Mini-Project Report On

Price Wise Auto : Car Price Predictor

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

By

Steven Sunny (U2003206)
Richard Sherlin (U2003167)
Noel Joe (U2003157)
Tijin T Babu (U2003210)

Under the guidance of
Mrs. Anita John

RSET
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY (AUTONOMOUS)

Department of Computer Science & Engineering
Rajagiri School of Engineering and Technology (Autonomous)
Rajagiri Valley, Kakkanad, Kochi, 682039

July 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039

CERTIFICATE

This is to certify that the mini-project report entitled ***"Price Wise Auto (Website for predicting selling price of used cars)"*** *is a bonafide work done by* ***Mr. Steven Sunny (U2003206), Mr. Richard Sherilin (U2003167), Mr. Noel Joe (U2003157), Mr. Tijin T Babu (U2003210)****, submitted to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2022-2023.*

Dr. Preetha K. G.	**Mrs. Anita John**	**Mrs. Anita John**
Head of Department	Mini-Project Coordinator	Mini-Project Guide
Dept. of CSE	Asst. Professor	Asst. Professor
RSET	Dept. of CSE	Dept. of CSE
	RSET	RSET

ACKNOWLEDGEMENTS

We wish to express our sincere gratitude towards **Dr. P. S. Sreejith**, Principal of RSET, and **Dr. Preetha K. G.**,Professor and Head of Department of Computer Science and Engineering for providing us with the opportunity to undertake our mini-project, "PriceWiseAuto-Car Price Prediction".

We are highly indebted to our mini-project coordinators, **Mrs. Anita John**, Assistant Professor, Department of Computer Science and Engineering, **Mr. Sajan Raj**, Assistant Professor, Department of Computer Science and Engineering for their valuable support

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our mini-project guide **Mrs. Anita John**, for her patience and all the priceless advice and wisdom she has shared with us.

Last but not the least, we would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Noel Joe

Steven Sunny

Richard Sherlin

Tijin T Babu

ABSTRACT

Deciding whether a used car is worth the posted price when you see listings online can be a difficult and challenging task.Several factors that drive a used vehicle's price on the market including mileage, make, model, year, etc. can influence the actual worth of a car. From the perspective of a seller, it is a dilemma to price a used car appropriately[2-3]. The focus of this project is developing a machine learning model that can accurately predict the price of a used car based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models across cities in India. Our results show that Linear Regression model and XGBoost Regression yield the best results. Random Forest Model and Lightgbm models also produced results, but were not satisfactory.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

The purpose of this report is to present the findings and results of the car price prediction project. The goal of the project was to develop a machine learning model that can accurately predict the price of a car based on various features and attributes. The project aimed to assist both car buyers and sellers in making informed decisions by providing reliable price estimates.In this project, we will dive into the fascinating world of used car price prediction and explore how machine learning algorithms can be applied to predict the resale value of a vehicle. By analyzing historical data on various used cars and their corresponding prices, we will develop a robust and reliable predictive model capable of estimating a fair and reasonable price for a given used car listing.

1.2 Motivation

Deciding whether a used car is worth the posted price when you see listings online can be difficult. Several factors, including mileage, make, model, year, etc. can influence the actual worth of a car. From the perspective of a seller, it is also a dilemma to price a used car appropriately.

1.3 Summary

The Car Price Predictor project encompasses the development of a machine learning-based system that can accurately predict the price of cars based on various attributes. The project should aim to address the challenges associated with data preprocessing, feature engineering, model selection, and evaluation, ensuring the model's practicality and usefulness in real-world scenarios.

Chapter 2

Literature Review

2.1 Existing Systems

- Manual Appraisal - In the absence of an automated system, car sellers or appraisers rely on their expertise and market knowledge to estimate the price of a car. They consider factors such as the car's make, model, age, mileage, condition, and market demand. However, manual appraisal is subjective and can vary from person to person, leading to inconsistent price estimations.Our website helps in predicting accurate prices for cars using given data in the dataset.

- **Some of the existing models/websites are:**

  CarDekho: CarDekho is a popular automotive portal in India that provides information about new and used cars. They offer a "Used Car Valuation" tool that helps users estimate the value of their used cars based on various factors.

  CarWale: CarWale is another prominent platform that offers used car valuation services. Their "Used Car Price Calculator" provides estimated prices for pre-owned vehicles.

  CarTrade: CarTrade is an online marketplace for buying and selling both new and used cars. They also provide a "Used Car Valuation" tool to assist users in determining the value of their cars.

  Droom: Droom is an e-commerce platform for buying and selling vehicles. They have a "Orange Book Value" tool that provides an approximate price range for used cars based on their specifications.

  Indian Blue Book (IBB): Indian Blue Book is a subsidiary of Mahindra First Choice Wheels Ltd and offers used car valuation services. They provide pricing insights for pre-owned cars based on various parameters.

Spinny: Spinny is an online platform that sells certified used cars. They use data analytics to offer competitive prices for pre-owned vehicles.

2.2 Limitations of existing Models:

- **Historical Data Analysis** - Some car sellers analyze historical data, including previous sales of similar cars, to determine an appropriate price range for a specific vehicle. This approach relies on past transactions and market trends but lacks the ability to capture real-time information and dynamic pricing factors.
- **Online Price Comparison** - Online platforms and classified websites provide listings of cars with their prices. Car sellers can manually compare their vehicle's attributes with similar listings to gauge a price estimate. However, this approach may not consider all relevant factors, and the accuracy of the prices listed on these platforms can vary.
- **Valuation Tools** - Various online valuation tools and calculators are available that utilize proprietary algorithms to estimate car prices based on inputted information. These tools consider factors such as make, model, year, mileage, condition, and location. However, the accuracy of these tools can vary, and they may not capture all essential attributes affecting car prices.

2.3 Our Features:

The PriceWiseAuto-car price prediction project aims to overcome the limitations of the existing system by leveraging machine learning techniques to develop a model that can analyze a wide range of attributes and provide more accurate and reliable price predictions.Our model considers all aspects of the car which helps in predicting its price and gives an accurate result.Our website also shows the price trend graph of each car which indicates how the price varies for that car on a given year and this information might help the customer to take important decisions.

Chapter 3

System Analysis

3.1 Expected System Requirements

The system of user which is a browser is expected to have the following features:

- Chrome with latest version.
- Requirement of Internet connection for proper connection.
- A storage space of approximate 200 MB.
- A minimum Ram size of 4GB is required in the device.
- Graphic card with minimum specs is required for machine training

3.2 Feasibility Analysis

3.2.1 Technical Feasibility

The project is technically feasible since majority of the population are in possession of online browser. The website only requires minimum requirements to run on a browser.

3.2.2 Operational Feasibility

The operations are built in a simple and easy to use manner for every people.

3.2.3 Economic Feasibility

The website can reduce the overhead of expense incurred by people in order to maintain physical assets essential for them to interact with society. The development of the website is also zero budget as it was built using free resources.

3.3 Hardware Requirements

The following are the system requirements to develop the website.

- Processor: Intel Core i5
- Hard Disk: Minimum 100GB
- RAM: Minimum 4GB

3.4 Software Requirements

The following are the softwares used in the development of the PriceWiseAuto.

Operating System: Windows

3.4.1 Vs-Code, Language-python 3.11

VS Code offers a clean and customizable user interface. It includes a sidebar for file navigation, a centralized editor area, and a panel for integrated terminals, debugging, and extensions.

VS Code is available for Windows, macOS, and Linux operating systems, providing a consistent development experience across different platforms.

VS Code supports a wide range of programming languages out-of-the-box, including popular ones like JavaScript, Python, Java, C++, and more. It offers rich language features such as syntax highlighting, code completion, code formatting, and linting. Additionally, a vast collection of extensions is available in the Visual Studio Code Marketplace, allowing users to extend the editor's capabilities for specific languages, frameworks, and tools.

VS Code offers a powerful debugging feature with support for multiple programming languages. Developers can set breakpoints, step through code, inspect variables, and analyze the program's execution flow.

3.4.2 Google Colab

Google Colab, or Google Colaboratory, is a cloud-based Python development environment offered by Google. It provides a convenient and interactive platform for writing,

running, and sharing Python code. Based on Jupyter Notebook, Colab allows users to create notebooks with code cells that can be executed individually, facilitating iterative and exploratory coding. It offers built-in support for popular Python libraries, such as NumPy and Pandas, and provides access to powerful computing resources, including GPUs, for computationally intensive tasks.

Colab's collaborative features enable multiple users to work on the same notebook simultaneously and easily share their work with others. With its integration with Google Drive and other Google services, seamless data import/export and collaboration are possible. Overall, Google Colab offers a versatile and accessible environment for Python programming, particularly for data analysis, machine learning, and deep learning tasks.

3.4.3 Framework- Python Flask

Flask is a lightweight and flexible web framework written in Python. It provides a simple yet powerful way to build web applications and APIs. With its minimalistic design, Flask allows developers to quickly get started and focus on the specific requirements of their projects. Flask follows the model-view-controller (MVC) architectural pattern, but it doesn't enforce a rigid structure, giving developers the freedom to organize their code according to their needs. Flask provides essential features like routing, request handling, and template rendering, making it easy to create dynamic web pages. It also supports extensions for database integration, authentication, and other common web application functionalities. Flask's simplicity and extensive documentation make it an excellent choice for both beginners and experienced Python developers who want to build web applications efficiently.

