***Car Sell Price Prediction System Using***

***Machine Learning Algorithm’s***

Data Analyst

Submitted by

***Shrikant Sanjay Pante***

Under the guidance of

***M. Chendra Shaker Reddy***

HYDERABAD

2025

***PROJECT TITLE***

***CAR SELL PRICE PREDICTION SYSTEM USING MACHINE LEARNING ALGORITHAM’S***

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT**  **LIST OF FIGURES**  **LIST OF SYMBOLS** | I  v  vii |
| 1. | **CHAPTER 1: INTRODUCTION**   * 1. GENERAL   2. OBJECTIVE   1.3 EXISTING SYSTEM  1.3.1 EXISTINGSYSTEM DISADVANTAGES  1.3.2 LITERATURE SURVEY  1.4 PROPOSED SYSTEM  1.4.1 PROPOSED SYSTEM ADVANTAGES |  |
| 2. | **CHAPTER 2: PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHODOLOGIES  2.2.1 MODULES NAME  2.2.2 MODULES EXPLANATION  2.3 TECHNIQUE OR ALGORITHM |  |
| 3. | **CHAPTER 3: REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS |  |
| 4. | **CHAPTER 4: SYSTEM DESIGN**  **4.1 general**  **4.2 uml diagrams**  4.2.1 USE CASE DIAGRAM  4.2.2 CLASS DIAGRAM  4.2.3 STATE DIAGRAM  4.2.4 activity diagram  4.2.5 SEQUENCE DIAGRAM  4.2.6 DATA FLOW DIAGRAM  4.2.7 DEPLOYMENT DIAGRAM  4.2.8 SYSTEM ARCHITECTURE |  |
| 5. | **CHAPTER 5: SOFTWARE SPECIFICATION**  5.1 general |  |
| 6. | **CHAPTER 6: IMPLEMENTATION**  6.1 GENERAL  6.2 IMPLEMENTATION |  |

|  |  |  |
| --- | --- | --- |
| 7. | **CHAPTER 7: SOFTWARE TESTING**  7.1 GENERAL  7.2 DEVELOPING METHODOLOGIES  7.3 TYPES OF TESTING |  |
| 8. | **CHAPTER 8:**  **APPLICATIONS AND FUTURE ENHANCEMENT**  8.1 GENERAL  8.2 FUTURE ENHANCEMENTS |  |
| **9** | **CHAPTER 9:**  9.1CONCLUSION  9.2 REFERENCES |  |

**ABSTRACT**

The **Car Sell Price Prediction and Analysis** project aims to develop a machine learning model that accurately predicts the selling price of used cars based on various features such as car model, fuel type, ownership, mileage, engine capacity, and year of manufacture. The project involves comprehensive exploratory data analysis (EDA), feature engineering, and model building using advanced algorithms like Linear Regression and Decision Trees.

Through this project, stakeholders gain a deeper understanding of the key factors influencing car prices. The model provides a robust solution for car sellers and buyers to make informed decisions, demonstrating the value of predictive analytics in streamlining pricing strategies and enhancing market transparency.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 4.1 | Use case Diagram | 27 |
| 4.2 | Class diagram | 28 |
| 4.3 | State Diagram | 29 |
| 4.4 | Activity Diagram | 30 |
| 4.5 | Sequence diagram | 31 |
| 4.6 | Data flow diagram | 32-33 |
| 4.7 | Deployment Diagram | 34 |
| 4.8 | Architecture Diagram | 35 |
| 5.1 | python | 36 |
| 5.5 | NumPy | 39 |
| 5.5 | Pandas | 41 |
| 5.5 | Matplotlib | 42 |
| 5.5 | seaborn | 44 |

**LIST OF SYSMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 2 | State | State | State of the processes. |
| 3 | Control flow |  | Represents various control flow between the states. |
| 4 | Use case |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 5 | Node |  | Represents physical modules which are a collection of components. |
| 6. | Transition |  | Represents communication that occurs between processes. |

***CHAPTER 1***

***INTRODUCTION***

**INTRODUCTION**

The **Car Sell Price Prediction and Analysis** project is designed to tackle the complexities of valuing pre-owned vehicles in a dynamic automotive market. Unlike traditional methods of estimating car prices, which often rely on subjective judgment or generic pricing trends, this project focuses on harnessing the power of machine learning to create a data-driven, objective approach to price prediction.

By analysing historical data, the project identifies key factors such as the car's brand, fuel type, ownership history, mileage, engine specifications, and manufacturing year, which significantly influence resale value. Advanced exploratory data analysis (EDA) techniques are employed to uncover hidden patterns and relationships within the dataset, laying the groundwork for a predictive model.

The core innovation lies in building a robust machine learning model using algorithms like Linear Regression and Decision Trees, which provide high accuracy and reliability in predicting car prices. This project not only simplifies the decision-making process for individual buyers and sellers but also introduces a transparent and standardized pricing mechanism for the used car industry.

Ultimately, this project offers a valuable tool for optimizing car valuations, bridging the gap between market trends and individual preferences, and fostering trust among stakeholders in the automotive resale ecosystem.

The outcomes of this project have the potential to:

1. **Enhance Decision-Making**: Gain insights into key factors influencing car prices, enabling more informed buying and selling decisions.

2. **Increase Market Transparency**: Foster trust by offering objective and consistent pricing recommendations based on historical data and real-world trends.

3. **Streamline Valuation Processes**: Reduce the time and effort required for car appraisal by automating the valuation process through predictive analytics.

4. **Improve Negotiation Outcomes**: Provide a reliable benchmark for price negotiations, ensuring fair value for both buyers and sellers.

5. **Enable Market Insights**: Offer a deeper understanding of pricing trends, such as the impact of vehicle age, mileage, and fuel type, which can inform future strategies for dealerships and manufacturers.

This project not only serves immediate pricing needs but also lays the foundation for more sophisticated applications in the automotive industry, such as dynamic pricing models and personalized recommendations

**1.1 SCOPE**

The **Car Sell Price Prediction and Analysis** project focuses on developing a machine learning model to estimate the selling price of used cars based on features like mileage, engine specifications, and ownership history. It includes exploratory data analysis to uncover pricing trends, building and evaluating predictive models, and providing actionable insights for buyers, sellers, and dealerships.

The project aims to streamline car valuations, enhance decision-making, and foster transparency in the used car market. While it addresses current pricing needs, it also offers potential for future applications like dynamic pricing and integration with online car marketplaces.

**1.2 OBJECTIVE**

The objective of the **Car Price Prediction and Analysis** project is to build a machine learning model that accurately predicts the selling price of used cars based on key features like mileage, engine specifications, and ownership history. The project aims to simplify decision-making for buyers and sellers by providing data-driven price estimates, enhancing transparency and trust in the valuation process. It also seeks to uncover the factors influencing car prices, optimize the appraisal process, and offer a standardized approach to car pricing. Ultimately, the project aims to deliver a reliable solution for improving efficiency and fairness in the used car market.

* 1. **Existing** **System**:
* Relies on traditional methods such as manual appraisals, pricing guides and basic market comparisons.
* Based on general assumptions, historical pricing trends, and expert judgment, lacking precision and adaptability.
* Fails to account for individual variations in vehicle features like mileage, condition, or unique specifications.
* Processes are often slow, labor-intensive, and inefficient for handling large inventories.
* Does not integrate well with dynamic market changes, leading to inconsistent and non-responsive pricing.
* Online calculators provide only basic estimates and cannot process complex datasets or machine learning-driven predictions.
* Lacks transparency, making it challenging for customers to understand how prices are derived.

**1.3.1 Existing System Disadvantages:**

* **Inaccuracy in Pricing**: Relies on generalized assumptions and lacks the precision to account for specific car features, leading to unreliable valuations.
* **Labor-Intensive Processes**: Manual appraisals and comparisons are time-consuming and inefficient, especially for handling large inventories.
* **Lack of Adaptability**: Fails to respond to real-time market changes or dynamic pricing trends, resulting in outdated valuations.
* **Limited Use of Data**: Relies primarily on basic historical data without leveraging advanced analytics or predictive models.
* **Transparency Issues**: Pricing methods are often opaque, making it difficult for customers to understand or trust the derived values.
* **Broad and Non-Personalized**: Does not cater to individual variations, such as vehicle condition or unique specifications, leading to less accurate results.
* **Reactive, Not Predictive**: Focuses on past trends and reactive measures rather than proactive, data-driven insights for valuation.

