

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 complicated process that demands a high level of knowledge and skills. For this reason, automobile users require skilled automobile technicians for diagnosing a fault detected in their automobile and for maintenance. However, some faults are minor and will not require the services of skilled mechanics. An automated system that can help automobile users in diagnosing and fixing common automobile failures will be of paramount importance. Expert systems are widely used in such fields as medical and trading for providing advice from a knowledge base database. However, fewer studies have used expert systems for automobile fault diagnosis that can be used by automobile users. The aim of this research is to develop an expert system for automobile fault diagnosis for automobile users. The knowledge base was acquired through interview and observation. The system was evaluated, and the results show that an expert system can be used for automobile fault diagnosis by automobile users. This will enable automobile users to be able to identify the fault in their automobile, fix it if it is a minor fault or take it to the necessary technician where necessary.

**Acknowledgments**

Foremost, I own my deepest gratitude to my supervisor, Dr. Moataz Samy, who helped me to get the opportunity to carry out my Graduation project, and gave me valuable constructive suggestions. Besides my supervisor, I am very grateful to Prof. Islam ElShaarawy, for his continuous support and the efforts he has paid to guide me throughout the research and always giving solution in anytime. He was always willing to help when I had questions to discuss. 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 another fields as well. Moreover, My huge thanks for Prof. Ayman Ezzat Prof. Hesham Mansour for their continuous help for the whole committee. Lastly, I would like to sincerely thank all the people who has supported me during the completion of my Graduation Project.

# Contents

Contents

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

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

[Contents 4](#_Toc128315335)

[List of Figures 4](#_Toc128315336)

[Introduction 5](#_Toc128315337)

[1.1 Introduction guide lines 5](#_Toc128315338)

[Background 6](#_Toc128315339)

[2.1 General Guidelines 6](#_Toc128315340)

[2.2 Example 7](#_Toc128315341)

[Specification - (SRS) 8](#_Toc128315342)

[Introduction 8](#_Toc128315343)

[3.1 Purpose of this document 8](#_Toc128315344)

[3.2 Scope of this document 8](#_Toc128315345)

[3.3 Overview 8](#_Toc128315346)

[3.4 Business Context 8](#_Toc128315347)

[General Description 9](#_Toc128315348)

[3.5 Product Functions 9](#_Toc128315349)

[3.6 Similar System Information 9](#_Toc128315350)

[3.7 User Characteristics 9](#_Toc128315351)

[3.8 User Problem Statement 9](#_Toc128315352)

[3.9 User Objectives 9](#_Toc128315353)

[3.10 General Constraints 9](#_Toc128315354)

[Functional Requirements 9](#_Toc128315355)

[Interface Requirements 10](#_Toc128315356)

[3.11 User Interfaces 10](#_Toc128315357)

[3.11.1 GUI 11](#_Toc128315358)

[3.11.2 CLI 11](#_Toc128315359)

[3.11.3 API 11](#_Toc128315360)

[3.11.4 Diagnostics or ROM 11](#_Toc128315361)

[3.12 Hardware Interfaces 11](#_Toc128315362)

[3.13 Communications Interfaces 11](#_Toc128315363)

[3.14 Software Interfaces 11](#_Toc128315364)

[Performance Requirements 12](#_Toc128315365)

[Design Constraints 12](#_Toc128315366)

[3.15 Standards Compliance 12](#_Toc128315367)

[Other non-functional attributes 12](#_Toc128315368)

[3.18 Security 12](#_Toc128315369)

[3.20 Reliability 12](#_Toc128315370)

[3.22 Portability 12](#_Toc128315371)

[3.24 Re-usability 12](#_Toc128315372)

[3.26 Resource Utilization 12](#_Toc128315373)

[Preliminary Object-Oriented Domain Analysis 13](#_Toc128315374)

[Operational Scenarios 14](#_Toc128315375)

[Design 15](#_Toc128315376)

[Introduction 15](#_Toc128315377)

[4.1 User Interfaces 15](#_Toc128315378)

[4.2 Scope 15](#_Toc128315379)

[4.3 Overview 16](#_Toc128315380)

[4.4 Definitions and Acronyms 16](#_Toc128315381)

[System Overview 16](#_Toc128315382)

[System Architecture 17](#_Toc128315383)

[4.5 Architectural Design 17](#_Toc128315384)

[4.5.1 Data Collection 17](#_Toc128315390)

[The first step is to collect a dataset that includes information about various car problems and their causes. This data can be used to train the decision tree model. 17](#_Toc128315391)

[4.5.2 Data Preprocessing: 17](#_Toc128315392)

[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. 17](#_Toc128315393)

[4.5.4 Deployment: 17](#_Toc128315394)

[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. 17](#_Toc128315395)

[4.5.5 Maintenance: 17](#_Toc128315396)

[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. 17](#_Toc128315397)

[4.6 Decomposition Description 19](#_Toc128315398)

[4.6.1 User Interface Subsystem 19](#_Toc128315399)

[4.7 Design Rationale 19](#_Toc128315400)

[5 Data Design 19](#_Toc128315401)

[5.1 Data Description 19](#_Toc128315402)

[4.1.1 Data collection 19](#_Toc128315403)

[4.1.2 Cleaning Data 21](#_Toc128315404)

[4.1.3 Data transformation 21](#_Toc128315405)

[4.1.4 Feature selection 22](#_Toc128315406)

[4.1.5 Machine learning tools - classification 25](#_Toc128315407)

[5.1.1 Decision tree 26](#_Toc128315408)

[5.2 Data Dictionary 28](#_Toc128315409)

[Dataset Design 28](#_Toc128315410)

[6 Human Interface Design 29](#_Toc128315411)

