the possible problems and provide potential solutions to the user. The chatbot will be developed using Python and integrated with a Flutter mobile application that contains Google Maps API.

* + 1. **Goals and Objectives:**

The main goal of this project is to provide a user-friendly and effective solution for diagnosing car problems. By using the mobile application system, users can easily open their smartphones and to make it more evolutionary and easier to the user, we designed to implement chatbot to diagnose the user concerns for his car.

The user enters their symptoms and receive a diagnosis quickly and accurately, without the need for expensive diagnostic tools or professional assistance.

* + 1. **The specific objectives:**
* To create a chatbot that is user-friendly, engaging, and able to effectively communicate with users.
* NLP Model is feeded enough with data because it must to be accurate and reliable as we as the decision tree-based algorithm for diagnosing car problems based on user inputs.
* To integrate the chatbot with a mobile application that includes Google Maps API to provide users with nearby repair shops and dealerships.
* To test the chatbot-based diagnostic tool with a sample of real-world car problems to ensure accuracy and usability.
* To collect Raw Data
  + 1. **Benefits:**

The project provides wide range of advantages. I can save users time and money by displacing the need for pricy diagnostic tools and expert assistance with a chatbot-based diagnostic tool. Furthermore, customers will have rapid access to nearby mechanics thanks to the mobile application's connectivity with the Google Maps API.with Google Maps API will provide users with quick access to nearby repair shops and mechanics

## Overview

The Intelligent Car Diagnostic Application's user interface will be created with simplicity and user friendliness in mind. The mobile application will offer other features like Google Maps API to find nearby repair shops, while the chatbot will serve as the main interface for customers to input their symptoms and receive a diagnosis.

## Definitions and Acronyms

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Natural Language Proccessing (NLP) | Used as the primary medium for communicating software  design information. |
| API | An element of a design that is structurally and functionally  distinct from other elements. |

# System Overview

The project aims to create a chatbot for diagnosing car problems using decision trees. The chatbot will take input from the user in the form of symptoms, which will be fed into the decision tree to diagnose the problem with the car. The chatbot will then respond with the appropriate diagnosis.

The chatbot will be implemented using Python and will be linked to a Flutter application that contains Google Maps API. The application will be able to use the Google Maps API to provide location-based services to the user, such as directing them to a nearby mechanic.

The project will involve the creation of a dataset containing categorical data, which will be used to train the decision tree. The dataset will contain four columns: Symptom 1, Symptom 2, Symptom 3, and Car Problem.

The goals of the project are to create a user-friendly chatbot that can diagnose car problems and provide useful information to the user, such as the location of nearby mechanics. The objectives of the project are to collect and preprocess the data, train the decision tree, implement the chatbot, and integrate it with the Flutter application.

The benefits of the project are that it will provide a useful tool for car owners who are experiencing problems with their cars. The chatbot will allow them to quickly and easily diagnose the problem and get information on nearby mechanics, saving them time and money. The project will also provide valuable experience in data preprocessing, machine learning, and software development.

# System Architecture

## Architectural Design



## Data Collection

Data Collection is to collect dataset that includes information about various car problems and their causes. This data can be used to train the decision tree model.

## Data Preprocessing:

The collected dataset may require preprocessing to handle missing values, remove outliers, and convert categorical variables to numerical ones. This step is important to ensure that the data is in a format that can be used to train the decision tree model.

* + 1. **Model Evaluation:**

The trained model is then evaluated on the validation data to measure its accuracy and make any necessary adjustments. The model is considered to be ready for deployment once it has been evaluated and any necessary adjustments have been made.

## Deployment:

The final step is to deploy the trained model in a chatbot system. This involves integrating the decision tree model into a chatbot platform and designing the conversational flow to ask the user questions and provide recommendations based on the model's predictions.

## Maintenance:

Finally, the chatbot system should be regularly updated and maintained to ensure that it continues to perform well and provide accurate results. This may involve updating the dataset, retraining the model, or making any necessary improvements to the chatbot's conversational flow.

Fig 4.1: Architectural Design

## Decomposition Description

## 4.6.1 User Interface Subsystem

**Login Module:** If your login and password are entered successfully, you will be sent to the main interface.

**User Input module**: Collects user input from the chatbot interface and sends it to the Backend Server subsystem.

**Decision tree module:** Receives the symptom from the NLP and the Decision Tree subsystem and passes it to the User Interface subsystem for display.