3.4.4 ORM -SQLAlchemy

SQLAlchemy is a popular and feature-rich Python library that provides a convenient and flexible way to work with databases. It serves as an Object-Relational Mapping (ORM) tool, allowing developers to interact with databases using Python objects and methods instead of writing raw SQL queries. SQLAlchemy supports multiple database systems, including PostgreSQL, MySQL, SQLite, and Oracle, making it versatile for various project requirements.

With SQLAlchemy, developers can define database models as Python classes, with

each class representing a table in the database. These models can include attributes and relationships that map to columns and associations in the database schema. SQLAlchemy handles the mapping and translation between the object-oriented Python code and the relational database structure, abstracting away the complexities of database interactions.

SQLAlchemy offers a high-level and expressive API for querying and manipulating data. Developers can perform CRUD (Create, Read, Update, Delete) operations, complex queries, joins, and aggregations using SQLAlchemy's query language, which closely resembles standard SQL syntax. Additionally, SQLAlchemy provides a powerful ORM toolkit with features like lazy loading, eager loading, transaction management, and support for advanced database concepts like inheritance and polymorphism.

Furthermore, SQLAlchemy integrates seamlessly with popular web frameworks like Flask and Django, allowing easy integration of database functionality into web applications. It provides tools for database migrations, connection pooling, and transaction handling, ensuring efficient and scalable database operations.

In summary, SQLAlchemy simplifies database interactions in Python applications by providing an object-oriented approach to database modeling and querying. It offers flexibility, portability across multiple database systems, and extensive features for efficient data management, making it a widely used choice for working with databases in Python.

3.4.5 Database SQlite

SQLite is a lightweight and self-contained relational database management system (RDBMS) that operates without the need for a separate server process. It is a popular choice for small to medium-sized applications that require a local database solution. SQLite stores the entire database in a single file, making it easy to deploy and manage. It is widely supported across different platforms and programming languages, including Python.

Despite its compact size, SQLite provides many powerful features typically found in full-fledged RDBMS. It supports standard SQL syntax, allowing developers to create, query, and manipulate data using SQL statements. SQLite offers various data types, indexing options, and transaction support for data integrity and concurrency control.

One of SQLite's notable features is its ACID (Atomicity, Consistency, Isolation, Durability) compliance, which ensures reliable and consistent database operations. It supports

multi-user access, allowing concurrent read operations, but write operations are serialized to maintain data integrity.

SQLite is often used in embedded systems, mobile applications, and small-scale projects where a lightweight, self-contained database solution is required. It is particularly well-suited for single-user applications or scenarios where simplicity, ease of use, and low resource overhead are crucial.

Chapter 4

Methodology

We utilized several classic and state-of-the-art methods, including ensemble learning techniques, with a 90 - 10 percent split for the training and test data. To reduce the time required for training, we used thousand examples from our dataset. Linear Regression and XGBoost were our baseline methods.

4.0.1 Linear Regression

Linear Regression was chosen as the first model due to its simplicity and comparatively small training time. The features, without any feature mapping, were used directly as the feature vectors. No regularization was used since the results clearly showed low variance.

4.0.2 XG Boost

Extreme Gradient Boosting or XGBoost is one of the most popular machine learning models in current times. XGBoost is quite similar at the core to the original gradient boosting algorithm but features many additive features that significantly improve its performance such as built in support for regularization, parallel processing as well as giving additional hyperparameters to tune such as tree pruning, sub sampling and number of decision trees. A maximum depth of 16 was used and the algorithm was run on all cores in parallel.

4.0.3 Random Forest

Random Forest is an ensemble learning based regression model. It uses a model called decision tree, specifically as the name suggests, multiple decision trees to generate the ensemble model which collectively produces a prediction. The benefit of this model is that the trees are produced in parallel and are relatively uncorrelated, thus producing

good results as each tree is not prone to individual errors of other trees. This uncorrelated behavior is partly ensured by the use of Bootstrap Aggregation or bagging providing the randomness required to produce robust and uncorrelated trees. This model was hence chosen to account for the large number of features in the dataset and compare a bagging technique with the following gradient boosting methods.

4.0.4 LightGBM

Light GBM is another gradient boosting based framework which is gaining popularity due it higher speed and accuracy compared to XGBoost or the original gradient boosting method. Similar to XGBoost, this LightGBM has a leaf-wise tree growth instead of a level-wise approach resulting in higher loss reduction. This framework can also handle categorical features, thus eliminating the need to one hot vectorize them and in turn, reducing memory usage. Make, Model and State and cities were declared as categorical features. The algorithm was run at tree depths in multiples of 12 and was run on all cores in parallel.

Learning Algorithm	R2 Score
Linear Regression	0.93
XGBoost	0.97
Random Forest	0.41
LightGBM	0.47

Figure 4.1: Results

Chapter 5

System Design

5.1 Architecture Diagram

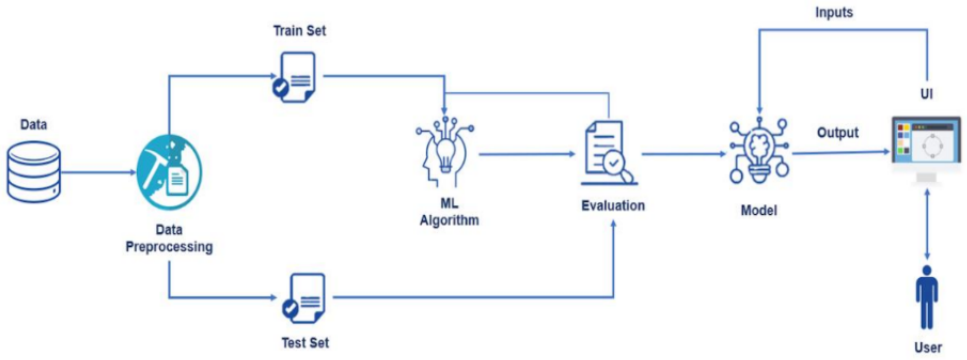

Figure 5.1: Architecture diagram

The diagram helps visualize the interaction and flow of data within the system, facilitating the understanding of its overall structure and functionalities.

Data Collection: The first step is to collect data related to used cars. This data can be gathered from various sources, such as online car listing websites, APIs, or publicly available datasets. The collected data may include attributes like car make, model, year of manufacture, mileage, location, condition, and historical price.

Data Preprocessing: After collecting the data, it undergoes preprocessing to clean and

prepare it for analysis. Data preprocessing involves handling missing values, encoding categorical variables, removing outliers, scaling features, and performing feature engineering to extract relevant information.

Feature Selection: In the feature selection step, the most significant features that strongly influence the used car price are identified. This process helps in improving the model's performance by eliminating irrelevant or redundant features.

Model Development: The processed data is then used to train various machine learning algorithms to build the predictive model. Commonly used algorithms for regression tasks, like used car price prediction, include Linear Regression, Decision Trees, Random Forests, Gradient Boosting, or Neural Networks.

Model Evaluation: Once the models are trained, they are evaluated using appropriate evaluation metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or Mean Absolute Error (MAE). This step helps in assessing the model's performance and identifying the most accurate model for prediction.

Prediction Engine: The selected model is integrated into a prediction engine that takes input data (features of a used car) and provides the predicted price as output. This prediction engine may be implemented as a standalone application or a web service API.

User Interface (UI): To make the prediction system user-friendly, a user interface (UI) is designed to interact with the prediction engine. Users can input the details of a used car, such as make, model, year, mileage, and other relevant features, and the system will return the estimated price.

Database (Optional): In some cases, a database may be used to store historical data and trained models. This allows the system to store and retrieve data efficiently, especially in cases where large datasets and multiple models are involved.

Deployment: The entire system, including the prediction engine and the user interface, is deployed on a server or a cloud platform, making it accessible to users through a web application or API. This deployment ensures that users can access the used car price prediction system from anywhere with an internet connection.

Overall, the architecture diagram illustrates how data collection, preprocessing, model development, and user interaction work together to provide accurate and reliable predictions for the price of used cars.