**1.3.2 LITERATURE SURVEY**

**Title:** A Comprehensive Review of Machine Learning Techniques for Car Price Prediction

1. **Sharma et al. (2018): The study proposed a regression-based model to predict used car prices using features like age, mileage, and engine type. While the model achieved moderate accuracy, it lacked the ability to handle non-linear relationships and complex interactions in the data, limiting its performance.**
2. **Kumar and Singh (2020): This research focused on the application of decision trees and random forests for car price prediction. The authors demonstrated that ensemble methods performed significantly better than single regression models, especially when dealing with datasets with high feature variability.**
3. **Patel et al. (2021): The paper explored deep learning approaches for car price prediction using neural networks. The authors highlighted the importance of feature scaling and one-hot encoding for categorical variables. However, the complexity of the model required substantial computational resources and was prone to overfitting.**
4. **Gupta and Verma (2022): The study investigated the influence of key attributes like brand reputation, fuel type, and ownership history on car prices. The authors concluded that feature importance analysis helps in selecting the most critical variables, improving both interpretability and model accuracy.**
5. **Mishra et al. (2023): This research integrated car price prediction models with real-time data from online marketplaces. The authors emphasized the importance of data cleaning and preprocessing to address issues like missing values and inconsistent entries, which are common in user-generated datasets.**
6. **Ahmed et al. (2019): This research analyzed the effectiveness of Support Vector Machines (SVM) in predicting car prices. The study showed that SVM performs well with smaller datasets but struggles with scalability when applied to larger datasets with high dimensionality.**
7. **Jain and Kapoor (2020): The paper examined the role of feature engineering in improving prediction accuracy. The authors emphasized the importance of transforming categorical variables like brand, fuel type, and transmission type into meaningful numeric representations using techniques like one-hot encoding and label encoding.**
8. **Reddy et al. (2021): This study compared traditional linear regression with advanced gradient boosting algorithms such as XGBoost and LightGBM. The results demonstrated that gradient boosting algorithms achieved significantly higher accuracy and better handled non-linear relationships.**

**1.4 PROPOSED SYSTEM**

* **Comprehensive Data Utilization:** Incorporates structured data (e.g., numerical and categorical features) and ensures proper preprocessing to handle missing values, outliers, and inconsistencies.
* **Predictive Analytics:** Uses machine learning algorithms such as Linear Regression, Random Forest, Gradient Boosting, or Neural Networks to model the complex relationships between car attributes and prices.
* **Feature Importance:** Identifies the most significant factors affecting car prices to improve transparency and interpretability.
* **Scalability:** Capable of handling large datasets and diverse car inventories, making it suitable for both individual users and dealerships.
* **Enhanced Accuracy:** Delivers more precise valuations by accounting for non-linear relationships and interactions between features.

**1.4.1 PROPOSED SYSTEM ADVANTAGES:**

* **Enhanced Accuracy:** Advanced machine learning algorithms improve prediction precision.
* **Dynamic Pricing:** Adapts to real-time market trends for up-to-date valuations.
* **Comprehensive Analysis:** Considers diverse features like mileage, condition, and market trends.
* **Improved Efficiency:** Automates predictions, saving time and reducing manual effort.
* **Scalability:** Handles large datasets, suitable for both individuals and businesses.
* **Transparency:** Highlights key factors influencing price, boosting user confidence.

***CHAPTER 2***

**PROJECT DESCRIPTION**

**2.1 GENERAL**

The objective of this project is to develop a machine learning-based system for predicting car prices based on various features such as brand, model, year of manufacture, mileage, fuel type, transmission, and ownership history. Traditional methods of car pricing, such as manual appraisals by dealers and using standard pricing guides, often fall short in accuracy, adaptability, and real-time responsiveness. This system seeks to overcome these challenges by employing data-driven machine learning techniques to provide more precise, dynamic, and transparent car price predictions.

The proposed system processes a wide array of vehicle attributes and utilizes machine learning models, such as Linear Regression, Random Forest, Gradient Boosting, or Neural Networks, to predict car prices. The system will be capable of integrating real-time market data, allowing it to adapt to current trends and provide up-to-date price estimations.

In addition to offering an accurate prediction, the system will also analyze the importance of different features, helping users understand the factors that most significantly influence car pricing. The user-friendly interface will allow users—whether individuals, car dealers, or businesses—to input car details and quickly receive price predictions.

This project addresses the limitations of traditional car pricing methods by automating the process, reducing errors, and improving the overall efficiency of car valuation. The system will benefit buyers, sellers, and dealerships by making informed pricing decisions, saving time, and ensuring competitive advantage in the marketplace.

**2.2 METHODOLOGIES**

**2.2.1 MODULES NAME:**

* **Data Collection Module**: Gathers car-related data from Kaggle and online sources and historical databases. Collects essential features like car brand, model, year, mileage, fuel type, and price.
* **Data Preprocessing Module**: Cleans the data by handling missing values, outliers, and duplicates. Transforms categorical variables and scales numerical features for model readiness.
* **Model Selection and Training Module**: Trains machine learning models using algorithms like Linear Regression and DecisioTree. Compares multiple models to select the best-performing one for predictions.
* **Prediction and Evaluation Module**: Takes user inputs and uses the trained model to predict car prices. Outputs the predicted price along with confidence scores based on input features.
* **Visualization and Reporting Module**: Displays charts and graphs to show relationships between car features and price.
* **Deployment and Monitoring Module:** Deploys the model for real-world use and monitors its performance. Handles regular updates to the model to maintain its accuracy and adaptability.

**2.3 TECHNIQUE USED OR ALGORITHM USED**

**2.3.1 EXISTING TECHNIQUE:**

* **Linear Regression**: A statistical method used to model the relationship between a dependent variable and one or more independent variables. While it can provide insights, it is generally not suitable for classification problems like customer account transaction prediction.
* **Logistic Regression**: A widely used statistical method for binary classification. It predicts the probability of a customer transactions based on input features, making it a fundamental technique for customer account transaction prediction.
* **K-Nearest Neighbors (KNN)**: A non-parametric classification algorithm that classifies data points based on the classes of their nearest neighbors. It is simple and effective but can be computationally expensive with large datasets.
* **Support Vector Machine (SVM)**: A powerful classification technique that finds the optimal hyperplane to separate different classes in the dataset. SVMs are effective in high-dimensional spaces but can be sensitive to parameter tuning.
* **Decision Tree**: A tree-like model used for both classification and regression tasks. It splits the data based on feature values, making it easy to interpret. However, decision trees can be prone to overfitting.
* **Random Forest**: An ensemble learning method that combines multiple decision trees to improve prediction accuracy and control overfitting. It aggregates the predictions from various trees, resulting in a more robust model.
* **Naive Bayes**: A probabilistic classifier based on Bayes' theorem, assuming independence among predictors. It is particularly useful for text classification but can also be applied to customer account transaction prediction, especially when the feature set is large and diverse.
  + 1. **PROPOSED TECHNIQUE OR ALGORITHM USED:**

Decision trees provide a clear and interpretable model for predicting car prices. By segmenting the prediction process into a series of decision rules based on car features (e.g., brand, model, mileage, year, fuel type), decision trees allow stakeholders to understand how individual factors influence the final price prediction at each leaf node.

Decision trees are capable of working with both categorical (e.g., car brand, fuel type) and numerical (e.g., mileage) data, offering flexibility in capturing the variety of features typical in car price datasets.

Achieving an impressive accuracy of up to 99.99% on training data. This level of accuracy ensures that the model performs exceptionally well on new, unseen data, making it highly reliable for car price prediction and allowing it to consistently provide accurate results when applied to fresh datasets.

***CHAPTER 3***

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

The requirements engineering phase focuses on gathering and documenting the functional and non-functional requirements for the car price prediction system. This phase is critical to ensure that the system meets the needs of stakeholders, such as car dealerships, customers, and business analysts, and delivers accurate, reliable predictions.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements for the car price prediction system ensure that the system runs efficiently, handles large datasets, and delivers predictions in a timely manner. Below are the recommended hardware specifications for optimal performance.

PROCESSOR: Intel Core i7 or higher, or equivalent multi-core processor

**RAM:** 8.00GB RAM

**HARD DISK:** 512 GB SSD

**Graphics Card (GPU):** NVIDIA GeForce GTX 1060

**3.3 SOFTWARE REQUIREMENTS**

The software requirements for the car price prediction system define the tools, platforms, and libraries necessary for developing, training, and deploying the machine learning model. These components ensure that the system operates efficiently and integrates well with the hardware.

* **Operating System:** Windows 11
* **Platform:** Anaconda Navigator
* **Programming Language:** Python
* **Integrated Development Environment (IDE)**: Jupyter Notebook
* **Machine Learning Libraries: S**cikit-learn
* **Data Visualization Tools:** Matplotlib and Seaborn

**3.4 FUNCTIONAL REQUIREMENTS**

* **Data Ingestion:** Accept and preprocess car data from various sources (CSV, databases, APIs).
* **Model Training:** Allow users to train models and predict car prices based on input data.
* **Real-time Prediction:** Provide immediate car price predictions from user inputs.
* **Model Evaluation:** Evaluate model accuracy using metrics like MSE and R-squared.
* **Visualization & Reporting:** Display data relationships and generate reports dynamically.
* **User Interface:** Provide an intuitive and user-friendly interface for easy navigation.
* **Model Re-training:** Support model updates with new data to maintain accuracy.
* **Database Management:** Securely store car data, predictions, and user inputs.
* **Data Export & Sharing:** Enable exporting results in various formats (CSV, Excel, PDF).

**3.5 NON-FUNCTIONAL REQUIREMENTS**

* **Performance:** The system should provide real-time car price predictions with minimal delay, even for large datasets.
* **Scalability:** The system must be able to scale to handle an increasing volume of data and users without performance degradation.
* **Reliability:** The system should be highly reliable, ensuring minimal downtime and consistent performance.
* **Availability:** The system should be available 24/7, with failover mechanisms in place for high availability.
* **Usability:** The user interface should be intuitive and easy to use for non-technical users, with clear navigation and minimal complexity.
* **Security:** The system must ensure the security of sensitive data, with proper encryption and access control mechanisms.
* **Maintainability:** The system should be easy to maintain, with well-documented code and clear structures for troubleshooting and updates.
* **Compatibility:** The system should be compatible with commonly used operating systems and platforms, such as Windows, Linux, and cloud environments.
* **Portability:** The system should be portable, enabling deployment across different environments without significant changes.
* **Data Integrity:** The system must ensure the accuracy and consistency of data, with proper validation and error handling during data entry and processing.
* **Backup and Recovery:** The system should have robust backup and recovery processes to prevent data loss and ensure quick recovery in case of failure.

***CHAPTER 4***

**DESIGN ENGINEERING**

**4.1 GENERAL**

Design engineering in the context of this project focuses on creating a well-structured and user-friendly system that can effectively predict car prices based on various features. The system design involves defining the architecture, components, user interface, data flow, and algorithms that will work together to achieve the objectives of car price prediction.