**Google Maps API module**: Initiates the Google Maps API subsystem to retrieve the user's location and search for nearby mechanics.

**NLP and Decision Tree Subsystem**

NLP module: Extracts relevant symptoms from user input using natural language processing techniques.

Decision Tree module: Uses the extracted symptoms to diagnose the car problem using the decision tree algorithm.

**Google Maps API Subsystem**

**Location module:** the module will direct the user to the nearest mechanic.

**Search module:** if the user opens his GPS, Searches for nearby mechanics using the Google Maps API.

Fig 4.2: Dataflow Diagram of the Intelligent Car Diagnosis Application

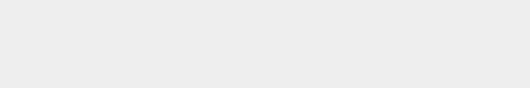
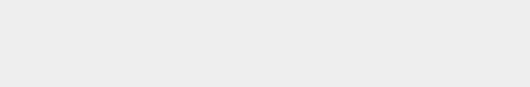
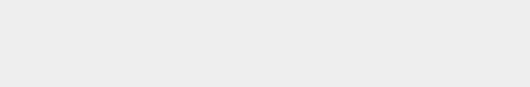
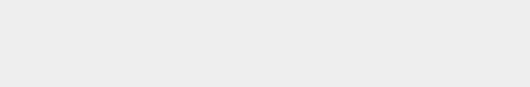
# Data Design

## Data Description

### Data collection

The dataset obtained from Github as it was the main challenge is to collect raw data. Even the strongest engine in the world “ChatGBT” informed that there isn’t dataset contains car fault symptoms. The raw data collected with 34 symptoms only from 3 different Repository During almost 3 weeks. However, there is in the last days new. json dataset contains 8800 rows with 16 features,96 subfeatures,86 sub-sub-features and 6 sub-sub-sub features and 254 unique target . So, it very important to clean the data and data preprocessing.

For supervised learning models, data preprocessing counts [6]. Data integration, data cleaning, data type transformation, and feature selection are examples of common data pre-processing steps.



Predictions

Model Building

Machine Learning

Improve Data Features

Feature Extraction

Cleaning Data

Data Collection

[1] Data cleaning, which entails dividing a feature with multiple part numbers into individual part numbers, removing irrelevant features, and replacing "null" data values;

[2] Data transformation, which converts categorical strings into numerical values; and

[3] Feature selection, which involves assessing the degree to which features are dependent on the target feature and choosing features for data analysis.

Fig 4.3: Processes Model for Intelligent Car Diagnosis Application

Fig 4.4: A Standard Data Analysis Pipeline

### Cleaning Data

One value report record, encoded in strings, contains all of the symptoms associated with a single diagnosis.

We substitute some rows in the dataset that have "null" values with the String value "empty" to distinguish them by part number. The decision tree model is assessed by the system's prediction accuracy. [7].

### Data transformation

The process of converting unwanted data types into original data types is known as data transformation. In the thesis, we employ the one-hot encoder approach to convert nominal data into numerical data.

A common technique for encoding data is the one-hot encoder, also referred to as the one-of-K scheme. One-hot encoding transfers categorical values from a dataset containing multiple categorical values by establishing columns for each categorical value and representing them with the binary values 0, 1, and 0. The binary bit group's valid value is 1, which indicates that the record belongs to a particular category. Value 0, which indicates that the record does not belong to the category, is also invalid. Let us assume,

Let's say that one dataset has s total category values and is a m n matrix dataset. A new dataset with shape m s is created after one-hot encoding. Each categorical value in the new dataset is uniquely represented as one column (feature).

There are 4columns, multiple rows and multiple categorical values in sum. For ex: one-hot encoding, the table is

|  |  |
| --- | --- |
| **Symptom 1** | **Symptom 2** |
| The car shakes | The whole car |
| Smell Gasoline | Only occurs while idling |
| Smell Gasoline | All the time |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Smell Gasoline** | **Smell Gasoline** | **The car shakes** | **All the time** | **Only occurs while idling** | **The whole car** |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |

Table 4.1: Before and after OneHotEncoding

### Machine learning tools - classification

Let L be a random vectors X, Y, X denotes labels and Y is the target. L = (x11, x12, , x1n, y1), (x21, x22, , x2n, y2), , (xm1, xm2, , xmn, yn) (Y). The goal of classification is to choose and train classification models to map attributes X and labels Y, then use the trained model to determine the label for unknown data [9]. A training dataset and a testing dataset are often separated into the dataset. Using the training dataset, the machine learning model picks up information and discovers data patterns. The data patterns that are the learning outcome are assessed for prediction accuracy and put to the test using a test dataset. Ideally, the classification rules will divide data with similar features into the same subset.