[6.1 Overview of User Interface 29](#_Toc128315412)

[6.2 Screen Images 30](#_Toc128315413)

[30](#_Toc128315414)

[6.3 Screen Objects and Actions 30](#_Toc128315415)

[Bibliography 31](#_Toc128315416)

# List of Figures

[Fig 1: Working of the UCB from end-to-end processes 14](#_Toc128315316)

[Fig 4.1 : Processes Model for Intelligence car diagnostic Application 20](file:///E:\MSA\Fall%202023\Big%20Data\GP\fall-2022-spring-2023-october-university-for-modern-sciences-and-arts-cs-gp-thesis-bsc-template.docx#_Toc128315317)

[Fig 4.2: A Standard Data Analysis Pipeline 20](file:///E:\MSA\Fall%202023\Big%20Data\GP\fall-2022-spring-2023-october-university-for-modern-sciences-and-arts-cs-gp-thesis-bsc-template.docx#_Toc128315318)

**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 the design and implementation of a cross-platform application with open-source languages for implementing Machine Learning (ML) Algorithms . Machine learning-based, Expert systems, statistical-based approaches have been applied for car fault diagnosis [2-8]. Similarly, decision trees has also been proposed for car fault diagnosis. Expert systems are reliable, accurate and cost effective [2-3] and . However, due to the ever-changing specifications, the knowledge base and approaches which used but Expert Systems in car fault diagnosis also needs to be constantly improved. 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 an car and covers general faults in specific automobile models, and as such different from existing systems that mostly targets trained personnel and covers specific automobile parts like gearbox and brake systems. However, has the following contributions: identified the common faults in specific automobiles using qualitative methods and proposed an expert system that can help drivers and inexperienced mechanics in diagnosing common faults found in an automobile.

## Example

This is Example for writing some related work you can extract them from your Proposal and SRS

The study of gesture recognition with a presentation viewer application was shown in [1]. They show an active region for starting and ending gesture interaction. Also, they point out that gestures can be useful in crowded or noisy situations, such as in a stock exchange or manufacturing environment. Head and hand gestures have been used for limited interactions as demonstrated in by Keates et al. [1]. They discussed the problem of learning gestures and showed the importance of customization. Kurze et al. [2] presented penalization of multi- modal applications as a design approach. They focus on implicit and explicit customization of systems according to a user’s preferences. Kawsar et al. [2] presented customizing the proactive applications preferences in a ubiquitous environment. They present customization in many levels of artifact, action, interaction, and timing preferences

**Chapter 3**

# Specification - (SRS)

# Introduction

## Purpose of this document

Describes the purpose of the document, and the intended audience.

## Scope of this document

The proposed system consists of a user-friendly chat interface through which a user can communicate with the system. The user can either enter symptoms which he is experiencing or enter some health-related queries. Depending on the user input, the chatbot will predict the disease or provide relevant information about his queries.

The system has pre-processed the disease dataset, converting the categorical values into a suitable structured numerical dataset for training of the Machine Learning model. The system was trained with decision tree with training accuracy 100% and 98.3% testing overall, and Based on [8] The system has trained and compared five different Machine Learning classification models - SVM, KNN, Decision Tree, MNB and Random Forest Classifier - and got the best accuracy using the decision of 98.43%. Thus, the primary classifier which has been used in the system is the decision tree Classifier.

Initially, the user must select their preferred language of communication. Currently, the system supports three languages,i.e., English, Hindi and Gujarati. The next step for the user is to select their preferred mode of communication, i.e., voice or text.

The input entered by the user will first be converted into text by the system if the user is communicating via speech. The speech to text conversion has been accomplished using the SpeechRecognition library available in python. The input text will be converted into English if it is in some other language. This language translation has been achieved using the Googletrans python library. The translated input is passed on to the NLP module.

The NLP module performs tokenization which is splitting the sentence into words, it then converts the bag of words into a lower case format, and later removes the stop words

i.e. commonly used words. After the removal of stop words, those words are converted into their root form using stemming. The system performs keyword extraction on the processed corpus.

The system checks if the obtained keywords correspond to a health-related user query or they correspond to symptoms that the user is experiencing. If the keywords are user symptoms, the system performs disease diagnosis. The trained Machine Learning model is used for this purpose. For user symptoms, there is a threshold of four symptoms for better disease prediction. If a user enters less than four symptoms, the accuracy of the prediction will be less as many diseases have common symptoms. Thus, a threshold of four has been decided. If the chatbot receives a lesser number of symptoms than the threshold value, it prompts the user to enter more symptoms. When the disease has been predicted,

indicates that the disease does not exhibit the corresponding symptom.

*C. NLP text pre-processing*

NLP is used to give users a human chat-like and easy communication experience. The user input text has to be pre-processed so as to derive symptoms and proper keywords from it. Pre-processing methods – tokenization[8], stemming[6], TF-IDF[12] and Cosine Similarity[13] have been used to generate an appropriate response for the user query.

the result is sent to the user along with the corresponding disease description and precautions or steps which the user can take. The disease description and precaution information is obtained from the dataset. If the system is not able to extract any symptoms from the user input, it classifies the input as a health-related query. The system then applies the TF-IDF and Cosine Similarity techniques to find the most appropriate response to the user query from the knowledge database that has been provided to it. The final response is then converted back to the user’s preferred language using the Googletrans python library. The output is presented to the user as text if their preferred mode of communication is text. Otherwise, if the user’s preferred mode of communication is voice, then the system reads out the output to the user with the help of the gTTS and playsound libraries. Fig. 1 describes the proposed system in detail.