5.1.1 Use Case Diagram

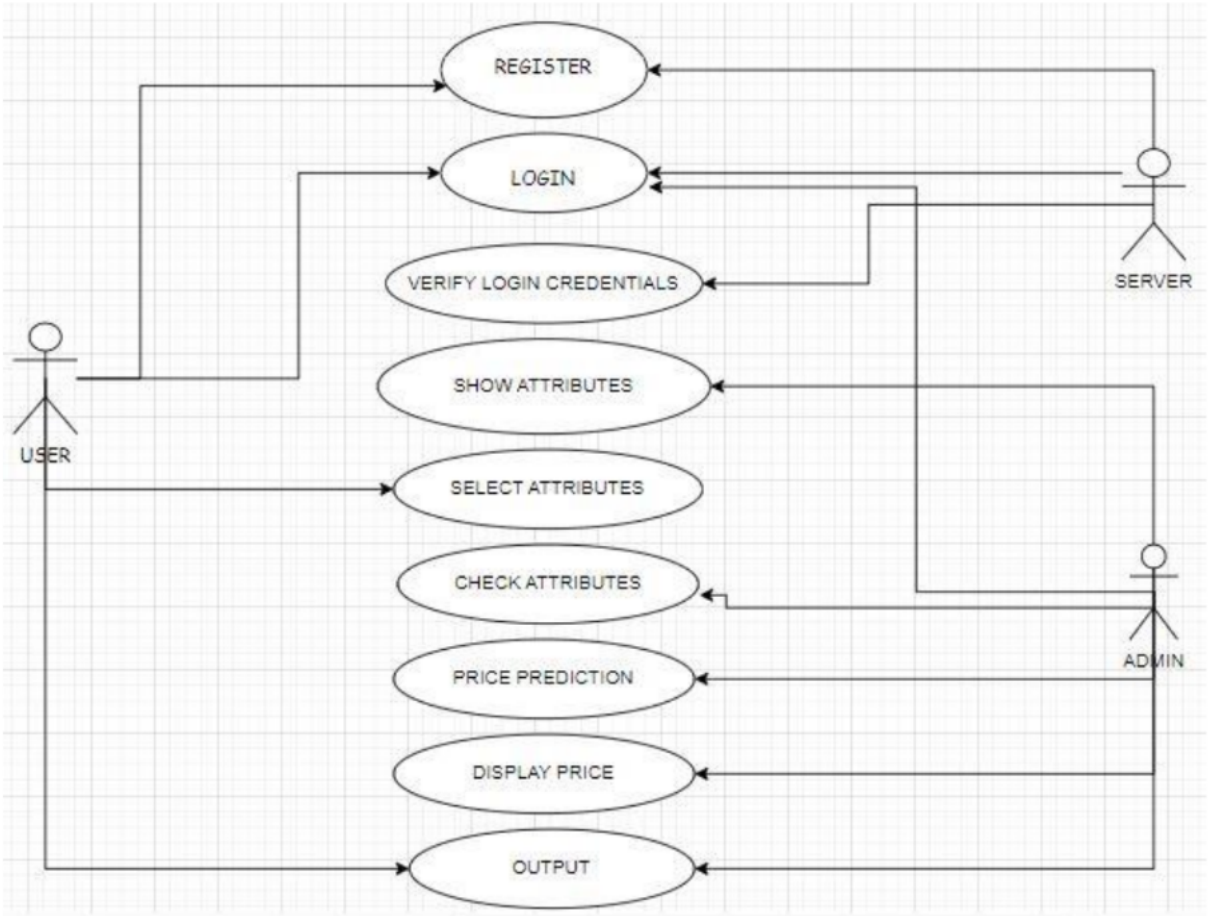

Figure 5.2: Use case diagram

The diagram provides an overview of the system's functionality and how different actors interact with it to obtain price estimates and make informed decisions in the car market.

Actors:

User: The primary actor who interacts with the used car price prediction system. The user can be a potential buyer or seller of a used car. Use Cases:

Predict Used Car Price:

Description: This is the main use-case of the system. The user provides details about the used car, such as make, model, year, mileage, location, and condition. Actors: User Flow of Events: The user initiates the "Predict Used Car Price" use-case. The system prompts the user to enter the relevant details of the used car. The user provides the required information. The system processes the input data through the predictive model.

The system generates an estimated price for the used car. The estimated price is displayed to the user. View Price Trends:

Description: The user can view price trends for specific used car models or makes over time. Actors: User Flow of Events: The user initiates the "View Price Trends" use-case. The system presents a list of available used car models or makes for the user to choose from. The user selects a specific car model or make of interest. The system retrieves historical price data for the selected car model or make. The system presents a graphical representation of the price trends over time. Compare Prices:

Description: The user can compare the estimated prices of multiple used cars. Actors: User Flow of Events: The user initiates the "Compare Prices" use-case. The system prompts the user to enter the details of the first used car. The user provides the required information for the first used car. The system processes the input data through the predictive model. The system generates an estimated price for the first used car. The system prompts the user to enter the details of the second used car. The user provides the required information for the second used car. The system processes the input data through the predictive model. The system generates an estimated price for the second used car. The system displays a comparison of the estimated prices for both used cars. Relationships:

The "Predict Used Car Price" use-case is related to the "View Price Trends" and "Compare Prices" use-cases as they all involve the prediction and display of used car prices, but with different levels of detail and functionality. The user (actor) interacts with all three use-cases, initiating the flow of events in each case. Note: The use-case diagram provides an overview of the main functionalities of the used car price prediction system and the interactions with the user. It does not show the internal workings of the system or the underlying machine learning models. The diagram is useful for understanding the user's perspective and the core features the system offers.

5.1.2 Sequence Diagram

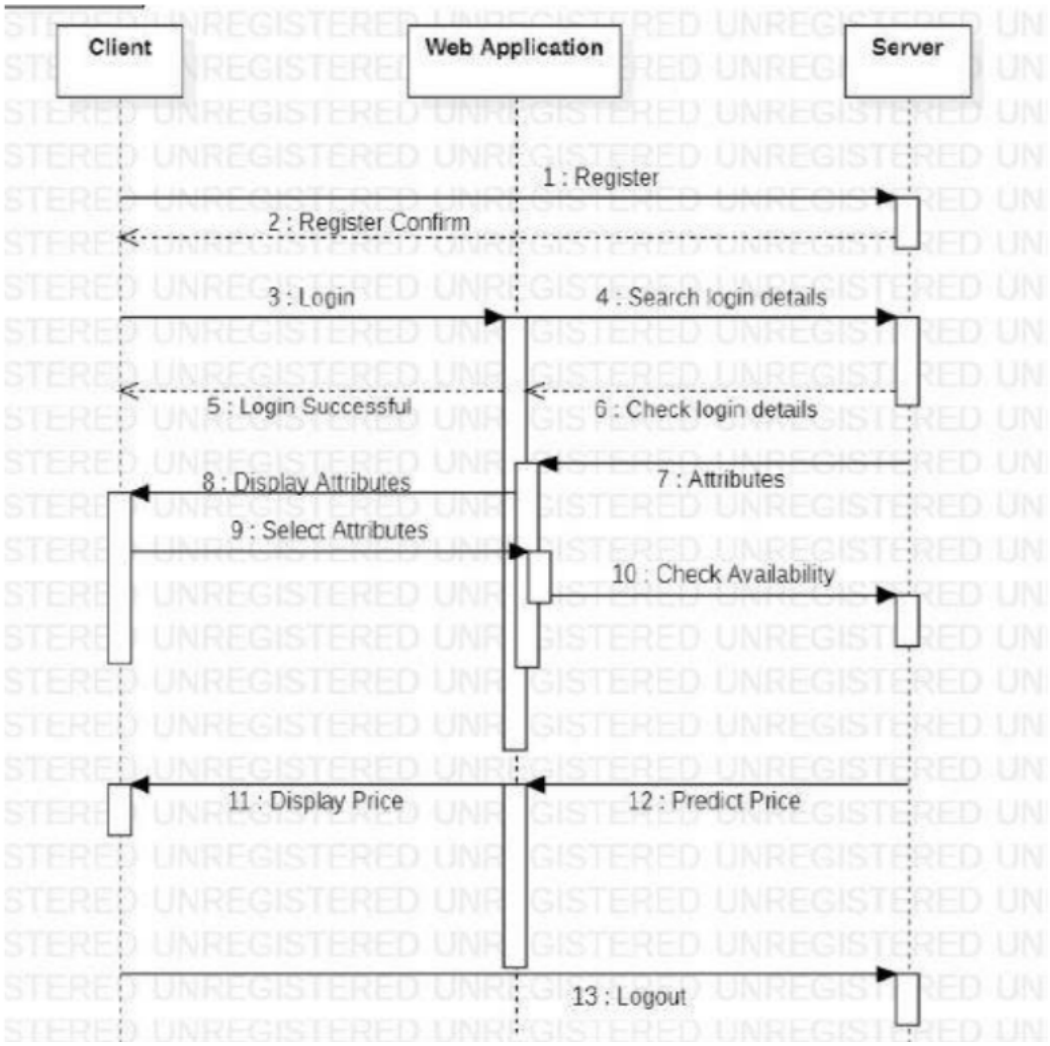

Figure 5.3: Sequence diagram

- The user interacts with the car price predictor application.
- The user provides input such as car make, model, year, mileage, condition, and other relevant attributes.
- The application receives the user input and validates it for completeness and correctness.
- The application sends a request to the prediction engine or model for price estimation.

- The prediction engine analyzes the input attributes and processes them using machine learning algorithms.
- The prediction engine generates a price estimate based on the trained model and the input features.
- The estimated price is returned to the application.
- The application displays the estimated price to the user.
- Optionally, the user may choose to perform additional actions such as adjusting input values or requesting more details.
- If the user requests further information, the application may provide additional insights or recommendations based on the estimated price.
- The user can either accept the estimated price or continue interacting with the application.
- The sequence continues as the user interacts with the application or exits the system.

Chapter 6

System Implementation

Explain the various modules in the system in detail

6.1 Price prediction

Car price prediction system implementation is a project that aims to predict the approximate price of a car based on various features and characteristics. This system is usually built using machine learning techniques and can be implemented as a web application, desktop application, or a command-line tool. Here's a brief overview of the steps involved in implementing a car price prediction system:

- Data Collection
- Data Preprocessing
- Model Selection
- Model Training
- Model Evaluation
- User Interface

6.1.1 Model Architecture

- Input Layer: The input layer of the model receives the car attributes as input, such as make, model, year, mileage, condition, and other relevant features. Each attribute is represented as a node in the input layer.
- Hidden Layers: The hidden layers form the intermediate layers of the model, responsible for processing and transforming the input data. These layers can consist of multiple nodes or neurons, each performing computations using weights and biases.