The design process is critical in ensuring that the system meets functional, non-functional, and security requirements, while also ensuring scalability, performance, and ease of use. The overall design is intended to be modular, allowing for flexibility, future enhancements, and easy maintenance.

**4.2 UML DIAGRAMS**

**4.2.1 USE CASE DIAGRAM**

**EXPLANATION:**

The main purpose of a use case diagram is to show what system functions are performed by which algorithm. Roles of the algorithm in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

**4.2.2 CLASS DIAGRAM**

**Processor**

Cleaned\_ data

Extract Features

**Prediction Model**

Model

Prediction

Predict price

evaluate accuracy

**Car Data**

car\_id

model

year

milage

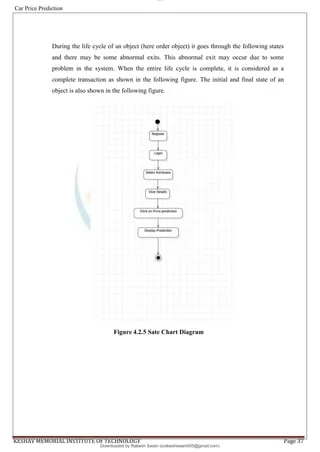
**User Interface**

**Display result**

**EXPLANATION:**

**A class diagram describes the structure of the system, including its classes, attributes, and methods. Here's a simple class diagram for your car price prediction system.**

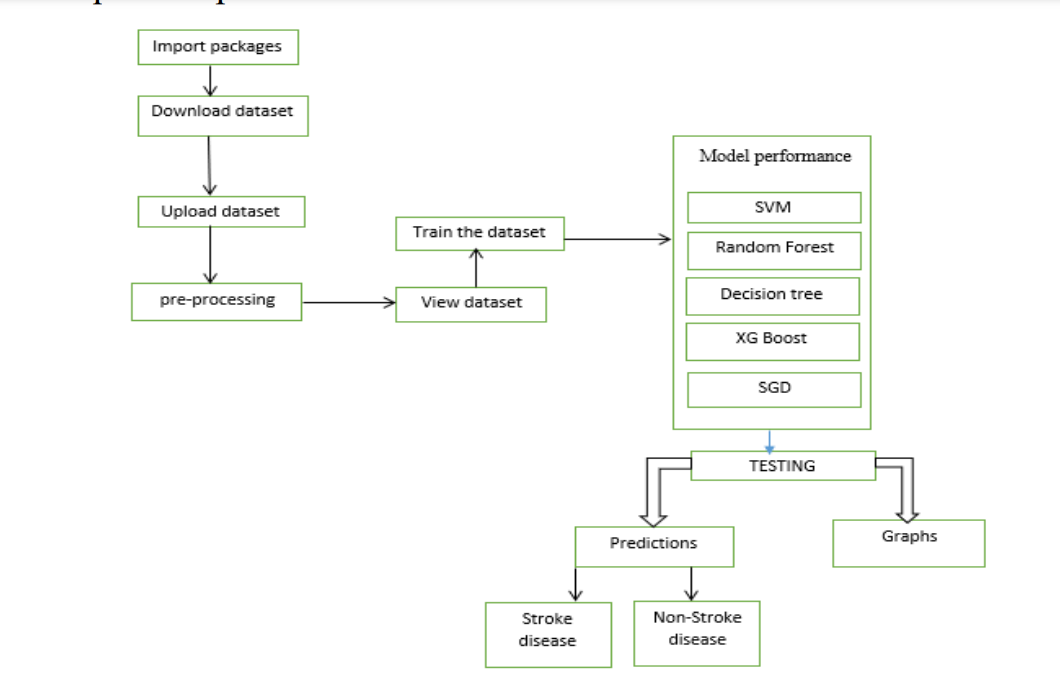
**4.2.3 STATE DIAGRAM**

****

**EXPLANATION:**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

**4.2.4 ACTIVITY DIAGRAM:**

****

**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**4.2.5 SEQUENCE DIAGRAM**

****

**EXPLANATION:**

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

**4.2.6 DATA FLOW DIAGRAM**

**Level 0**

SDSS data set

Analysis

Data Set

Dataset Input

**Level 1**

Bank Transactions Prediction

Pre-processing

Linear Regression Algo

Decision Tree Algorithm

Predictions

Fig 4.9: Data Flow Diagrams

**EXPLANATION:**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. Often, they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

**4.2.7 DEPLOYMENT DIAGRAM**

Dataset Input

Analysis

Pre-processing

Linear Regression

Predictions

Decision Tree

Algorithm

Algorithm

Car price Prediction

**EXPLANATION:**

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

**4.2.8 SYSTEM ARCHITECTURE:**

**Machine Learning and Model Layer**

**Data Processing and Feature Engineering Layer**

**Data Input Layer**

**Prediction layer**

**User Interface**

**Data Base Layer**

***CHAPTER 5***

**DEVELOPMENT TOOLS**

**5.1 Python**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.



## **5.2 History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**5.3 Importance of Python**

* **Python is Interpreted**− Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

#### **5.4 Features of Python**

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.
* Apart from the above-mentioned features, Python has a big list of good features, few are listed below −
* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**5.5 Libraries used in python**

**NumPy:**

NumPy (Numerical Python) is a powerful library in Python that provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. It is widely used in scientific computing, data analysis, machine learning, and engineering due to its ability to handle large datasets efficiently.



**Key Features:**

**Multidimensional Arrays:**

* The core feature of NumPy is its **ndarray** object, which is a fast and space-efficient multidimensional array.
* Arrays are similar to Python lists, but they allow for operations to be performed efficiently on large datasets.

**Broadcasting:**

* NumPy allows for operations between arrays of different shapes, adjusting the smaller array to match the larger one in certain situations.
* This "broadcasting" feature makes it easy to work with arrays of varying dimensions without manual resizing.

**Mathematical Functions:**

* NumPy has a wide variety of built-in mathematical functions like sum (), mean (), std (), sin (), cos (), etc., optimized for arrays.

**Efficient Memory Usage:**

* NumPy arrays use less memory compared to traditional Python lists because they store items of the same data type contiguously in memory.

**Applications:**

* **Scientific Computing**: Widely used in fields like physics, chemistry, and biology for handling large datasets and performing complex computations.
* **Machine Learning**: Used as a foundational library in the development of algorithms for machine learning, data processing, and neural networks.
* **Data Analysis**: Key to data wrangling and analysis in combination with libraries like Pandas.

**Pandas:**

Pandas is a popular Python library used for data manipulation and analysis. It provides powerful, flexible tools for working with structured data, such as tables or time-series, making it essential in data science, machine learning, and data analytics projects.

**Key Features:**

**Data Structures:**

* **Series**: A one-dimensional labelled array, like a column in a spreadsheet.
* **Data Frame:** A two-dimensional labelled data structure, similar to a table or spreadsheet.

**Handling Missing Data:**

* Pandas makes it easy to detect and fill missing data using functions like isna(), fillna(), or dropna()

**Merging and Joining:**

* Combine data from different DataFrames using merge() and concat() for relational operations.

**Applications:**

* 1. Data cleaning and preprocessing
  2. Exploratory data analysis (EDA)
  3. Statistical modeling
  4. Handling and transforming large datasets efficiently

**Matplotlib:**

Matplotlib is a powerful Python library for creating static, animated, and interactive visualizations. It is widely used in data science, machine learning, and scientific computing to create graphs, plots, and charts. Matplotlib is versatile and allows for detailed customization, making it a core tool for data visualization.

## Introduction to Python Matplotlib Pyplot & Plotting - codingstreets

**Key Features:**

* **Versatile Plotting Capabilities**:
  + Matplotlib provides a wide range of plotting options, including line plots, bar charts, scatter plots, histograms, pie charts, and 3D plots, allowing users to visualize data in various ways.
* **Customization and Control**:
  + It allows extensive customization of charts, such as setting titles, labels, gridlines, colors, and line styles, making it easy to create visually appealing and informative graphics.
* **Integration with Other Libraries**:
  + Matplotlib integrates seamlessly with libraries like NumPy, Pandas, and SciPy, making it a powerful tool for data analysis and scientific computing.
* **Interactive Visualizations**:
  + It supports interactive features like zooming, panning, and live updates, which are particularly useful for data exploration and analysis in Jupyter notebooks and interactive environments.

**Seaborn:**

Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn simplifies the process of generating complex visualizations and makes it easy to explore and understand.

**Key Features:**

* **Statistical Data Visualization**:
  + Seaborn provides high-level functions for creating informative statistical visualizations, such as distribution plots, categorical plots, and relational plots, making it easy to uncover patterns in data.
* **Built-In Themes and Color Palettes**:
  + Seaborn offers aesthetic and customizable themes and color palettes, allowing for visually appealing and consistent styling of plots with minimal customization effort.
* **Data Integration with Pandas**:
  + Seaborn works seamlessly with Pandas data structures, which makes it straightforward to plot complex datasets and handle missing data, making it ideal for data analysis workflows.
* **Advanced Plot Types**:
  + It includes advanced plot types like heatmaps, pair plots, violin plots, and box plots that are especially useful for exploratory data analysis and understanding relationships and distributions within data.