*A. Data Preprocessing*

The dataset obtained from Kaggle [9] was raw data with 4920 records and 41 unique diseases, including the mapping of disease with the corresponding symptoms. Dataset also included a description of each disease and corresponding precautions. The dataset was checked for inconsistencies and then the count vectorizer technique was applied to convert unstructured categorical data into structured numerical data. The final dataset consists of each column as a unique symptom and row with a disease. If a symptom belongs to any disease, then the corresponding cell has a value of 1, otherwise the value is 0. Thus, for any disease- symptom pair, a value of 1 indicates the presence of that particular symptom for that disease while a value of 0

1. *Tokenization:* User input text is first converted into lower case which is known as case-folding. The lower case user input is raw text which is transformed into a bag of words using tokenization. This will separate the text into the terms that compose it. Tokenization is useful for dealing with and analyzing each word independently. Further, all the punctuations are removed and the final bag of words is obtained.
2. *Stop words removal:* In order to extract important keywords, stop words such as ‘a’, ‘an’, ‘the’, etc are removed from the bag of words obtained from the previous pre-processing step. Removal of stop words is necessary as they take valuable pre-processing time and space.
3. *Stemming:* The bag of words is then iterated and the root form of each word is generated by removing suffixes or prefixes. This process is known as stemming in NLP. The input of the stemmer is the tokens that are generated. Here, the system is using the Porter Stemmer algorithm which gives the best output as compared to the other stemmers.
4. *TF-IDF:* Term Frequency (TF) is an indicator of the number of occurrences of a term in the sentence. Inverse Document Frequency (IDF) measures the uniqueness and hence the significance of a given term in the document. Weight of the term in the document is obtained by combining TF and IDF values.

*Wi = tf \* idf* (1)

*tf* = number of times term occurred in a sentence

*idf* = inverse document frequency

Using the above formula, weight of each term from the user input is calculated and the resultant vector is obtained. Similarly, after the same pre-processing and weight calculation of terms in the list of questions database, another vector is formulated.

1. *Sentence similarity:* Similarity between two sentences is the distance between two vectors formed by those sentences. Cosine similarity[13] method is used to find the distance between those vectors. If the cosine angle between two sentence vectors is greater than 0, corresponding text is returned to the user in the form of chatbot response else the user is asked to enter proper information.

*D. Classification Algorithms*

The system has compared the following classification algorithms for disease classification:

1. Random Forest Classifier[14]
2. K-Nearest Neighbors(KNN)[15]
3. Support Vector Machine(SVM)[5]
4. Decision Tree[16]
5. Multinomial Naive Bayes(MNB)[17]

The above classifiers are supervised learning algorithms. Supervised learning is a type of machine learning where the models are trained by example. In this case, the training set consists of attributes that are mapped to labels or target values. The algorithm correlates the inputs and the outputs in the training data and learns from them.

The Random Forest Classifier is an ensemble of Decision Trees. It can be compared to a forest. A forest having more trees is said to be more robust. Similarly, a random forest makes use of multiple decision trees. It trains the Decision trees on subsamples of the original training set. This improves the accuracy of the Random Forest Classifier and reduces over-fitting.

KNN works on the idea of similar things existing in close proximity. The KNN algorithm uses the similarity between the features for the purpose of classification. The new data point is assigned a value based on its similarity with the training data points. It is one of the simplest classification algorithms that is used.

The main aim of the SVM algorithm is to discover a hyperplane in an n-dimensional space. For any classification problem, multiple hyperplanes are possible. The hyperplane which is chosen must have the maximum margin, i.e. the distance between the data points of the different classes must be maximized. The hyperplanes act as boundaries between different classes.

A Decision Tree is a classification tool that is in the form of a tree. The training data is continuously split according to the attributes.

The MNB algorithm is a probabilistic learning method. The Bayes theorem is based on the simple assumption that every feature is independent of the other features. It uses this assumption to classify the given sample.

## Overview

Provides a brief overview of the product defined as a result of the requirements elicitation process.

## Business Context

Provides an overview of the business organization sponsoring the development of this product. This overview should include the business’s mission statement and its organizational objectives or goals.

# General Description

## Product Functions

Describes the general functionality of the product, which will be discussed in more detail below.

## Similar System Information

Describes the relationship of this product with any other products. Specifies if this product is intended to be stand-alone, or else used as a component of a larger product. If the latter, this section discusses the relationship of this product to the larger product. This is how you can [1] a document.

## User Characteristics

Describes the features of the user community, including their expected expertise with software systems and the application domain.

## User Problem Statement

This section describes the essential problem(s) currently confronted by the user community.

## User Objectives

This section describes the set of objectives and requirements for the system from the user’s perspective. It may include a ”wish list” of desirable characteristics, along with more feasible solutions that are in line with the business objectives.

## General Constraints

Lists general constraints placed upon the design team, including speed requirements, industry protocols, hardware platforms, and so forth.

# Functional Requirements

* The chatbot must be trained well by the NLP model and prompt the user to provide information on the symptoms they are experiencing with their car.
* The chatbot should be able to diagnose by using decision tree algorithm the problem based on the symptoms provided by the user.
* The chatbot should be able to provide the user with detailed information on the problem and how to resolve it.
* The chatbot should be able to provide links to resources that the user can use to resolve the issue.

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.

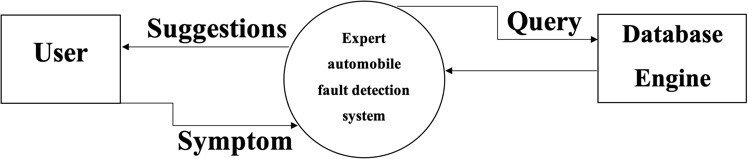
Post-condition: The user has access to detailed information on the problem and how to resolve it.