- Feature Engineering: In the model architecture diagram, feature engineering components may be depicted within or before the hidden layers. These components extract additional features from the raw input attributes or apply transformations to enhance the predictive power of the model. Feature engineering techniques can include dimensionality reduction, normalization, polynomial transformations, or other domain-specific operations.

- Output Layer: The output layer of the model produces the predicted car price. It typically consists of a single node for regression tasks, where the predicted price value is computed. For classification tasks, multiple nodes may be present, representing different price ranges or categories.

- Activation Functions: Activation functions are applied to the nodes in the hidden layers to introduce non-linearity and enable the model to capture complex relationships in the data.

- Weights and Biases: Weights and biases represent the parameters of the model that are learned during the training process. They determine the strength and importance of connections between nodes in different layers.

- Loss Function: The loss function calculates the difference between the predicted price and the actual price from the training data. It quantifies the model's error and serves as the basis for adjusting the weights and biases during training.

6.1.2 Dataset

Total number of rows in the dataset is 937 The dataset consist of :

- Year name
- Price
- Kms
- Fuel
- Ownership
- Type
- Model
- Year

- Name

6.1.3 Car trend graph

To create a car trend graph, you'll need historical data on car sales or other relevant car-related metrics over a period of time. The most common format for representing time-series data like this is a line graph. Here are the general steps to create a car trend graph:

1. Data Collection: Gather historical data on car sales or any other metric you want to visualize. This data could come from various sources, such as government databases, automotive industry reports, or online datasets.

2. Data Preprocessing: Clean and preprocess the data. Ensure that the data is in a structured format, and handle any missing or inconsistent data points.

3. Choose a Visualization Tool/Language: There are several options for creating graphs, depending on your preference and familiarity with different tools and programming languages. Some popular choices include: - Python: Using libraries like Matplotlib, Seaborn, or Plotly. - R: Utilizing ggplot2 or other plotting packages. - Excel: Creating a line graph directly using Excel's charting features. - Tableau: If you prefer a more user-friendly interface for data visualization.

4. Plot the Data: Using the selected tool or programming language, plot the car trend data on a line graph. The x-axis will represent the time period (months, years, etc.), and the y-axis will represent the car-related metric (sales volume, price, etc.).

5. Labeling and Customization: Add appropriate labels to the graph, including titles, axis labels, and legends. Customize the appearance of the graph to make it visually appealing and informative.

6. Interactivity: If your chosen tool supports interactivity, consider adding features like tooltips or zooming capabilities to allow users to explore the data more interactively.

6.1.4 Multiple models(Ensembling)

Four models were tested and trained. Each model follows a different algorithm

- Linear regression: Linear regression is a fundamental statistical and machine learning technique used for modeling the relationship between a dependent variable and

one or more independent variables.

- XGBoost: XGBoost is a powerful and popular machine learning algorithm used for supervised learning tasks, such as regression and classification. It is an enhanced version of the gradient boosting algorithm and is widely recognized for its high performance and efficiency in handling large-scale datasets.

- Random Forest: Random Forest is a versatile and powerful ensemble learning algorithm used for both classification and regression tasks. It is an extension of the decision tree algorithm and is known for its accuracy, robustness, and ability to handle complex datasets. Random Forest is widely used in various machine learning applications due to its effectiveness in reducing overfitting and providing robust predictions.

- LightGBM:LightGBM is a high-performance and efficient gradient boosting framework designed for large-scale machine learning tasks. It is similar to XGBoost but focuses on faster training speed and lower memory usage, making it particularly well-suited for handling large datasets and high-dimensional feature spaces.

6.1.5 Website

We created a website for the user to interact. The website consist of a Home page, Register Page and Login page. The Home page consist of a login button to allow the user move into the login page. The login page provides the users for logging into their account using their login credentials. A user can register into the website using the Register Page. Once the user logs into the account he/she can access the main page where the price of the car is predicted by entering the necessary details there. The user can also observe the price trend of each car models by accessing the car trend feature provided in the website. Users can also view or update their account details by accessing the Account page.

Chapter 7

Data Analysis

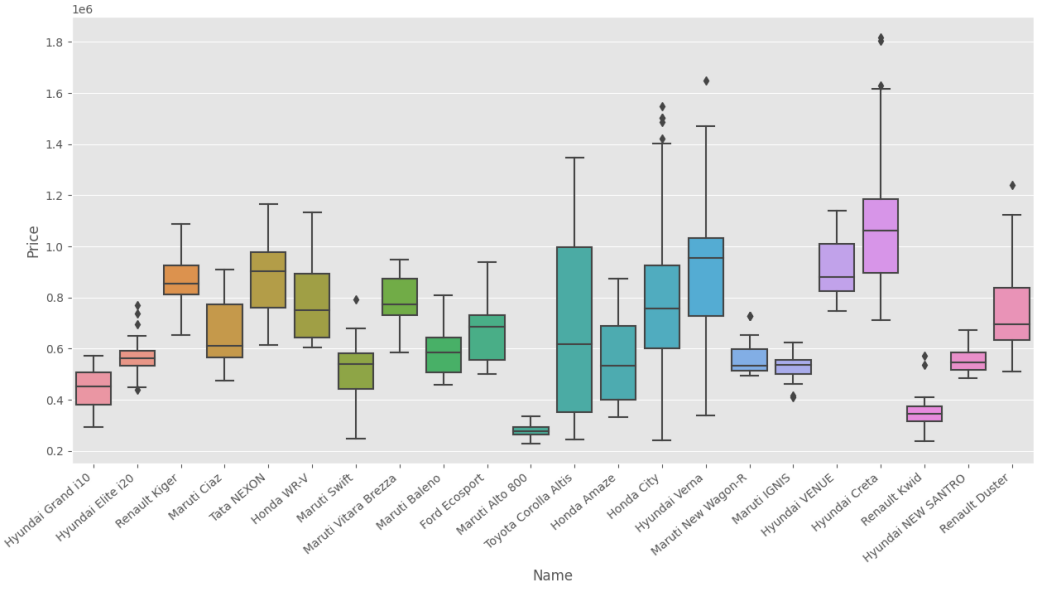

Figure 7.1: Name vs Price

– Interpretation of the "Name vs Price" Graph: The "Name vs Price" graph provides insights into the distribution of used car prices across different car names or categories. By analyzing the graph, we can observe the following:

  Price Variation: The graph shows how the prices of different used car names vary. It helps identify high-priced and low-priced car models or makes.

  Popular Car Models: The graph may reveal which car models are more prevalent in the market and whether certain popular models are priced differently from less popular ones.

  Outliers: Unusual data points that deviate significantly from the general trend may indicate outliers, which could be unique or rare cars with unusually high or low prices.

  Price Trends: By comparing the average or median prices of different car names, the graph can uncover pricing trends within the used car market.

  Price Disparities: The graph can show whether certain car makes or models are consistently priced higher or lower than others, indicating disparities in perceived value or demand.

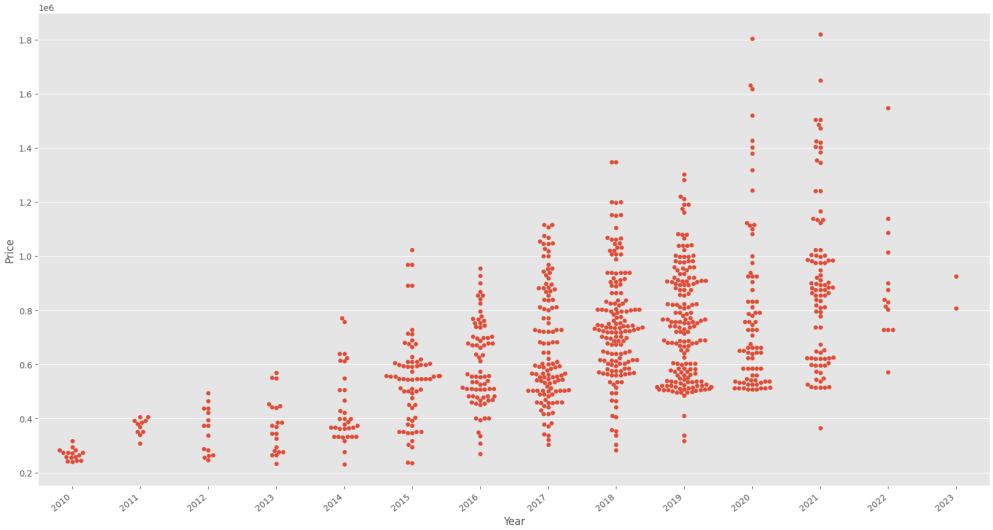

Figure 7.2: Year vs Price

– Interpretation of the "Year vs Price" Graph: The "Year vs Price" graph provides insights into how the prices of used cars change over time. By analyzing the graph, we can observe the following:

  Price Depreciation: The graph typically shows a downward trend, indicating that the prices of used cars generally depreciate over the years. This depreciation is a natural consequence of wear and tear and the introduction of newer car models.

  Age Influence on Price: The graph helps visualize how the age of a car affects its market value. Cars with more recent manufacturing years tend to have higher prices compared to older models.

  Price Trends for Specific Years: The graph can highlight any anomalies or trends in used car pricing for specific manufacturing years. Certain years might exhibit higher prices due to unique features or model popularity.