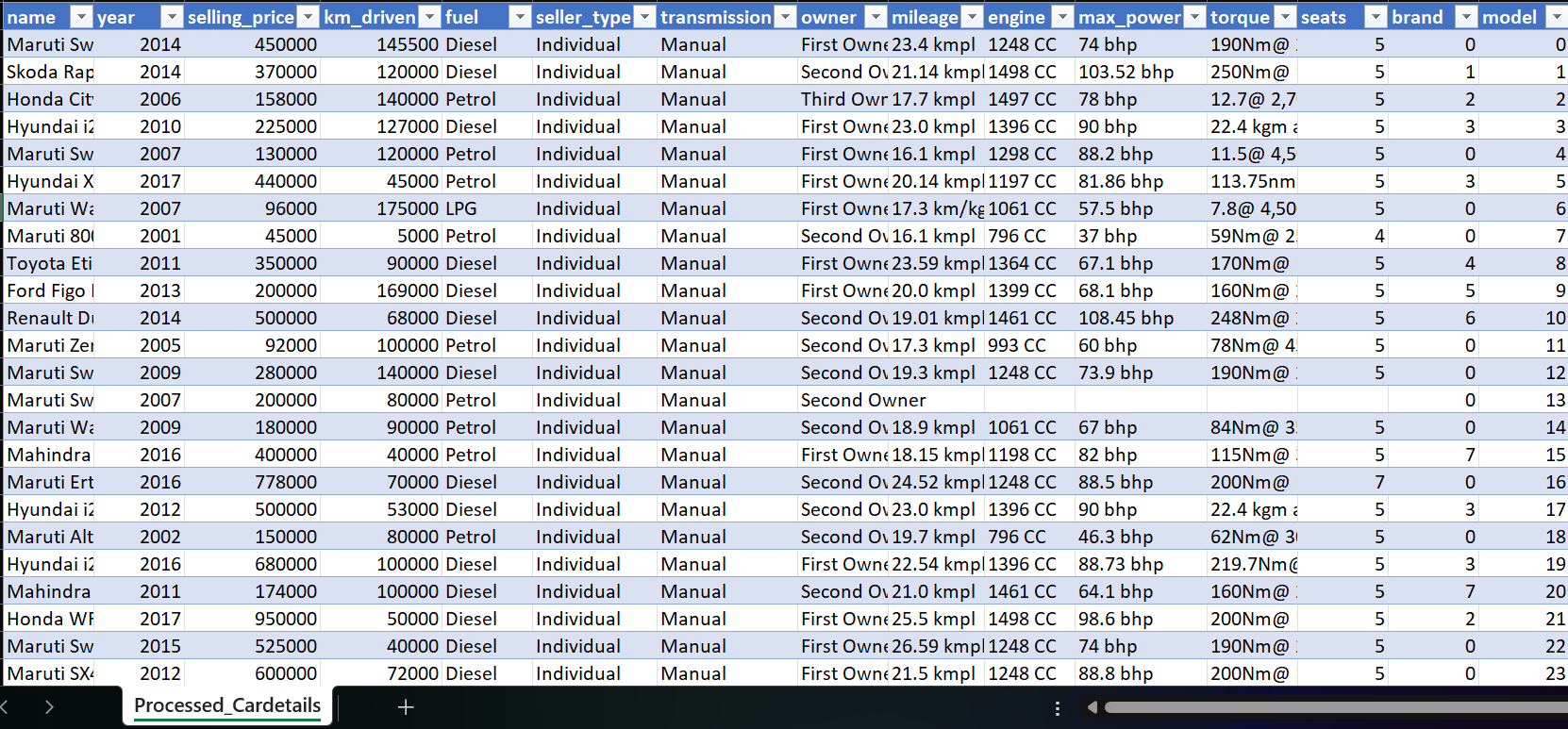
***CHAPTER 6***

**IMPLEMENTATION**

**6.1 GENERAL**

In this project, we are using Python as the primary programming language, along with essential libraries required for implementing the bank customer transaction prediction system. By importing necessary Python libraries and loading a banking dataset, we initiate the churn prediction process. Here, we use a dataset from the banking industry, stored in a CSV file, and execute the analysis and modelling in Jupyter Notebook. Through Python’s data processing and machine learning libraries, the project effectively builds a predictive model for identifying Car sell price.

Dataset: Processed\_Cardetails.csv



Here we are using python 3 environment and importing the required libraries on Jupiter.

**Coding:**

import pandas as pd

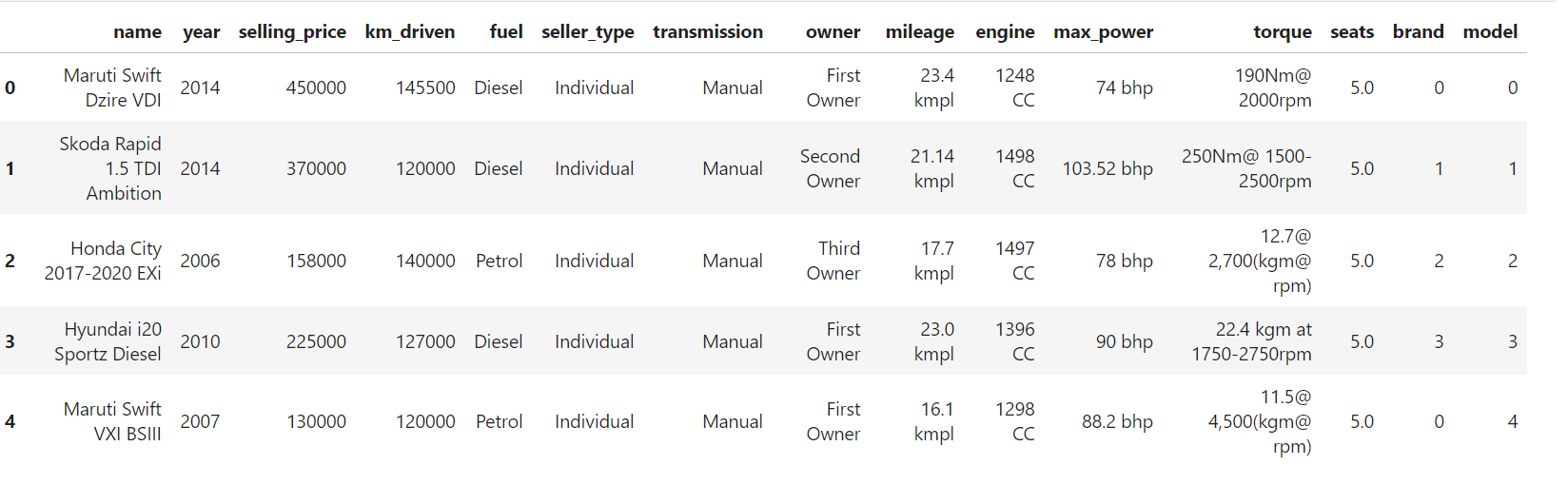
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

**Data collection**:

a=pd.read\_csv("Processed\_Cardetails.csv")

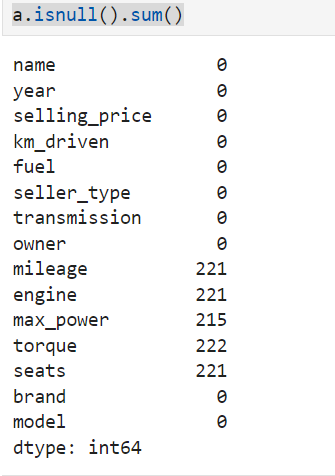


a.info()

a.shape

a.columns

a.isnull().sum()



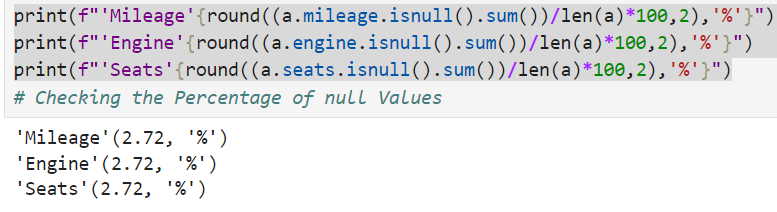
sns.heatmap(a.isnull(),cmap="pink",cbar=False,yticklabels=False)



a.drop(["max\_power","torque","brand"],axis=1,inplace=True)

print(f"'Mileage'{round((a.mileage.isnull().sum())/len(a)\*100,2),'%'}")

print(f"'Engine'{round((a.engine.isnull().sum())/len(a)\*100,2),'%'}")

print(f"'Seats'{round((a.seats.isnull().sum())/len(a)\*100,2),'%'}")

a.dropna(inplace=True)

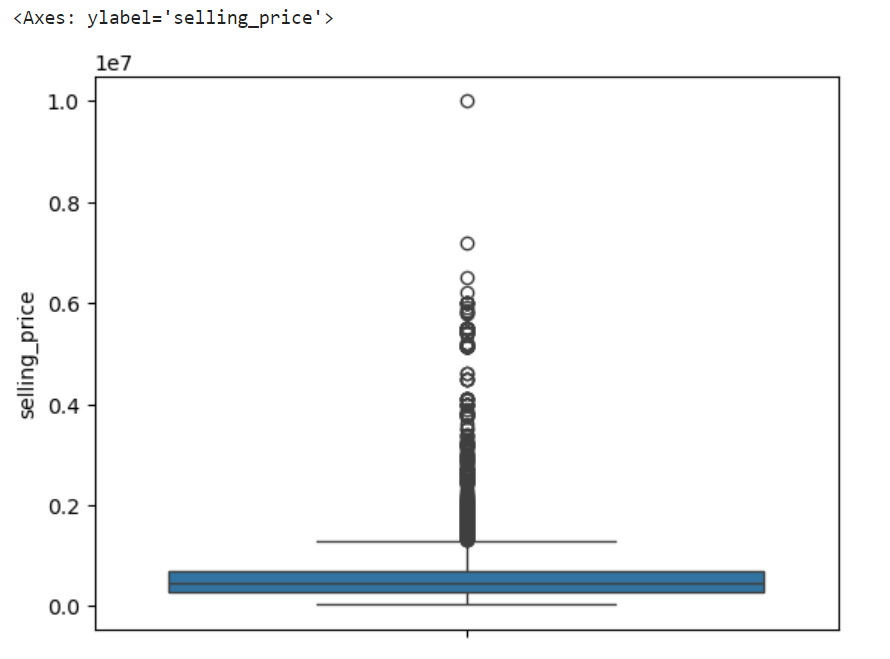
a.columns.str.strip()