A trained model's effectiveness is typically measured by how well it can approximate labels for unobserved data. The majority of the performance evaluation criteria are dependent on the accuracy of predictions. With bias and variance, the performance of a trained model may be evaluated and displayed. The bias is an assessment of the discrepancy between the anticipated and actual values.

### Decision tree

A decision tree is a tool that helps people make and describe decisions by using conditional statements and a tree-like graphic. It is frequently employed to address classification and regression issues. Decision nodes and leafs  make up a decision tree. One root node, internal nodes, and leaf nodes make up the decision nodes. A decision tree's root node and internal nodes stand in for the splitting variable and splitting rules. A tiny subset of observations are found in the leaf nodes, which are also known as terminal nodes. Data with identical predicted labels are organised into identical leaf nodes. One categorization rule is represented by the path leading from the root node to the leaf nodes.

Finding a decision tree's root node is the first step in creating one. Then, internal nodes are found, and the process is finished when no more decisions are needed.

Fig 4.5: Decision tree

Another often used splitting criterion in tree-based classifiers is the Gini Impurity measure. It evaluates the consistency of the data values. The risk of incorrectly selecting one random element labelled is measured using Gini Impurity, which takes into account the distribution of labels for a set of data. The Gini Impurity can be represented by letting pi stand for the likelihood that an element will be given label I and J for the number of labels.

*J*

Gini Impurity (p) = 1 — (p*i*)2

X

*i*=1

Maximum depth is the longest distance possible between the leaf nodes and the root node. Configuring a decision tree's maximum depth is crucial, specifically when a dataset contains a lot of characteristics. A deep decision tree learns from characteristics in order to effectively recognise data patterns. Hence, bias difficulties are infrequent in deep decision trees. On the other hand, because a deep decision tree learns a lot of false information from noisy input, it may produce an overtrained model. Thus, a decision tree can accurately identify data paths for the training dataset but cannot accomplish this for new data, leading to the over-fitting issue.

The use of the decision tree model has a few benefits. The decision tree model is simple to understand and interpret, which is the first benefit. Leaf nodes are the labels, and the routes leading from the root node to them are the rules for making decisions. The same labels are used for data that complies with the same standards. Its second benefit is that there are fewer hyper-parameters to adjust for. A decision tree model is therefore easy to create and visualize. A decision tree may also handle many data formats, which can make data pre-processing easier.

However, there are disadvantages to the decision tree model. The over-fitting issue is the first. Deep decision trees are simple to construct but have the over-training issue and are unable to apply to new data. The second is that the greedy nature of the decision tree method. In Conclusion, The result of data pattern for both the training dataset and the testing dataset may not be discovered by the decision tree, since it constantly selects the current "optimal" characteristics as splitting nodes. That is a common issue with greedy algorithms. The ensemble method can lessen the effects of the model's stated disadvantages.

(a) (b)

1 2 3 **1** (1) **2** (2) 3 (5)

2.1 (3) **2.2** (4)

2.1 2.2

(1) (2) (3) (4) (5)

*vi* = [1, 1, 0, 1, 0]

Fig. 4.6: A compact hierarchy. Class label names indicate where they fall in the hierarchy; for example, "2.1" is a subclass of "2". (b) The group of classes 1, 2, and 2, shown as a vector and designated in bold in the hierarchy.