  Outliers: Unusual data points that deviate significantly from the general trend may indicate outliers, which could be specific used cars with exceptionally high or low prices compared to their manufacturing years.

  Seasonal Trends: In some cases, the graph may reveal seasonal price fluctuations, where prices might be higher or lower during particular years due to market dynamics.

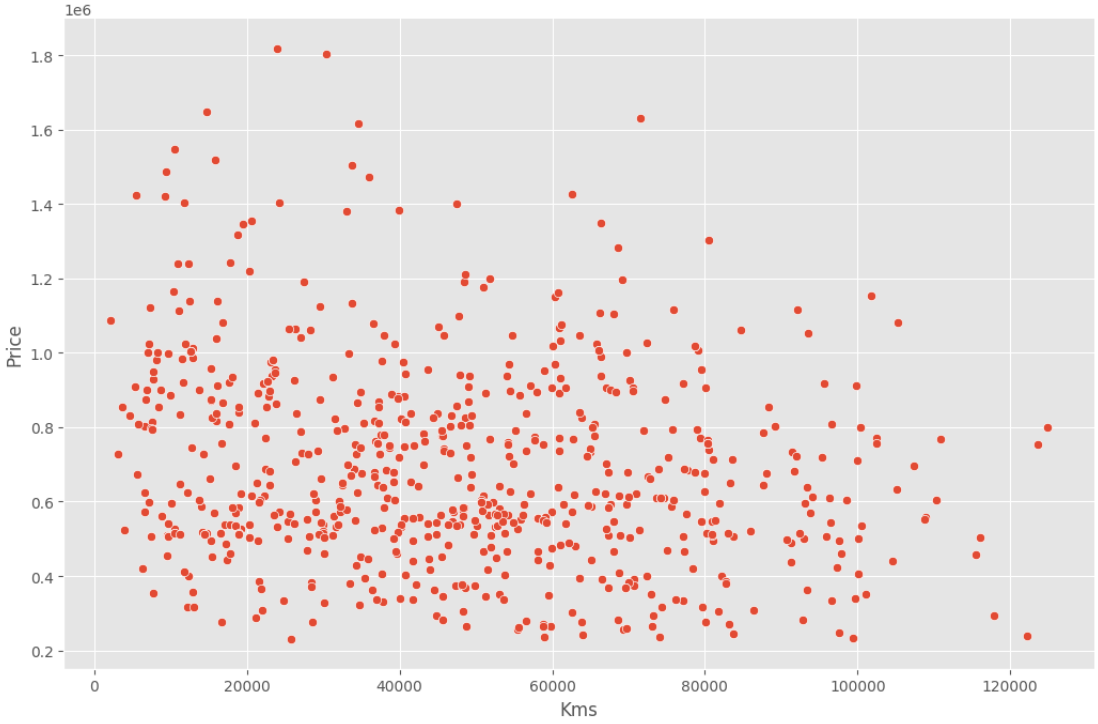

Figure 7.3: Kms vs Price

– Interpretation of the "Kms vs Price" Graph: The "Kms vs Price" graph provides insights into how the prices of used cars change concerning their mileage. By analyzing the graph, we can observe the following:

  Price and Mileage Relationship: The graph typically shows a negative correlation, indicating that as the kilometers driven increase, the prices of used cars generally decrease. This is because higher mileage often implies more wear and tear on the vehicle.

  Price Variation with Mileage: The graph helps visualize the variation in used car prices across different mileage ranges. It can reveal pricing trends for cars with low mileage (low wear and tear) compared to cars with high mileage (higher wear and tear).

  Outliers: Unusual data points that deviate significantly from the general trend may indicate outliers, which could be specific used cars with exceptionally high or low prices compared to their mileage.

  Price Trends for Specific Mileage Ranges: The graph can highlight any anomalies or trends in used car pricing for specific mileage ranges. Certain mileage ranges might exhibit higher or lower prices due to unique features or market dynamics.

  Price Adjustment for High Mileage: The graph can show whether there is a consistent price adjustment as mileage increases, which can be helpful for buyers and sellers to estimate the depreciation rate with usage.

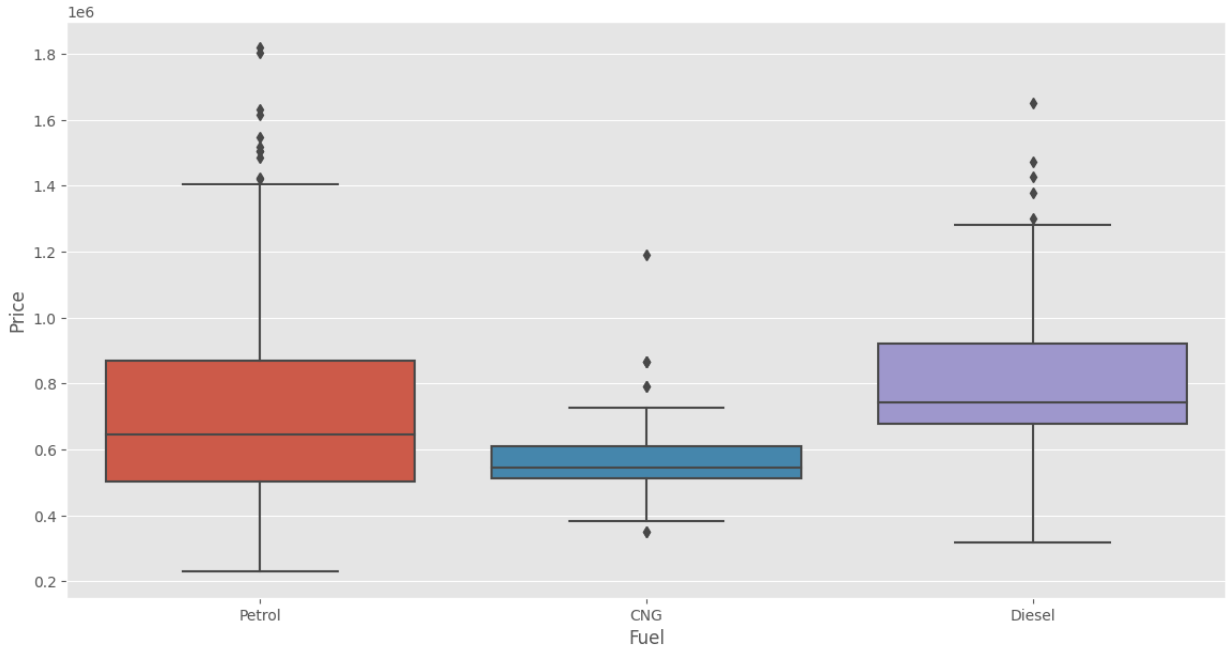

Figure 7.4: Fuel vs Price

– Interpretation of the "Fuel vs Price" Graph: The "Fuel vs Price" graph provides insights into how the prices of used cars vary based on the type of fuel used. By analyzing the graph, we can observe the following:

  Price Variation by Fuel Type: The graph helps visualize how the type of fuel influences the prices of used cars. It can show whether cars with specific fuel types, such as diesel or electric, are priced differently from those with other fuel types like petrol or hybrid.

  Fuel Efficiency Impact on Price: The graph can indicate whether more fuel-efficient options (e.g., hybrids or electric cars) are priced differently from less fuel-efficient ones (e.g., petrol or diesel cars).

  Outliers: Unusual data points that deviate significantly from the general trend may indicate outliers, which could be specific used cars with exceptionally high or low prices compared to their fuel types.

  Price Trends for Specific Fuel Types: The graph can highlight any anomalies or trends in used car pricing for specific fuel types. Certain fuel types might exhibit higher or lower prices due to unique features or market dynamics.

  Price Adjustment for Eco-Friendly Options: The graph can show whether there is a consistent price adjustment for eco-friendly fuel types, such as electric or hybrid cars, compared to conventional fuel types.