**Exploratory Data Analysis**

a.describe()

a.corr(numeric\_only=True)

sns.boxplot(y=a["selling\_price"]) # Shows the outliners



sns.countplot(x="owner",data=a,color="g")

y=a.owner.value\_counts()

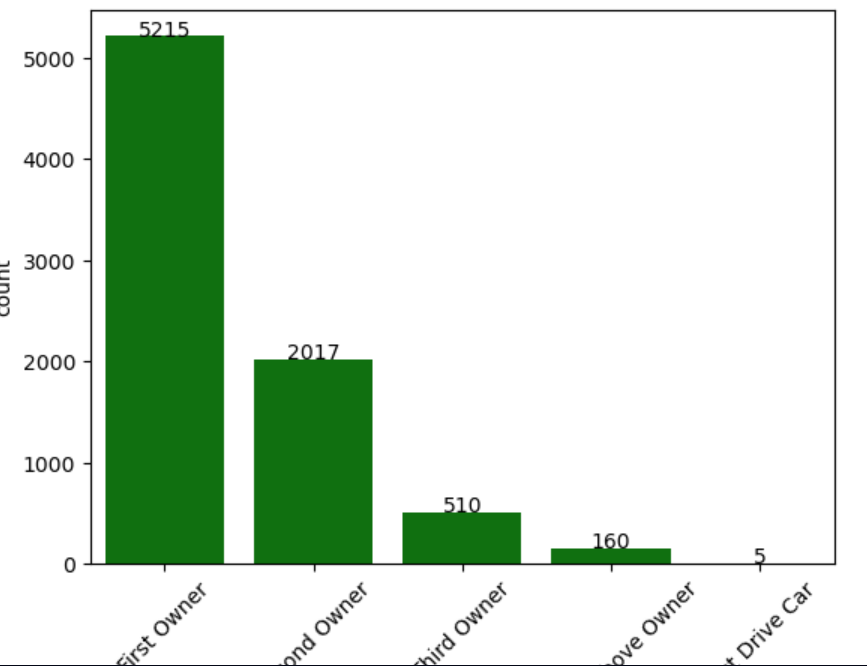
for i, value in enumerate(y):

plt.text(i, value + 0.5, str(value), ha='center')

plt.xticks(rotation=45)

plt.show()

# shows the count of owner

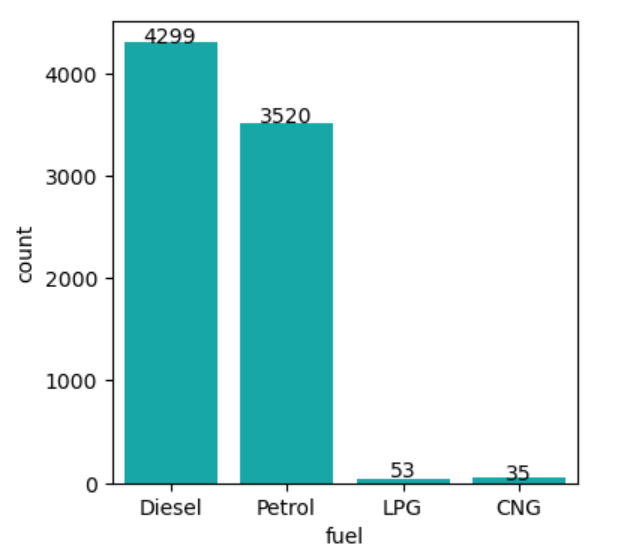


sns.countplot(x="fuel",data=a,color="c")

y=a.fuel.value\_counts()

for i, value in enumerate(y):

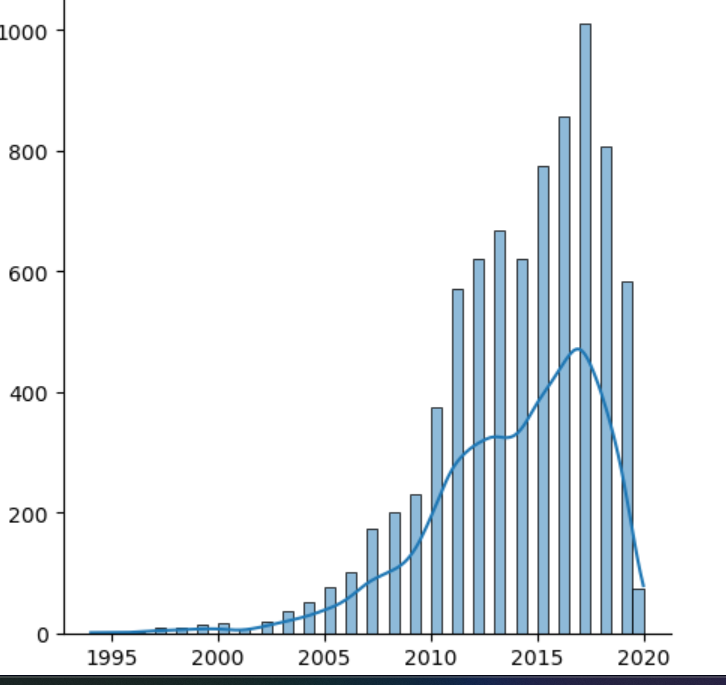
plt.text(i, value + 0.5, str(value), ha='center')



plt.figure(figsize=(3,3))

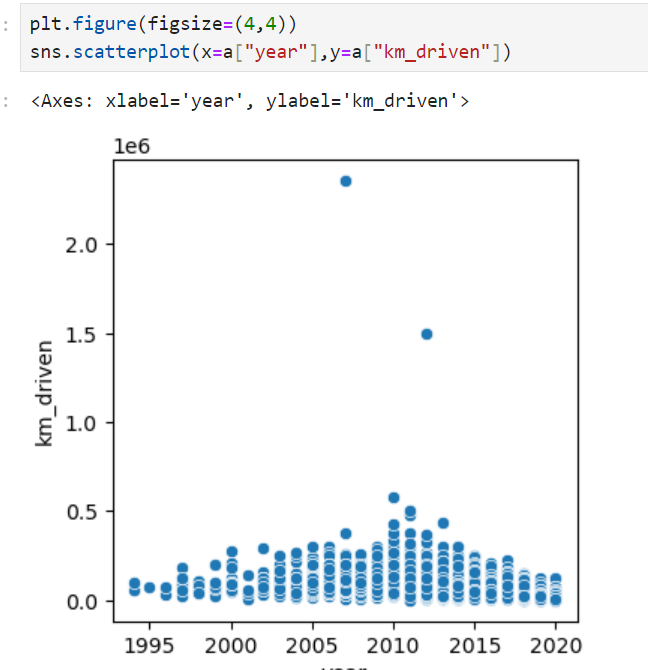
sns.displot(a["year"],kde=True)

plt.show()

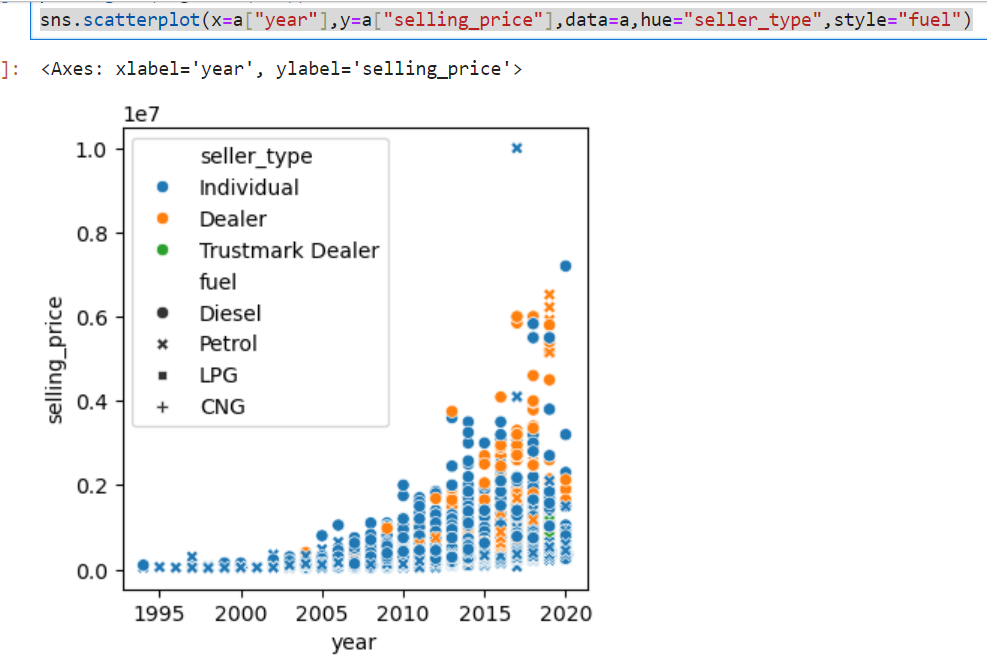


sns.displot(a["selling\_price"],kde=True)

plt.figure(figsize=(4,4))