# Interface Requirements

This section describes how the software interfaces with other software products or users for input or output. Examples of such interfaces include library routines, token streams, shared memory, data streams, and so forth.

## 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

Describes the graphical user interface if present. This section should include a set of screen dumps or mock-ups to illustrate user interface features. If the system is menu-driven, a description of all menus and their components should be provided.

### CLI

Describes the command-line interface if present. For each command, a description of all arguments and example values and invocations should be provided.

### API

Describes the application programming interface, if present. For each public in- terface function, the name, arguments, return values, examples of invocation, and interactions with other functions should be provided.

### Diagnostics or ROM

Describes how to obtain debugging information or other diagnostic data.

## Hardware Interfaces

Describes interfaces to hardware devices.

## Communications Interfaces

Describes network interfaces.

## Software Interfaces

Describes any remaining software interfaces not included above.

# Performance Requirements

Specifies speed and memory requirements.

# Design Constraints

Specifies any constraints for the design team using this document.

## Standards Compliance

* 1. **Hardware Limitations**
  2. **others as appropriate**

# 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**

## Re-usability

* 1. **Application Affinity/Compatibility**

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

## Resource Utilization

* 1. **Serviceability**
  2. **others as appropriate**

# 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

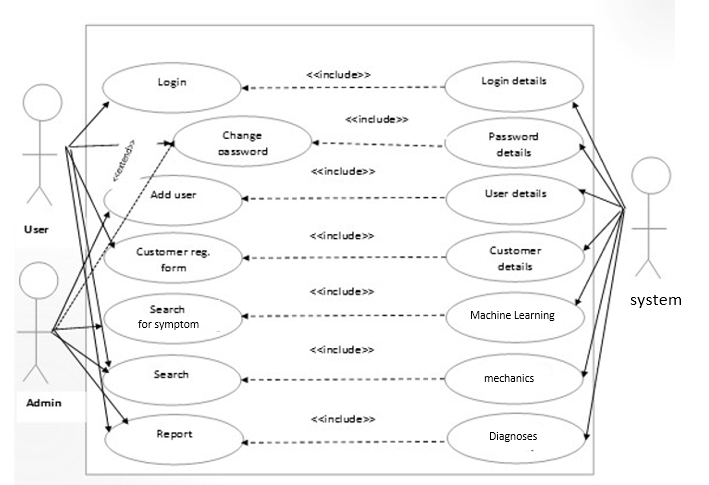
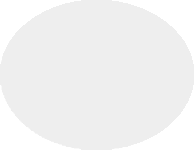
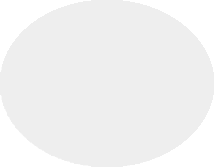
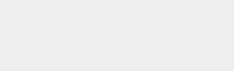
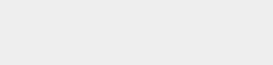


Fig : Working of the UCB from end-to-end processes



User

User opened the app

GUI User Interface

iRepair application

Machine Learning

Model Building

Response to User

Query Processing

Learning Data

Chatbot Interface

Fig 2: Working of the UCB from end-to-end processes

**Chapter 4**

# 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

Provide an overview of this document and its organization.

## 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

## The first step is to collect a 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.

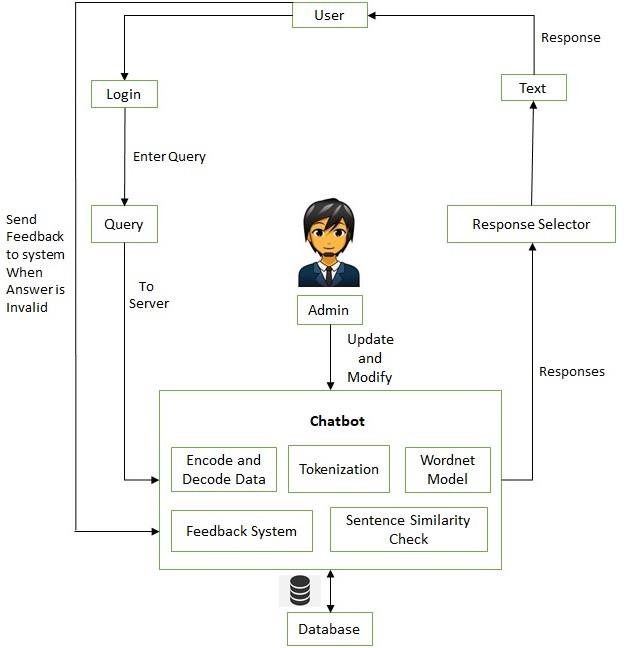


Fig 1: Architectural Design

## Decomposition Description

## 4.6.1 User Interface Subsystem

**Login Module:** This consists of entries for username and password in which if entered correctly, takes you into 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.

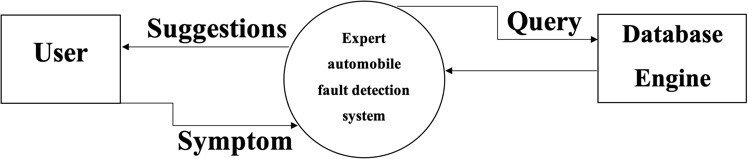


Fig. 1 Dataflow Diagram of the Intelligent Car Diagnosis Application

## Design Rationale

Discuss the rationale for selecting the architecture described in 3.1 including critical issues and trade/offs that were considered. You may discuss other architectures that were considered, provided that you explain why you didn’t choose them.