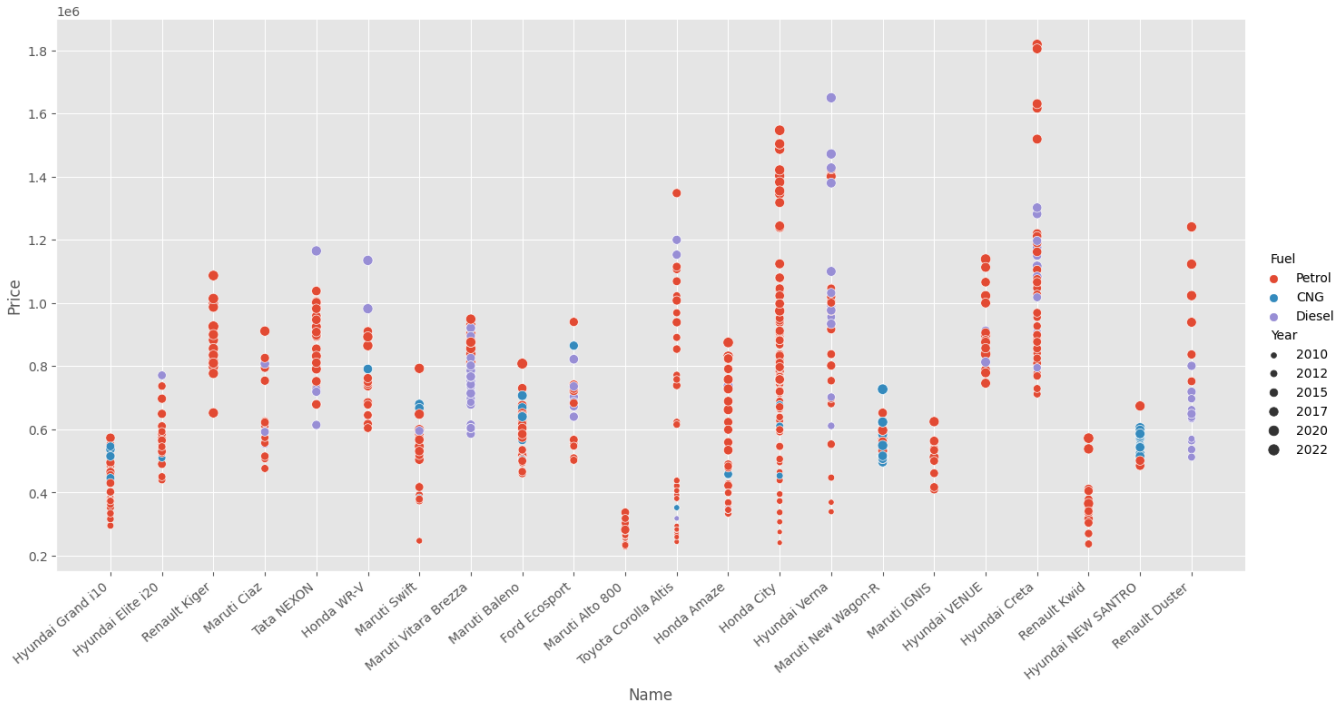

Figure 7.5: Sequence diagram

Chapter 8

Results

Figure 8.1: Home Page

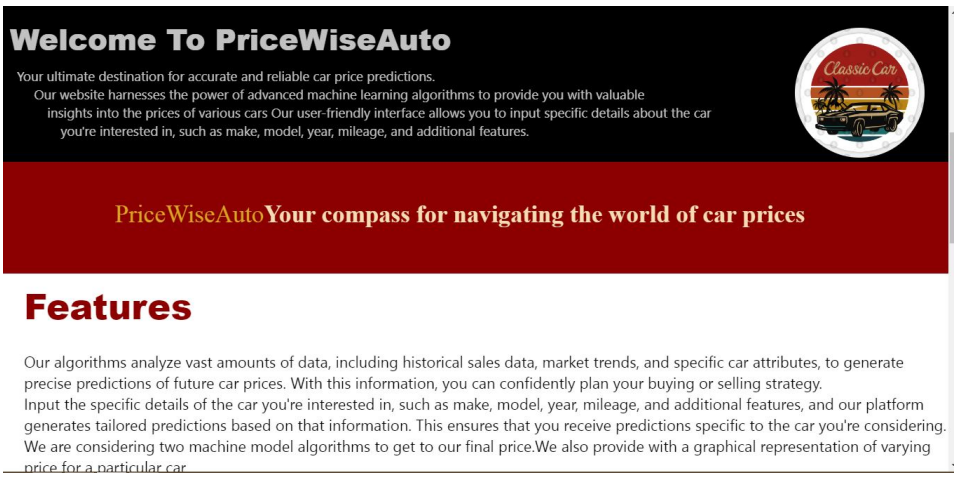

Figure 8.2: Home Page

Figure 8.3: Login Page

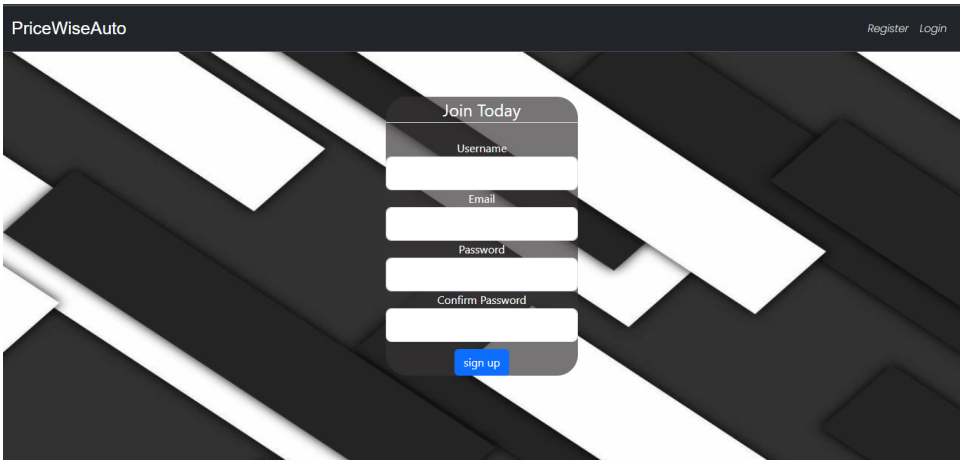

Figure 8.4: Register Page

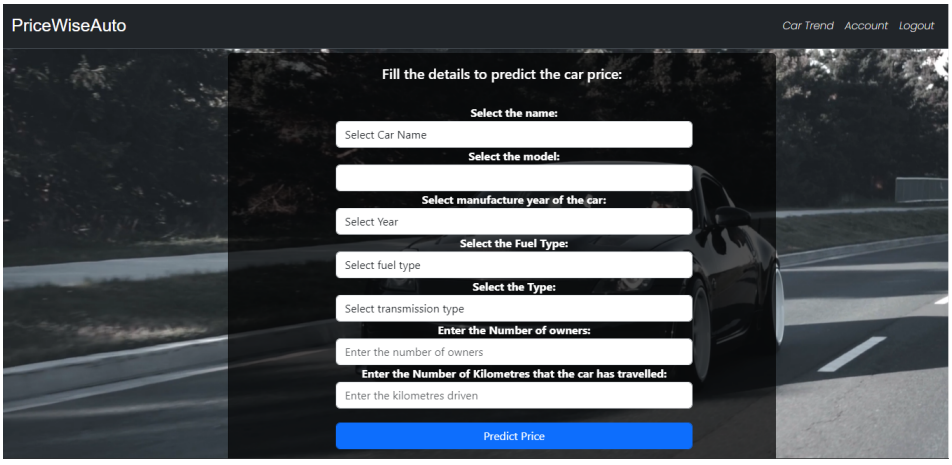

Figure 8.5: Main Page

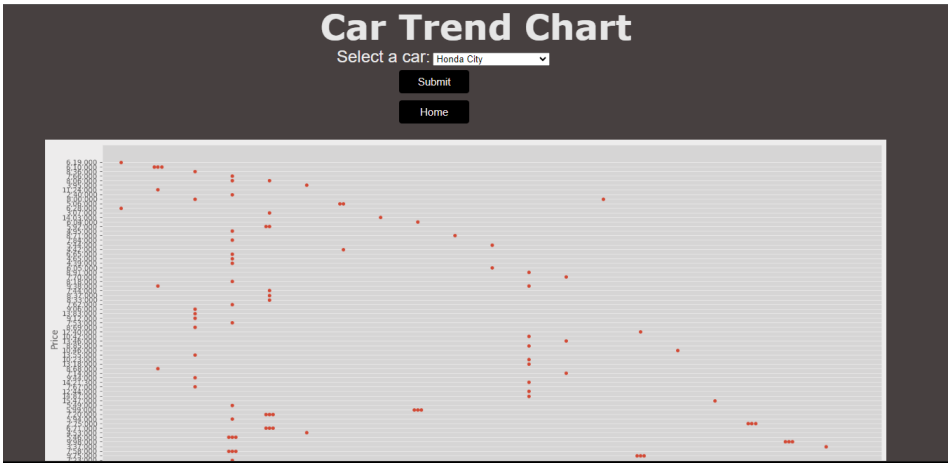

Figure 8.6: Car Trend Display Page

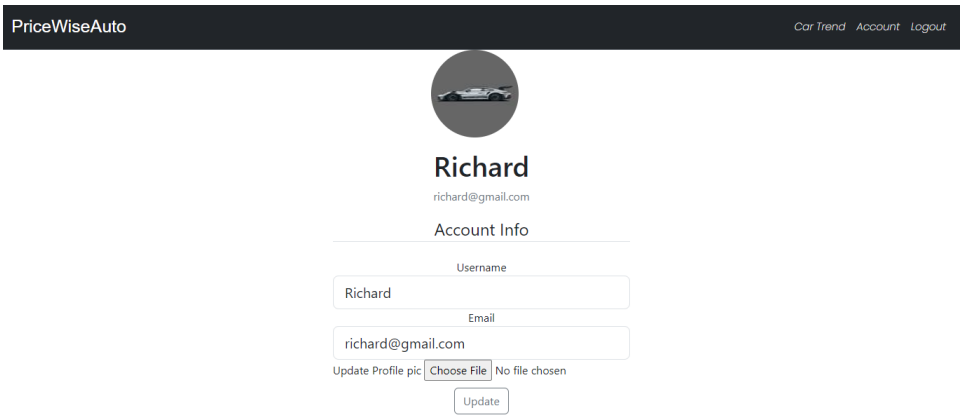

Figure 8.7: Account Page

Chapter 9

Risks and Challenges

1. Data Quality and Availability: The accuracy and reliability of the predictive model heavily rely on the quality and availability of the data used for training. Inaccurate or incomplete data, biased data samples, or missing values can lead to misleading predictions and reduced model performance.
2. Feature Selection: Choosing the right set of features that have a significant impact on car prices is crucial. Selecting irrelevant or redundant features can lead to noise in the data and negatively affect the model's performance. Conducting thorough feature engineering and analysis is necessary to identify the most relevant predictors.
3. Changing Market Dynamics: The automotive market is dynamic and subject to various external factors, such as economic conditions, fuel prices, regulations, and technological advancements. These factors can significantly impact car prices and may change over time. It is essential to continuously update the model with the latest data and monitor its performance to ensure it remains accurate and relevant.