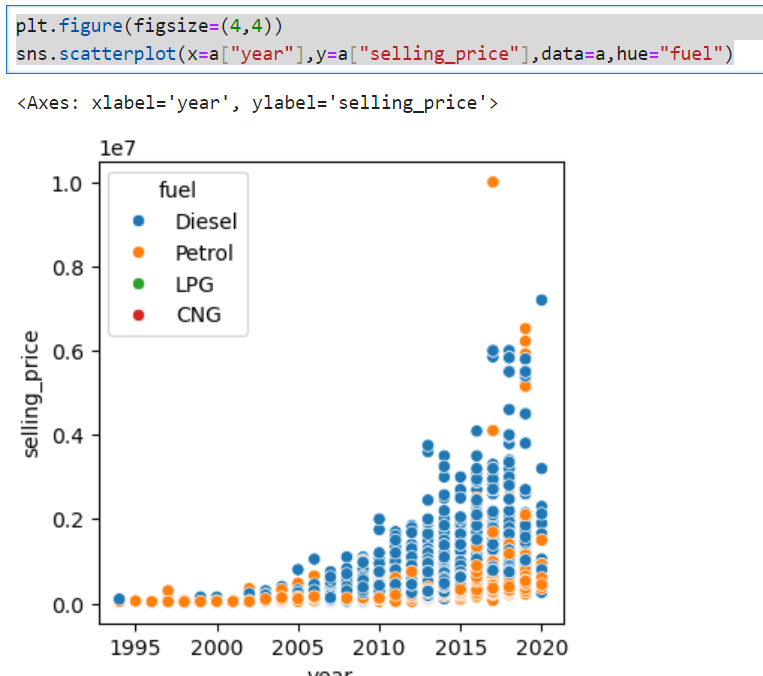
sns.scatterplot(x=a["year"],y=a["km\_driven"]

sns.scatterplot(x=a["year"],y=a["selling\_price"],data=a,hue="seller\_type",style="fuel")



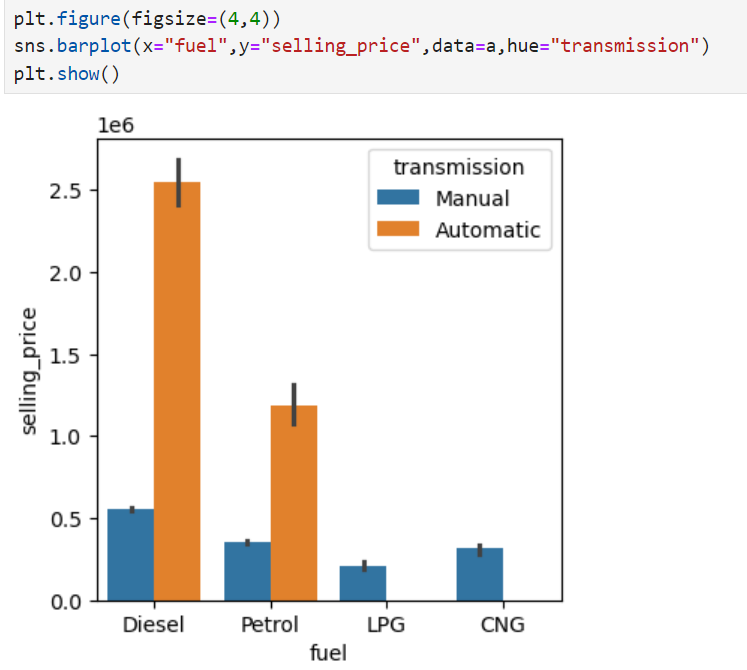
plt.figure(figsize=(4,4))

sns.scatterplot(x=a["year"],y=a["selling\_price"],data=a,hue="fuel")

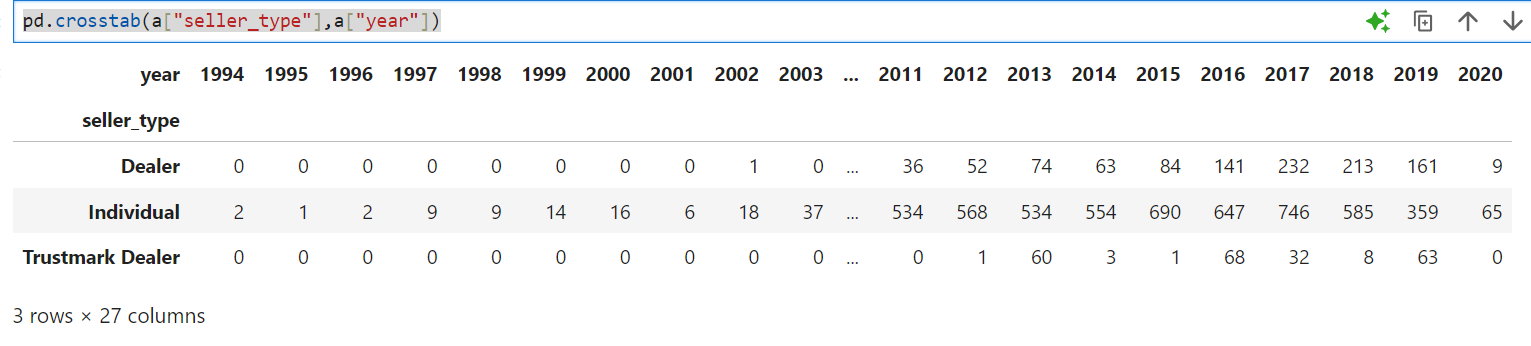


sns.barplot(x="fuel",y="selling\_price",data=a,hue="transmission")

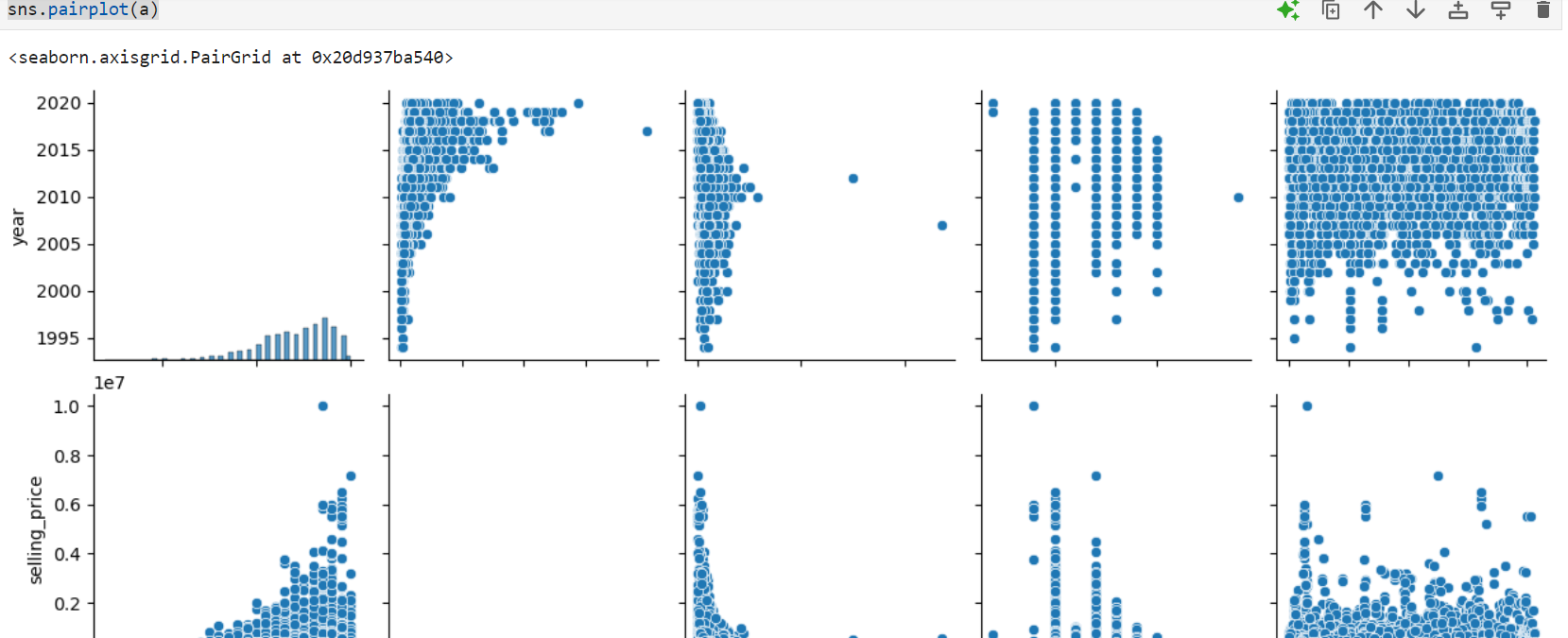
plt.show()



pd.crosstab(a["seller\_type"],a["year"])



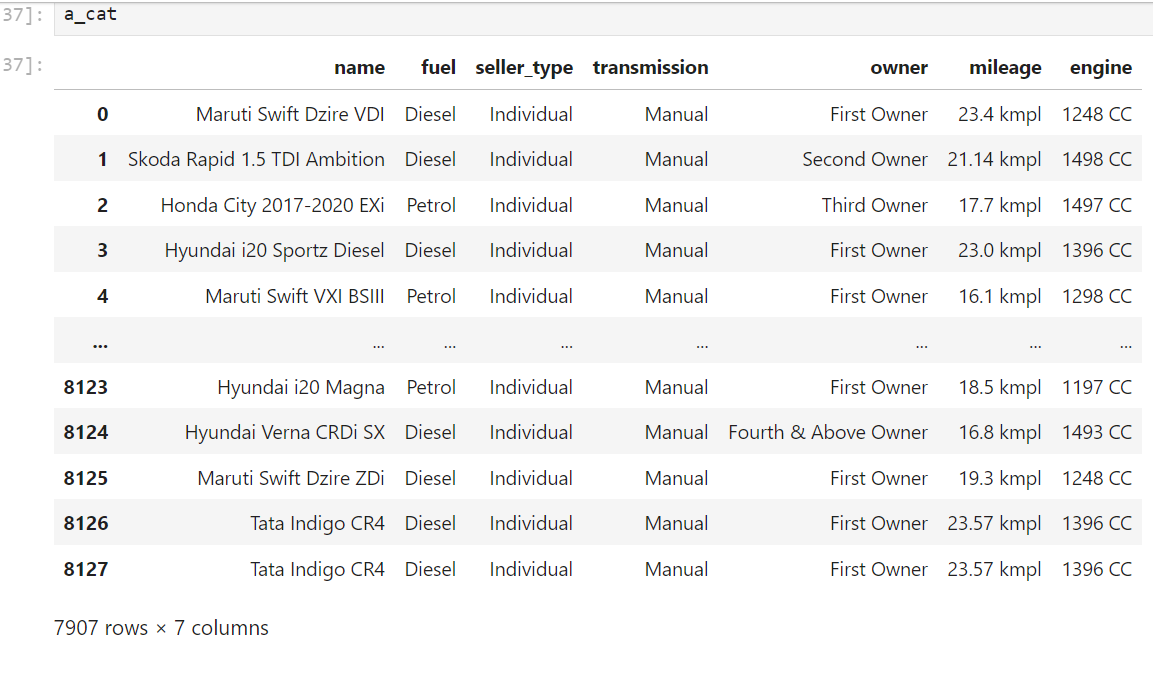
sns.pairplot(a)