# Data Design

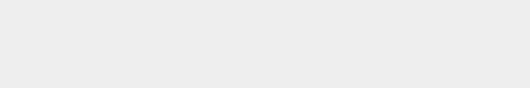
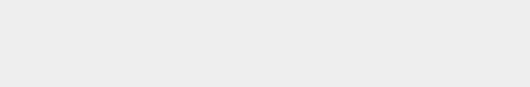
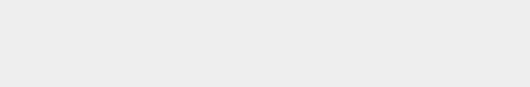
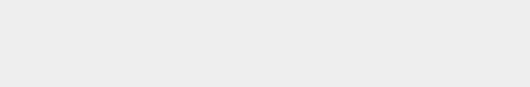
## Data Description

## 5.1.1 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 the raw data collected with 34 symptom 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. So it very important to clean the data and data preprocessing.

Data preprocessing is significant for supervised learning models [[6](#_bookmark82)]. Typical data pre- processing process includes data integration, data cleaning, data type transformation, and feature selection.

Fig 4.1 : Processes Model for Intelligence car diagnostic Application



Predictions

Model Building

Machine Learning

Improve Data Features

Feature Extraction

Cleaning Data

Data Collection

1. data cleaning, where we split the feature that contains multiple part numbers into individual part numbers, remove unrelated features and replace ”null” data values;
2. data transformation, where we transform categorical strings into numerical values;
3. feature selection, where we measure the dependency of features with the target feature and select features for data analysis.

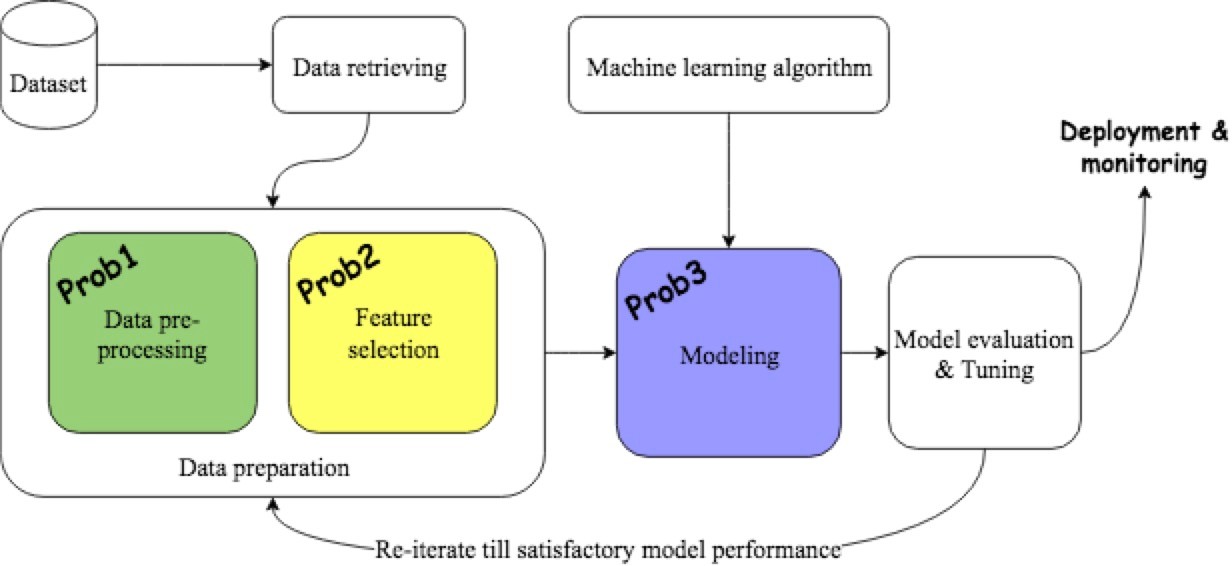


Fig 4.2: A Standard Data Analysis Pipeline

### Cleaning Data

The symptoms of one diagnose are saved as one record in value report and encoded in hexadecimal.

Some rows in the dataset have ”null” values, and we replace the ”null” with a String value ”empty” to separate them with part number. The System evaluating the decision tree model by prediction accuracy [7].

### Data transformation

Data transformation is a process of transforming original data types into desired data types. In the thesis, we transform nominal data into numerical data using one-hot encoder method.

The one-hot encoder, known as the one-of-K scheme, is a widely used data encod- ing method. For a dataset, consisting of multiple categorical values, one-hot encoding transfers the categorical values by creating columns for each of the categorical values and representing them with binary values 0, 1. For the group of binary bits, value 1 is the valid number, representing that the record is in a specific category. Likewise, value 0 is not valid, representing that the record is not in the category. Let us assume,

one dataset is a m ⇤ n matrix dataset, with the total number of categorical values

being s. After one-hot encoding, a new dataset is generated, with shape m ⇤ s. In the new dataset, each categorical values is uniquely represented as one column(feature).

There are 2columns, 3 rows and 6 categorical values in sum. After 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

### Feature selection

Feature selection is a process to select a subset of features, with which we can train a classification model without decreasing the performance of the model. Ideally, the feature selection process finds the minimal subset of features that are sufficient to train a model and approximate a target function [[13](#_bookmark86)]. Given a dataset with m features, X = {x*i*|i 2 {1, ·· · , m}}, feature selection is to select a subset of m*0* features from a set of m features, X = {x*i*|i 2 {1, ·· · , m*0*}}, m*0* < m, such that the value of a criterion function is optimized over all subsets of m features [[14]](#_bookmark87).

The mutual information theory is widely used in feature selection [[15](#_bookmark88)]. Mutual information score measures the general dependency of two random variables [[16](#_bookmark89)]. The score is based on information entropy.