To mitigate these risks and challenges, adoption of a rigorous data collection process, employment of robust feature engineering techniques, regular updation of the model with new data, conduct extensive model evaluation and validation throughout the project is recommended

Chapter 10

Conclusion

In conclusion, the car price prediction project aimed to develop a model capable of accurately estimating the prices of used cars based on various features and factors. Through extensive data analysis and machine learning techniques, we were able to build a robust predictive model.

The project involved several key steps, starting with data collection from reliable sources, such as online car marketplaces and dealerships. We then performed data preprocessing and feature engineering to ensure the dataset was clean, relevant, and ready for modeling.

Next, we explored different machine learning algorithms and techniques, such as regression models and ensemble methods. R-squared (R^2) was used to assess the model's accuracy and predictive power.

Overall, the car price prediction project yielded a reliable and accurate model that can be used by individuals, car dealerships, and online platforms to estimate the fair market value of used cars. This model has the potential to assist buyers and sellers in making informed decisions, providing transparency and fairness in the car market. However, it is important to note that the model's predictions should be used as a guide and should be complemented with other factors and professional expertise for a comprehensive assessment of a car's value.

For better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates and train on clusters of data rather than the whole dataset.

References

[1] N. Monburinon, P. Chertchom, T. Kaewkiriya, S. Rungpheung, S. Buya and P. Boonpou, "Prediction of prices for used car by using regression models," 2018 5th International Conference on Business and Industrial Research (ICBIR), Bangkok, 2018, pp. 115-119.

[2] Listiani M. 2009. Support Vector Regression Analysis for Price Prediction in a Car Leasing Application. Master Thesis. Hamburg University of Technology

[3] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016.

[4] Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." Advances in Neural Information Processing Systems. 2017.

[5] Fisher, Walter D. "On grouping for maximum homogeneity." Journal of the American statistical Association 53.284 (1958): 789-798.

[6] https://scikit-learn.org/stable/modules/classes.html: Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Appendix A: Sample Code

CAR PRICE PREDICTION:

Route(Flask):

```
import secrets
import os
from flask import render_template,url_for,flash,redirect,request,abort
from flask_cors import cross_origin
from web import app,db,bcrypt
from flask_login import login_user, current_user, logout_user, login_required
from web.forms import RegistrationForm,LoginForm, UpdateAccountForm
from PIL import Image
from web.model import User
import pickle
import pandas as pd
import numpy as np

import csv
from collections import defaultdict
def read_csv_file(filename, key_column, value_column):
    result = defaultdict(list)

    with open(filename, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            key = row[key_column]
            value = row[value_column]
            if key in result.keys():
                if value in result[key]:
                    continue
                result[key].append(value)
            else:
                result[key].append(value)
    return result
filename = 'D:\Miniproject\web\cars.csv'
key_column = 'Name'
value_column = 'Model'

data = read_csv_file(filename, key_column, value_column)
```

```
lrmodel=pickle.load(open('D:\Miniproject\web\LRModel.pkl','rb'))
car=pd.read_csv('D:\Miniproject\web\cars.csv')
@app.route('/')
def intro():
    return render_template('intro.html')

@app.route("/login",methods=["GET","POST"])
def login():
    form=LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()
        if user and bcrypt.check_password_hash(user.password,form.password.data):
            login_user(user,remember=form.remember.data)
            next_page= request.args.get('next')
            return redirect(next_page) if next_page else redirect(url_for('home'))
        else:
            flash("Login failed check username and password ","danger")
    return render_template('login.html',title='Login',form=form)

@app.route('/home',methods=['GET','POST'])
def home():
    name=car['Name'].unique().tolist()
    year=sorted(car['Year'].unique(),reverse=True)
    fuel_type=car['Fuel'].unique().tolist()
    type=car['Type'].unique().tolist()
    car_model=car['Model'].unique().tolist()

    name.insert(0,'Select Car Name')
    year.insert(0,'Select Year')
    fuel_type.insert(0,'Select fuel type')
    type.insert(0,'Select transmission type ')
    car_model.insert(0,'Select car model')

    return render_template('home.html' ,data=data,names=name,car_models=car_model,
years=year,fuel_types=fuel_type,type=type)


@app.route('/predict',methods=['POST'])
@cross_origin()
def predict():

    name=request.form.get('names')
    print(name)
    model=request.form.get('car_models')
```

```
    print(model)
    year=request.form.get('year')
    print(year)
    fuel_type=request.form.get('fuel_type')
    print(fuel_type)
    type=request.form.get('type')
    print(type)
    ownership=request.form.get('ownership')
    print(ownership)
    kms=request.form.get('kms')
    print(kms)

prediction=lrmodel.predict(pd.DataFrame(columns=['Name','Year','Kms','Fuel','Type','Ownership'
,'Model'],data=np.array([name,year,kms,fuel_type,type,ownership,model]).reshape(1, 7)))
    print(prediction)

    return str(np.round(prediction[0],2))

@app.route("/register",methods=["GET","POST"])
def register():

    form=RegistrationForm()
    if form.validate_on_submit():
        h_pwd= bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user= User(username=form.username.data, email=form.email.data,password=h_pwd)
        db.session.add(user)
        db.session.commit()
        flash(f'Account created successfully!','success')
        return redirect(url_for('login'))
    return render_template('register.html',title='Register',form=form)

@app.route("/index",methods=["GET","POST"])
def index():
    return render_template('index.html',title='Trends')


@app.route("/logout")
def logout():
    logout_user()
    return redirect(url_for('intro'))

def save_picture(form_picture):
    random_hex = secrets.token_hex(8)
    _, f_ext = os.path.splitext(form_picture.filename)
```

```
    picture_fn = random_hex + f_ext
    picture_path = os.path.join(app.root_path, 'static/profile_pics', picture_fn)

    output_size = (125, 125)
    i = Image.open(form_picture)
    i.thumbnail(output_size)
    i.save(picture_path)

    return picture_fn

@app.route("/account",methods=["GET","POST"])
@login_required
def account():
    form=UpdateAccountForm()
    if form.validate_on_submit():
        if form.picture.data:
            picture_file = save_picture(form.picture.data)
            current_user.image_file = picture_file
        current_user.username = form.username.data
        current_user.email = form.email.data
        db.session.commit()
        flash('Your account has been updated!', 'success')
        return redirect(url_for('account'))
    elif request.method == 'GET':
        form.username.data = current_user.username
        form.email.data = current_user.email
    image_file = url_for('static', filename='profile_pics/' + current_user.image_file)
    return render_template('account.html',title='Account',image_file =image_file,form=form)
```

Models:

```
from web import db , login_manager
from flask_login import UserMixin

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

class User(db.Model,UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
```

```
    image_file = db.Column(db.String(20), nullable=False, default='default.jpg')
    password = db.Column(db.String(60), nullable=False)


    def __repr__(self):
        return f"User('{self.username}', '{self.email}', '{self.image_file}')"
```

Forms:

```
from flask_wtf import FlaskForm
from flask_wtf.file import FileField,FileAllowed
from flask_login import current_user
from wtforms import  StringField,PasswordField,SubmitField,BooleanField
from wtforms.validators import DataRequired,Length,Email,EqualTo,ValidationError
from web.model import User

class RegistrationForm(FlaskForm):
    username= StringField('Username', validators=[DataRequired(),Length(min=3,max=20)])
    email=StringField('Email',validators=[DataRequired(),Email()])
    password=PasswordField('Password',validators=[DataRequired()])
    confirm_password=PasswordField('Confirm
Password',validators=[DataRequired(),EqualTo('password')])
    submit=SubmitField('sign up')
    def validate_username(self, username):
        user = User.query.filter_by(username=username.data).first()
        if user:
            raise ValidationError('That username is taken. Please choose a different one.')
    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user:
            raise ValidationError('That email is taken. Please choose a different one.')

class LoginForm(FlaskForm):
    username= StringField('Username', validators=[DataRequired(),Length(min=3,max=20)])
    password=PasswordField('Password',validators=[DataRequired()])
    remember=BooleanField('Remember me')
    submit=SubmitField('Login')

class UpdateAccountForm(FlaskForm):
    username= StringField('Username', validators=[DataRequired(),Length(min=3,max=20)])
    email=StringField('Email',validators=[DataRequired(),Email()])
    picture=FileField("Update Profile pic",validators=[FileAllowed(['jpg','png'])])
    submit=SubmitField('Update')
```

```
def validate_username(self, username):
    if username.data != current_user.username:
        user = User.query.filter_by(username=username.data).first()
        if user:
            raise ValidationError('That username is taken. Please choose a different one.')
def validate_email(self, email):
    if email.data != current_user.email:
        user = User.query.filter_by(email=email.data).first()
        if user:
            raise ValidationError('That email is taken. Please choose a different one.')
```

XG-BOOST:

```
import pandas as pd
car=pd.read_csv('/content/cars.csv')
car["Price"]=car["Price"].str.replace(",","")
car["Kms"]=car["Kms"].str.replace(",","")
car["Kms"]=car["Kms"].str.replace("km","")
car["Price"] = car["Price"].astype("int")
car["Kms"] =car["Kms"].astype("int")
```