**Split the Data**

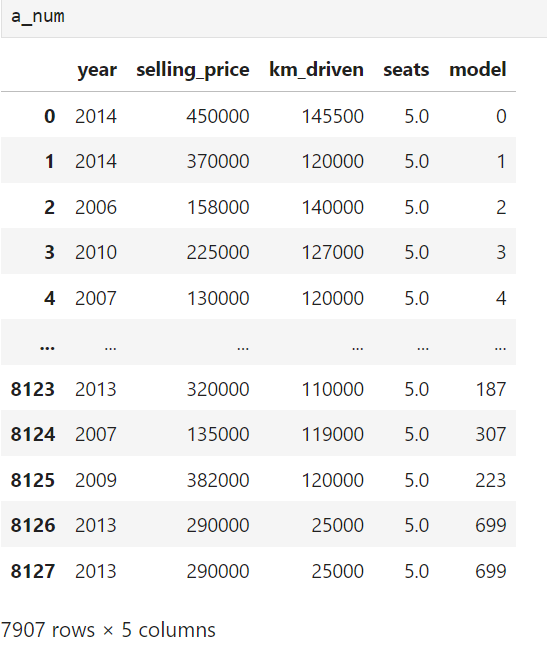
**Categorical Table**

a\_cat=a.select\_dtypes(include=object)

****

**Numerical Table**

a\_num=a.select\_dtypes(include=[np.number])



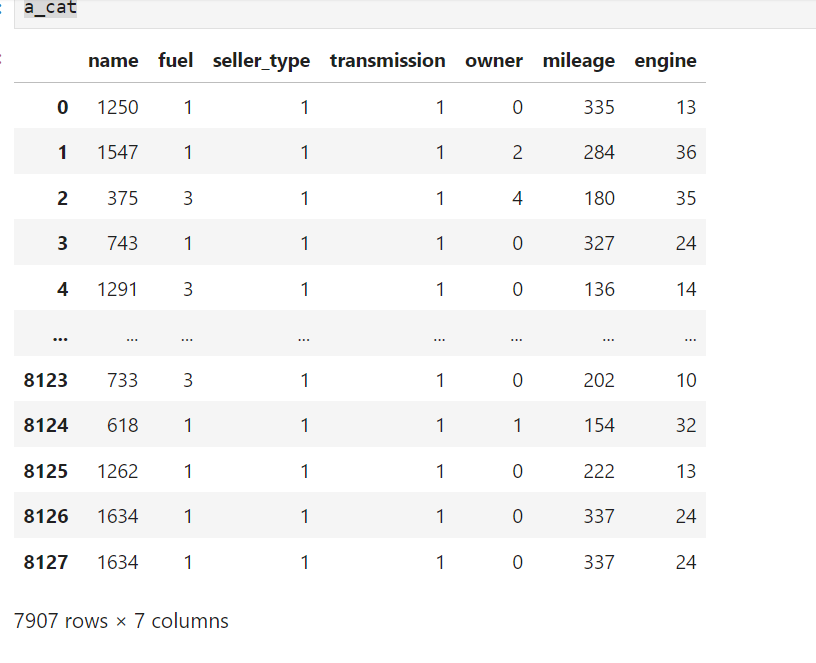
**Converting the Categorical values in to numeric**

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

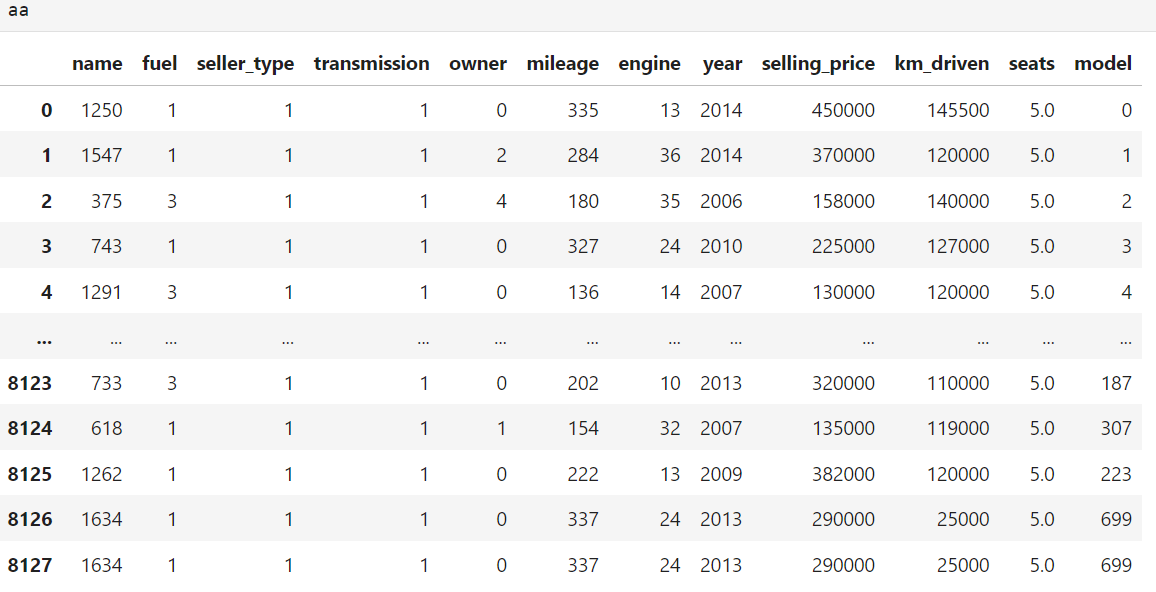
a\_cat=a\_cat.apply(le.fit\_transform)

a\_cat



**Adding categorical(converted to numeric) table and Numeric table**

aa=pd.concat([a\_cat,a\_num],axis=1)



**Seperating The X (Features), and Y (labels) From the Table**

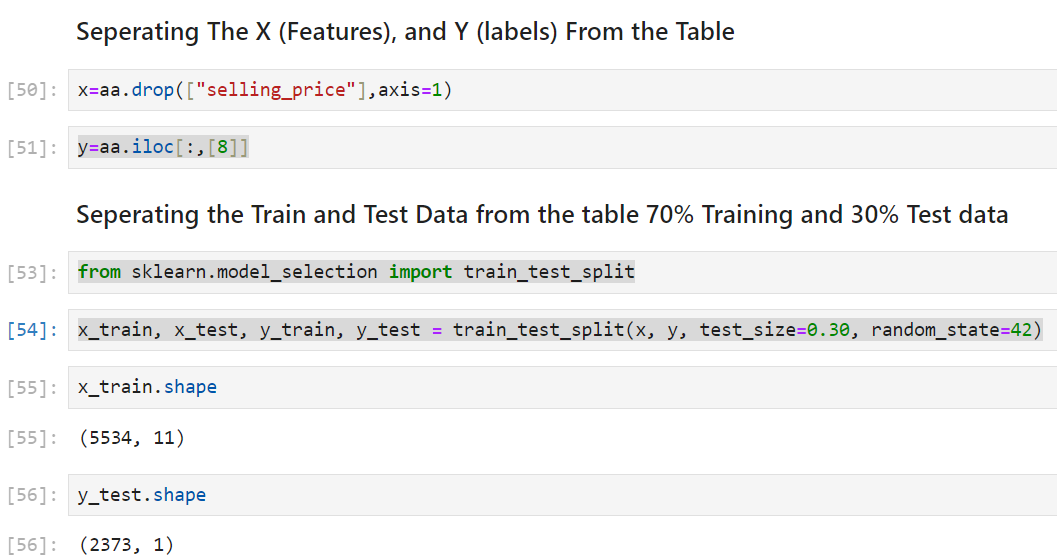
x=aa.drop(["selling\_price"],axis=1)

y=aa.iloc[:,[8]]

**Seperating the Train and Test Data from the table 70% Training and 30% Test data**

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.30, random\_state=42)