## Data Dictionary

Table 4.2: Data Description of the Intelligence diagnoses system

|  |  |
| --- | --- |
| **Module** | **Description** |
| Login Module | Username and password if correct the user will be routed to the interface |
| Main Interface | The main interface consists of five respective modules. The menus are as follows: registration,  change password,Google Maps,. |
| Registration | This module for user registration enables users to keep track of his car faults in a database. These documents assist in maintaining the user information. It includes the historical chat, phone number, and date.. |
| Change Password | is the window that users may utilise to change their existing password by entering their username, current password, new password, and new password confirmation information. When the modify button is pressed, the entries can be stored into the database, and the cancel button ends the procedure as normal.. |
| Fault/Knowledge database | This program enables the users to add or modify the entries in the knowledge database. It allows users to add or modify car type, problem,  symptoms, causes and solutions. |
| User | By selecting the user module, a new user may be added to the database. The form includes fields for the person's complete name, username, password, and confirm password. Both the create and cancel buttons store the entry in the database, respectively.. |
| Diagnosis | This module handles the car's diagnostics in order to find the problem and show the resulting report. It includes a decision tree model for entering the car's symptoms. |

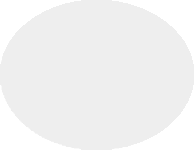
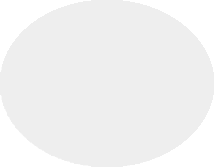
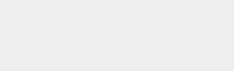
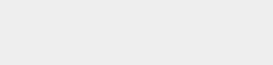
# Dataset Design

Table 4.3: The Dataset view used for training of the Intelligent Car Diagnosis Application.

# Human Interface Design

## Overview of User Interface

Fig 4.7: Real time work flow and interface of the Intelligent Car Diagnosis Application with human.



User

User opened the app

GUI User Interface

iRepair application

Machine Learning

Model Building

Response to User

Query Processing

Learning Data

Chatbot Interface

Fig 4.8: Real time work flow and interface of the Intelligent Car Diagnosis Application with human.

Fig 4.9: Flowchart for User Module

## Screen Images

## 

## Screen Objects and Actions

Google Maps API to suggest the nearly mechanic and the chatbot for diagnosing the car faults.

# Bibliography

1. R.S. Michalski, J.G. Carbonell, T.M. Mitchell,”*Machine learning: An artificial intelligence approach*” Springer Science & Business Media, pp.92, ,2013.
2. Abubakar, S.; Ahmad, I.S.; Gambo, F.L.; Gadanya, M.S. “A Rule-Based Expert System for Automobile Fault Diagnosis.” Int. J.Perceptive Cogn. Comput. pp.20–25, ,2021,
3. R. Jegadeeshwaran and V. Sugumaran, “Fault diagnosis of automobile hydraulic brake system using statistical features and support vector machines,” Mech. Syst. Signal Process., vol. 52–53,
4. G. Wang and S. Yin, “Data-driven fault diagnosis for an automobile suspension system by using a clustering based method,” J. Franklin Inst., vol. 351, no. 6, pp. 3231–3244, 2014, doi: 10.1016/j.jfranklin.2014.03.004.
5. Al Zeyadi, M., Andreu Perez, J., Hagras, H., Royce, C., Smith, D., Rzonsowski, P., et al.: Deep learning towards intelligent vehicle fault diagnosis. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE, Piscataway (2020)
6. S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, “Data preprocessing for super- vised leaning,” International Journal of Computer Science, vol. 1, no. 2, pp. 111– 117, 2006.
7. Hu, L. (2019). Smart Fault Tracing: Learning with Diagnostic Data for Predicting Vehicle States pp. 8-15. Uppsala Universitet. <https://uu.diva-portal.org/smash/get/diva2:1369705/FULLTEXT01.pdf>
8. S. Badlani, T. Aditya, M. Dave, and S. Chaudhari, “Multilingualhealthcare chatbot using machine learning,” in 2021 2nd InternationalConference for Emerging Technology (INCET). IEEE, may 2021.
9. M. Youssef and H. Ibrahim, "Automotive engine fault detection and isolation using LSTM for model-based residual sequence classification," 2021 18th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), Mexico City, Mexico, 2021, pp. 1-6, doi: 10.1109/CCE53527.2021.9633053.
10. J. Hu, C. Liu, F. Yan and Z. Zhang, "The development of vehicle diagnostic system based on Android platform," 2015 International Conference on Connected Vehicles and Expo (ICCVE), Shenzhen, China, 2015, pp. 42-47, doi: 10.1109/ICCVE.2015.7.
11. Zhai, J., & Zhang, W. (2018). Fault diagnosis of an automobile engine using decision tree algorithm. In 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC) (pp. 164-168). IEEE.