Information entropy is a measurement of the uncertainty of a random variable [[17](#_bookmark90)]. The minimal entropy score is 0, which implies the variable has one fixed value and the variable is quite certain. Let P(x*i*) denotes the probability that X = x*i*, the entropy of variable X can be defined as

*n*

X

H(X) = — P(x*i*) log P(x*i*) (3.1)

*i*=1

Where H(X) is the entropy of variable X, n is the number of the variable values.

Joint entropy is a measurement of uncertainty associated with a set of variables [[18](#_bookmark91)]. Similar to entropy, joint entropy of two variables X and Y relate to the joint probability that P(X = x*i*,Y = y*j*). Let P(x*i*, y*j*) denotes the probability that X = x*i*,Y = y*j*, the joint entropy of the variables X and Y can be defined by

*m,n*

X

H(X, Y ) = — P(x*i*, y*j*) log P(x*i*, y*j*) (3.2)

*i,j*

Where H(X, Y ) is the joint entropy of variable X and Y , m, n are the number of the variable values for X and Y respectively.

Conditional entropy is a measurement of uncertainty of one variable, when the other variable is known [[17]](#_bookmark90) [[19].](#_bookmark92) Given two variables X and Y , the minimal value of conditional entropy of the two variables is 0. That means there is no uncertainty in X if the variable Y is known. In other words, the variable X is certain and dependent on variable Y . Let P(x*i*|y*j*) denotes the probability that X = x*i* on condition that

Y = y*j*, formally the conditional entropy is defined by

*m,n*

X

H(X|Y ) = — P(x*i*|y*j*) log P(x*i*|y*j*) (3.3)

*i,j*

The mutual information tells us the reduction in entropy in one variable when the value of the other is known. It is a measurement of the amount of information that one random variable has about the other variable [[17]](#_bookmark90) [[20].](#_bookmark93) It can be used to measure the mutual dependency between the two variables. A higher mutual information score means higher dependency between the variables. With mutual information, we can measure the dependency between part numbers and software downloading

status. Figure shows the entropy, joint entropy, conditional entropy and mutual

[3.2](#_bookmark15)

information of two random variables X and Y . It quantifies the relevance of a feature with the output vector.

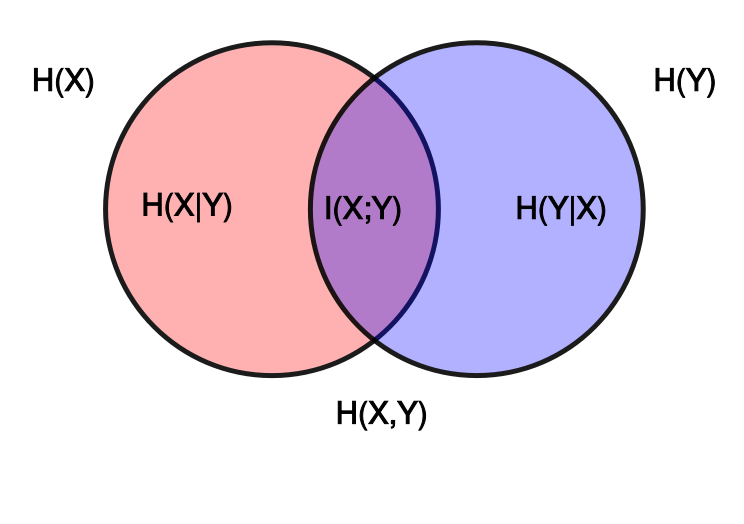


Figure 3.2: Mutual information

The area covered by both circles represents joint entropy H(X, Y ), which can be calculated by equation [3.2.](#_bookmark13) The red area represents conditional entropy H(X|Y ), representing the uncertainty of variable X given variable Y is known. Similarly, the purple area represents conditional entropy H(Y |X), representing the uncertainty of variable Y given variable X is known. Conditional entropy can be calculated using equation equation [3.3.](#_bookmark14) The violet area is mutual information of the two variables X

and Y . Mutual information can also be denoted by

I(X, Y ) = H(X) — H(X|Y )

= H(Y ) — H(Y |X)

= H(X) + H(Y ) — H(X, Y )

### Machine learning tools - classification

(3.4)

Let L = (x11, x12, ·· · , x1*n*, y1), (x21, x22, ·· · , x2*n*, y2), ·· · , (x*m*1, x*m*2, ·· · , x*mn*, y*n*) de-

notes a set of random vectors X, Y , where X represents features, and Y represents target variable (label). The target of classification is to select and train classification models in order to map the features X and the label Y , and then use the trained model to identify the label for unseen data [[21](#_bookmark94)]. The classification model training process can be represented by figure **??**. Typically, the dataset is divided into 2 parts, a training dataset and a testing dataset. The machine learning model learns knowledge and finds out data patterns from the training dataset. Its learning re- sult: the data patterns, is evaluated by prediction accuracy and tested by the testing dataset. Ideally, data that with similar features are partitioned into the same subset by the classification rules.

The performance of a trained model is usually evaluated by its capability of ap- proximating labels for unseen data. There are many performance evaluation criteria, and most of them are based on prediction accuracy. The performance of a trained model can be measured and represented with bias and variance. The bias is a mea- surement of the di↵erence between predicted values and actual values. The variance

is the measurement of sensitivity of a machine learning model in data samples. A

model with a high bias means that the model is not able to represent the relationship between features and target variables, which is under-fitting problem. A model with high variance is not able to generalize over new data. That means that the model not only learns knowledge from the expected data but also from random noises, which is over-fitting problem.

n-fold cross-validation technique is widely used to alleviate the over-fitting prob- lem. By utilising the technique, the dataset is randomly split into n parts, and each time the training model select n — 1 parts as training dataset and the remaining one

as testing dataset. The dataset is trained for n times. The average of the n times of

training scores as a measurement score of prediction performance.