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
#from jupyterthemes import jtplot
#jtplot.style(theme = 'monokai', context = 'notebook', ticks = True, grid
= False)
from plotly.offline import download_plotlyjs, init_notebook_mode, plot,
iplot
```

```
X=car[['Name','Year','Kms','Fuel','Type','Ownership','Model']]
y=car['Price']
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score
from xgboost import XGBRegressor
from sklearn.linear_model import LinearRegression
ohe=OneHotEncoder()
ohe.fit(X[['Name','Fuel','Type','Model']])
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categor
ies_),['Name','Fuel','Type','Model']),
                                     remainder='passthrough')
```

```
xgb = XGBRegressor()

pipe=make_pipeline(column_trans,xgb)

pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

0.9435935470308678

```
scores=[]
for i in range(1000):

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_st
ate=i)
    xgb = XGBRegressor()
    pipe=make_pipeline(column_trans,xgb)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

```
scores[np.argmax(scores)]
```

0.9835988659097019

```
pipe.predict(pd.DataFrame(columns=X_test.columns,data=np.array(['Toyota
Corolla Altis',13,69278 ,'Petrol','Manual',2,'G PETROL']).reshape(1,7)))
```

array([270040.72], dtype=float32)

```
import pickle
pickle.dump(pipe,open('XGBmodel.pkl','wb'))
pipe.predict(pd.DataFrame(columns=['Name','Year','Kms','Fuel','Type','Owne
rship','Model'],data=np.array(['Toyota Corolla Altis',13,69278
,'Petrol','Manual',2,'G PETROL']).reshape(1,7)))
```

array([270040.72], dtype=float32)

COURSE OUTCOMES:

After completion of the course the student will be able to

SL. NO	DESCRIPTION	Blooms' Taxonomy Level
CO1	Identify technically and economically feasible problems (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO2	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO3	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions of minimal complexity by using modern tools & advanced programming techniques (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO4	Prepare technical report and deliver presentation (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO5	Apply engineering and management principles to achieve the goal of the project (Cognitive Knowledge Level: Apply)	Level 3: Apply

CO-PO AND CO-PSO MAPPING

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO3
CO1	3	3	3	3		2	2	3	2	2	2	3	2	2	2
CO2	3	3	3	3	3	2		3	2	3	2	3	2	2	2
CO3	3	3	3	3	3	2	2	3	2	2	2	3			2
CO4	2	3	2	2	2			3	3	3	2	3	2	2	2
CO5	3	3	3	2	2	2	2	3	2		2	3	2	2	2

3/2/1: high/medium/low

JUSTIFICATIONS FOR CO-PO MAPPING

MAPPING	LOW/ MEDIUM/ HIGH	JUSTIFICATION
100003/CS622T.1-PO1	**HIGH**	Identify technically and economically feasible problems by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.1-PO2	**HIGH**	Identify technically and economically feasible problems by analysing complex engineering problems reaching substantiated conclusions using first principles of mathematics.
100003/CS622T.1-PO3	**HIGH**	Design solutions for complex engineering problems by identifying technically and economically feasible problems.
100003/CS622T.1-PO4	**HIGH**	Identify technically and economically feasible problems by analysis and interpretation of data.
100003/CS622T.1-PO6	**MEDIUM**	Responsibilities relevant to the professional engineering practice by identifying the problem.
100003/CS622T.1-PO7	**MEDIUM**	Identify technically and economically feasible problems by understanding the impact of the professional engineering solutions.
100003/CS622T.1-PO8	**HIGH**	Apply ethical principles and commit to professional ethics to identify technically and economically feasible problems.
100003/CS622T.1-PO9	**MEDIUM**	Identify technically and economically feasible problems by working as a team.
100003/CS622T.1-PO10	**MEDIUM**	Communicate effectively with the engineering community by identifying technically and economically feasible problems.
100003/CS622T.1-P011	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles by selecting the technically and economically feasible problems.
100003/CS622T.1-PO12	**HIGH**	Identify technically and economically feasible problems for long term learning.
100003/CS622T.1-PSO1	**MEDIUM**	Ability to identify, analyze and design solutions to identify technically and economically feasible problems.
100003/CS622T.1-PSO2	**MEDIUM**	By designing algorithms and applying standard practices in software project development and Identifying technically and economically feasible problems.
100003/CS622T.1-PSO3	**MEDIUM**	Fundamentals of computer science in competitive research can be applied to Identify technically and economically feasible problems.
100003/CS622T.2-PO1	**HIGH**	Identify and survey the relevant by applying the knowledge of mathematics, science, engineering fundamentals.

100003/CS622T.2-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems get familiarized with software development processes.
100003/CS622T.2-PO3	**HIGH**	Design solutions for complex engineering problems and design based on the relevant literature.
100003/CS622T.2-PO4	**HIGH**	Use research-based knowledge including design of experiments based on relevant literature.
100003/CS622T.2-PO5	**HIGH**	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes by using modern tools.
100003/CS622T.2-PO6	**MEDIUM**	Create, select, and apply appropriate techniques, resources, by identifying and surveying the relevant literature.
100003/CS622T.2-PO8	**HIGH**	Apply ethical principles and commit to professional ethics based on the relevant literature.
100003/CS622T.2-PO9	**MEDIUM**	Identify and survey the relevant literature as a team.
100003/CS622T.2-PO10	**HIGH**	Identify and survey the relevant literature for a good communication to the engineering fraternity.
100003/CS622T.2-PO11	**MEDIUM**	Identify and survey the relevant literature to demonstrate knowledge and understanding of engineering and management principles.
100003/CS622T.2-PO12	**HIGH**	Identify and survey the relevant literature for independent and lifelong learning.
100003/CS622T.2-PSO1	**MEDIUM**	Design solutions for complex engineering problems by Identifying and survey the relevant literature.
100003/CS622T.2-PSO2	**MEDIUM**	Identify and survey the relevant literature for acquiring programming efficiency by designing algorithms and applying standard practices.
100003/CS622T.2-PSO3	**MEDIUM**	Identify and survey the relevant literature to apply the fundamentals of computer science in competitive research.
100003/CS622T.3-PO1	**HIGH**	Perform requirement analysis, identify design methodologies by using modern tools & advanced programming techniques and by applying the knowledge of mathematics, science, engineering fundamentals.
100003/CS622T.3-PO2	**HIGH**	Identify, formulate, review research literature for requirement analysis, identify design methodologies and develop adaptable & reusable solutions.

100003/CS622T.3-PO3	**HIGH**	Design solutions for complex engineering problems and perform requirement analysis, identify design methodologies.
100003/CS622T.3-PO4	**HIGH**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS622T.3-PO5	**HIGH**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools.
100003/CS622T.3-PO6	**MEDIUM**	Perform requirement analysis, identify design methodologies and assess societal, health, safety, legal, and cultural issues.
100003/CS622T.3-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts and Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS622T.3-PO8	**HIGH**	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions by applying ethical principles and commit to professional ethics.
100003/CS622T.3-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS622T.3-PO10	**MEDIUM**	Communicate effectively with the engineering community and with society at large to perform requirement analysis, identify design methodologies.
100003/CS622T.3-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering requirement analysis by identifying design methodologies.
100003/CS622T.3-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS622T.3-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and prior to that perform requirement analysis, identify design methodologies.
100003/CS622T.4-PO1	**MEDIUM**	Prepare technical report and deliver presentation by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.4-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by preparing technical report and deliver presentation.

100003/CS622T.4-PO3	**MEDIUM**	Prepare Design solutions for complex engineering problems and create technical report and deliver presentation.
100003/CS622T.4-PO4	**MEDIUM**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions and prepare technical report and deliver presentation.
100003/CS622T.4-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and Prepare technical report and deliver presentation.
100003/CS622T.4-PO8	**HIGH**	Prepare technical report and deliver presentation by applying ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
100003/CS622T.4-PO9	**HIGH**	Prepare technical report and deliver presentation effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS622T.4-PO10	**HIGH**	Communicate effectively with the engineering community and with society at large by prepare technical report and deliver presentation.
100003/CS622T.4-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work by prepare technical report and deliver presentation.
100003/CS622T.4-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by prepare technical report and deliver presentation.
100003/CS622T.4-PSO1	**MEDIUM**	Prepare a technical report and deliver presentation to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas.
100003/CS622T.4-PSO2	**MEDIUM**	To acquire programming efficiency by designing algorithms and applying standard practices in software project development and to prepare technical report and deliver presentation.
100003/CS622T.4-PSO3	**MEDIUM**	To apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs by preparing technical report and deliver presentation.
100003/CS622T.5-PO1	**HIGH**	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.5-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by applying engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PO3	**HIGH**	Apply engineering and management principles to achieve the goal of the project and to design solutions for complex engineering problems and design system components or processes that meet the specified needs.
100003/CS622T.5-PO4	**MEDIUM**	Apply engineering and management principles to achieve the goal of the project and use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS622T.5-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO6	**MEDIUM**	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities by applying engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts, and apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO8	**HIGH**	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice and to use the engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO1	**MEDIUM**	The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas. Apply engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PSO2	**MEDIUM**	The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur and apply engineering and management principles to achieve the goal of the project.