### Decision tree

A decision tree is a tool, which uses conditional statements and a tree-like diagram to determine and represent decisions. It is widely used to solve regression and classification problems. A decision tree is comprised of decision nodes and edges. The decision nodes include one root node, internal nodes and leaf nodes. The root node and internal nodes represent a splitting variable and splitting rules of a decision tree. The leaf nodes, which also are called terminal nodes, contain a small subset of observations. Data with same predicted labels is grouped in the same leaf nodes. A path from the root node to the leaf nodes represents one classification rules .

Two determinations are needed for determining splitting nodes: splitting variables and splitting points. The splitting variables are features in the dataset. The splitting points, which are based on a set of splitting values, determine the way of splitting one variable. There are some criteria to select splitting nodes, such as gini and information gain. According to the splitting criteria, a decision tree is divided into branches (subsets).

The construction of a decision tree starts with the determination of its root note, continues with the determination of its internal nodes and ends when no further

decisions are needed. Figure3.3 is a decision tree, includes 5 nodes and 4 branches.

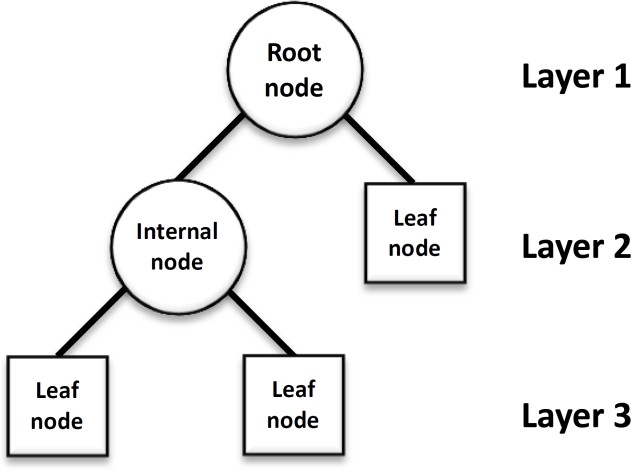


Figure 3.3: Decision tree

The information gain is based on the decrease of entropy after taking one feature as splitting node [[23](#_bookmark96)]. With ”information gain” splitting criterion, features with the highest information gain score are selected as the splitting node. The ”information gain” calculation starts with the calculation of entropy of currently selected variables, and then calculate conditional entropy of the variables with these not-selected fea- tures. The ”information gain” score is the subtraction of the entropy of the selected variables to their conditional entropy with other features. It measures the change of entropy from a prior state to a state after taking the not-selected variables as splitting

nodes. The information gain can be represented by [3.5](#_bookmark19).

IG(Ex, a) = H(T ) — H(T |a) (3.5)

where H(T ) represents entropy of a prior state, IG(Ex, a) represents information gain by adding a feature a as splitting node.

The Gini Impurity (GI) metric is also a widely used splitting criterion in tree based classifier. It measures the homogeneity of data values [[24](#_bookmark97)]. Given the distribution of labels of a set of data, GI measures the frequency of one random element being

incorrectly labelled. If we let p*i* to represent the possibility of an element being labelled as label i, and J represent the number of labels, the GI can be represented

*J*

GI(p) = 1 — (p*i*)2 (3.6)

X

*i*=1

Maximum depth is the maximal length from the root node to the leaf nodes. Setting maximum depth for a decision tree is necessary, especially when a dataset has a large number of features. A deep decision tree continuously learns knowledge from features so that it can capture the data patterns well. A deep decision tree thus rarely has a bias problem. On the other side, a deep decision tree can result in an over-trained model as it learns much misleading information from noise data. That means a deep decision tree can estimate data patterns well for the training dataset, but can not estimate the labels accurately for new data, which cause the over-fitting problem.

There are a few advantages when using the decision tree model. The first one is that the decision tree model is easy to understand and interpret. The paths from the root node to leaf nodes are decision-making rules, and leaf nodes are the labels. Data that meets same rules is placed in the same labels. The second advantage is that it has fewer hyper-parameters to tune with. It is therefore simple to implement and visualize a decision tree model. Besides that, a decision tree can handle di↵erent

data types, which can simplify data pre-processing work.

There are also disadvantages of decision tree model. The first one is the over-fitting problem. It is easy to create a deep decision tree, which can cause the over-training problem and can not generalize over new data. The second is that the decision tree algorithm is a greedy algorithm. The decision tree consistently picks current ”optimal” features as splitting nodes, and in the end the decision tree may not find the optimal data pattern for both training dataset and testing dataset. That is a

typical problem for greedy algorithms. The mentioned disadvantages of the model can be mitigated by the ensemble method.

### Decision tree Approaches for HMC

Decision Tree Approaches for HMC We start this section by defining the HMC task more formally (Section 3.1). Next, we present the framework of predictive clustering trees (Section 3.2), which will be used to instantiate three decision tree algorithms for HMC tasks: an HMC algorithm (Section 3.3), an SC algorithm (Section 3.4), and an HSC algorithm (Section 3.5). Section 3.6 compares the three algorithms at a conceptual level. In this section, we assume that the class hierarchy has a tree structure. Section 4 will discuss extensions towards hierarchies structured as a DAG. 3.1 Formal Task Description We define the task of hierarchical multi-label classification as follows: Given: – an instance space X, – a class hierarchy (C, ≤h), where C is a set of classes and ≤h is a partial order (structured as a rooted tree for now) representing the superclass relationship (for all c1, c2 ∈ C: c1 ≤h c2 if and only if c1 is a superclass of c2), – a set T of examples (xi , Si) with xi ∈ X and Si ⊆ C such that c ∈ Si ⇒ ∀c 0 ≤h c : c 0 ∈ Si , and – a quality criterion q (which typically rewards models with high predictive accuracy and low complexity). Find: a function f : X → 2 C (where 2C is the power set of C) such that f maximizes q and c ∈ f(x) ⇒ ∀c 0 ≤h c : c 0 ∈ f(x). We call this last condition the hierarchy constraint. In this article, the function f is represented with decision trees. Table 1. The top-down induction algorithm for PCTs. I denotes the current training instances, t an attribute-value test, P the partition induced by t on I, and h the heuristic value of t. The superscript ∗ indicates the current best test and its corresponding partition and heuristic. The functions Var, Prototype, and Acceptable are described in the text. procedure PCT(I) returns tree 1: (t ∗ , P ∗ ) = BestTest(I) 2: if t ∗ 6= none 3: for each Ik ∈ P∗ 4: treek = PCT(Ik) 5: return node(t ∗ , S k {treek}) 6: else 7: return leaf(Prototype(I)) procedure BestTest(I) 1: (t ∗ , h∗ , P ∗ ) = (none, 0, ∅) 2: for each possible test t 3: P = partition induced by t on I 4: h = Var(I) − P Ik∈P |Ik| |I| Var(Ik) 5: if (h > h∗ ) ∧ Acceptable(t, P) 6: (t ∗ , h∗ , P ∗ ) = (t, h, P) 7: return (t ∗ , P ∗ )

(a) (b)

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

2.1 2.2

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

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

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

**Fig. 2.** (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.

## Data Dictionary

|  |  |
| --- | --- |
| **Module** | **Description** |
| Login Module | This consists of entries for username and password in which if entered correctly, takes you  into the main interface. |
| Main Interface | The main interface consists of five respective modules. The menus are as follows: registration,  change password, fault, user and diagnosis. |
| Registration | This is a car registration module that allows users to store faulty car records in a database. These records help in keeping the information of the customer. It consists of the plate number, car type, customer name, the amount paid, address,  phone number and date. |
| Change Password | Is the window that enables users to change the existing password by filling the spaces for username, password, new password and confirm new password. The entries can be saved into the database when the change button is clicked, and the cancel button terminates the process as  usual. |
| 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 | New user can be added to the database by clicking on the user module, the form has spaces for accepting full name, username, password and confirm password. The create button store the record in the database and cancel button  terminate the process. |
| Diagnosis | This is the module that takes care of diagnoses on the car to detect the fault and display the generated report. It consists of a decision tree model for inputting the symptoms observed in a car. |

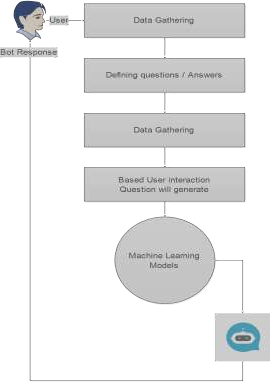
# Dataset Design

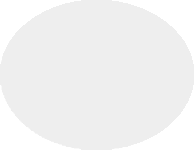
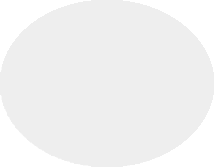
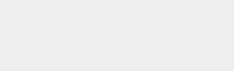
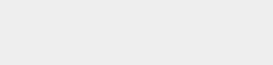
|  |  |  |  |
| --- | --- | --- | --- |
| Symptom 1 | symptom 2 | symptom 3 | car\_problem |
| The engine turns over slowly | The headlights go out as the engine was cranking | There is a cruddy-like substance on the terminals | Flow of electrical current to starter interfered |
| The engine turns over slowly | The headlights go out as the engine was cranking | There is no cruddy-like substance on the terminals | Weak battery |
| The engine turns over slowly | The headlights did not go out as the engine was cranking | There is grinding or growling sound | Bad starter |

# Human Interface Design

## Overview of User Interface

Describe the functionality of the system from the user s perspective. Explain how the user will be able to use your system to complete al the expected features and the feedback information that will be displayed for the user.





User

User opened the app

GUI User Interface

iRepair application

Machine Learning

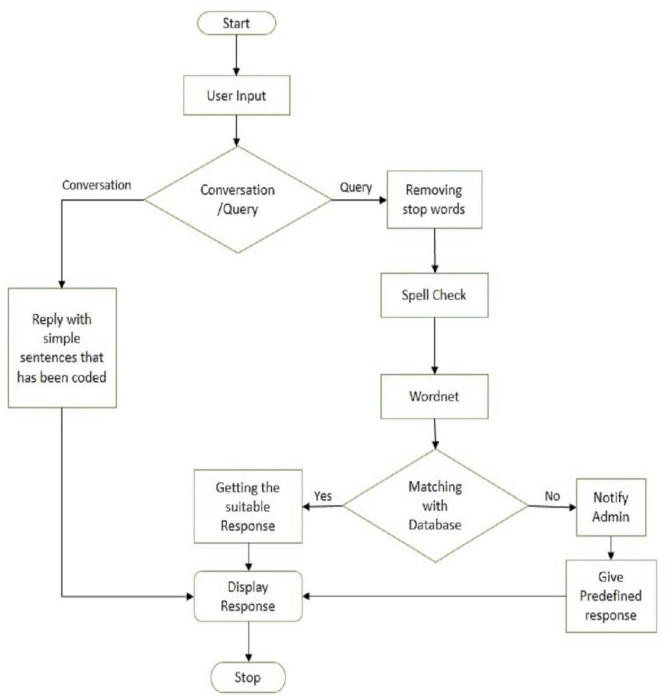
Model Building

Response to User

Query Processing

Learning Data

Chatbot Interface



## 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.