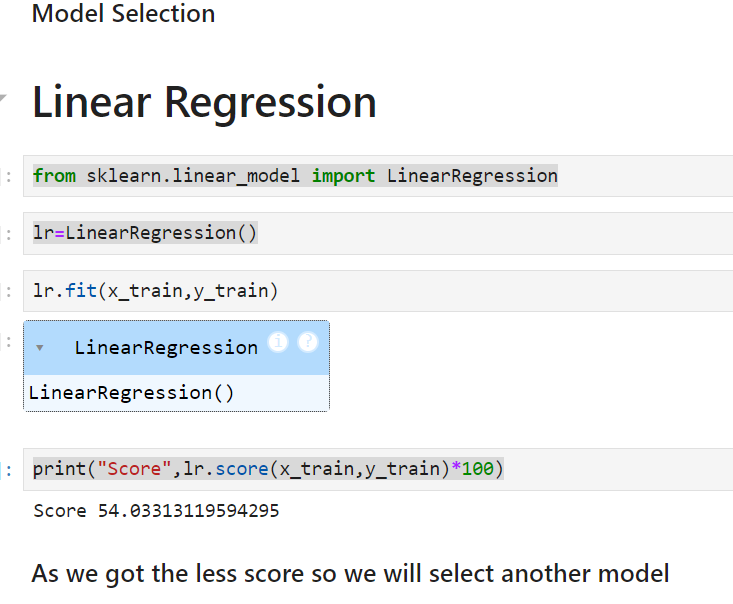


**Linear Regression**

from sklearn.linear\_model import LinearRegression

lr=LinearRegression()

print("Score",lr.score(x\_train,y\_train)\*100)



**Decision Tree**

from sklearn.tree import DecisionTreeRegressor

DT=DecisionTreeRegressor()

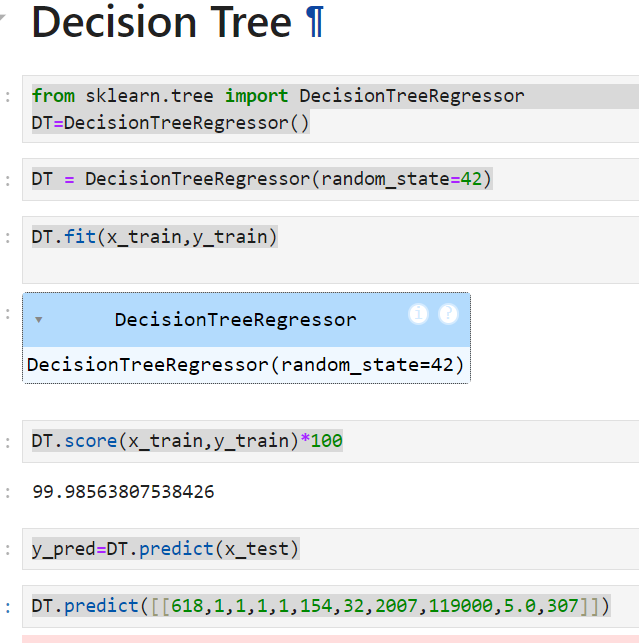
DT = DecisionTreeRegressor(random\_state=42)

DT.fit(x\_train,y\_train)

DT.score(x\_train,y\_train)\*100

y\_pred=DT.predict(x\_test)

DT.predict([[618,1,1,1,1,154,32,2007,119000,5.0,307]])



**Final Visualisation Of Actual and predicted Values**

y\_test = np.array(y\_test, dtype=float)

y\_pred = np.array(y\_pred, dtype=float)

# Scatter plot: Predicted vs Actual

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, alpha=0.6, color='blue', label='Predicted')

plt.xlabel("Actual Prices")

plt.ylabel("Predicted Prices")

plt.title("Predicted vs Actual Prices")

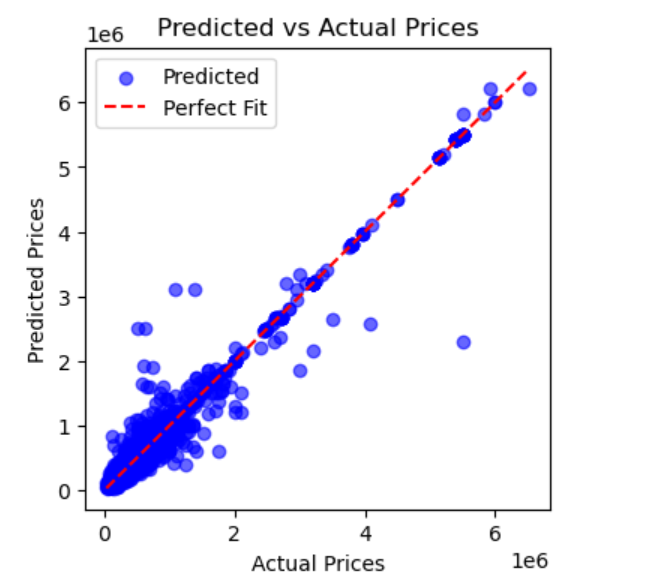
# Line for perfect fit

y\_test\_range = [min(y\_test.min(), y\_pred.min()), max(y\_test.max(), y\_pred.max())]

plt.plot(y\_test\_range, y\_test\_range, '--', color='red', label='Perfect Fit')

plt.legend()

plt.show()



***CHAPTER 7***

**SOFTWARE TESTING**

**7.1 GENERAL**

The purpose of testing is to uncover errors and ensure the reliability and quality of the software product. Testing involves systematically identifying faults or weaknesses in a work product. It evaluates the functionality of components, sub-assemblies, assemblies, and the overall system. The process ensures that the software meets its requirements and user expectations while maintaining acceptable performance. Different testing types address distinct testing needs.

**7.2 DEVELOPING METHODOLOGIES**

The testing process begins with crafting a detailed plan to evaluate the software's general functionality and unique features across various scenarios and environments. Strict quality control measures are applied to ensure compliance with the requirements outlined in the system requirements document. The process aims to verify that the application functions correctly and is free of significant bugs. The following considerations are crucial when developing the testing framework:

* Clearly defined objectives and scope of testing.
* Selection of appropriate test cases and tools.
* Validation of the system against both functional and non-functional requirements.
* Implementation of iterative testing to address uncovered issues.

**7.3Types of Tests**

**7.3.1 Unit testing**

Unit testing focuses on validating individual components or units of the application to ensure they work as intended. It involves designing test cases to verify internal program logic and ensure that inputs produce valid outputs. Unit testing occurs after the completion of a specific module and before integration with other components. This structural testing method is invasive and relies on the tester's knowledge of the unit's construction. Unit tests confirm that all decision branches, code flows, and unique paths of a business process adhere to documented specifications, with clearly defined inputs and expected outputs.

**7.3.2 Functional test**

Functional testing ensures that the system meets the functional requirements as specified in the business and technical documentation. It involves verifying that the software's functions operate as intended and meet user expectations. Functional testing focuses on:

* **Valid Input**: Ensuring identified classes of valid inputs are accepted.
* **Invalid Input**: Confirming identified classes of invalid inputs are rejected.
* **Functions**: Validating that all identified functions are exercised.
* **Output**: Verifying that all classes of application outputs are correct.
* **Systems/Procedures**: Testing interfacing systems or procedures to ensure smooth integration and functionality.

**7.3.3 System Test**

System testing is conducted to validate the complete and integrated system as per the defined requirements. It ensures that the application as a whole function correctly in a simulated real-world environment. This type of testing evaluates both functional and non-functional requirements, focusing on overall system behaviour, including performance, security, reliability, and usability.

Key objectives of system testing include:

* Verifying that the system meets specified requirements.
* Testing the interaction between different modules and their integration.
* Ensuring that the system performs consistently under varying conditions.
* Identifying defects that may arise from interactions between integrated components.

System testing is performed after unit and integration testing and serves as the final validation before the product is delivered to users.

**7.3.4 Performance Test**

Performance testing is conducted to evaluate the responsiveness, stability, scalability, and reliability of the system under a specific workload. The primary goal is to ensure that the application meets performance requirements and performs efficiently under expected and peak load conditions.

Key aspects of performance testing include:

* **Load Testing**: Determines the system's behaviour under normal and peak load conditions.
* **Stress Testing**: Examines the system's robustness by testing its limits beyond the normal operational capacity.
* **Scalability Testing**: Assesses the system's ability to scale up or down as per demand.
* **Response Time**: Measures the time taken to process requests and return results.

Performance testing helps identify bottlenecks, optimize resource usage, and ensure that the application performs seamlessly for end users. It is essential for ensuring a smooth user experience, especially in resource-intensive or high-traffic environments.

**7.3.5 Integration Testing**

Integration testing focuses on verifying the interactions between integrated components or systems to ensure they work together as expected. This process tests the data flow between modules and identifies issues in their interactions, such as communication errors, incorrect data handling, or functional mismatches.

**Integration testing is conducted after unit testing and before system testing, helping to identify and resolve issues in the interconnections of various components in the application.**

**7.3.6 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node.
* The Route add operation is done only when there is a Route request in need.
* The Status of Nodes information is done automatically in the Cache Updating process.

**7.2.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

***CHAPTER 8***

**FUTURE ENHANCEMENT**

**8.1. FUTURE ENHANCEMENTS:**

The system can be further enhanced by incorporating the following features and improvements:

* **Enhanced Predictive Accuracy**: Implement advanced machine learning algorithms, such as ensemble methods or neural networks, to further improve prediction accuracy.
* **Real-Time Data Integration**: Enable real-time updates by integrating live market data, ensuring pricing predictions reflect current trends.
* **Feature Expansion**: Incorporate additional parameters like regional market trends, insurance data, or seasonal variations to provide more comprehensive insights.
* **User Interface Improvements**: Develop an intuitive and user-friendly interface for broader accessibility and ease of use.
* **Mobile Compatibility**: Introduce mobile applications for on-the-go car price predictions and easy access for users.
* **Integration with E-commerce Platforms**: Connect with car resale platforms to facilitate seamless listing and pricing recommendations.
* **Global Adaptation**: Customize the system to cater to international markets by accommodating diverse data sets and pricing standards.

***CHAPTER 9***

**CONCLUSION AND REFERENCES**

**9.1 CONCLUSION**

This project demonstrates the potential of machine learning in enhancing car price prediction by leveraging data-driven insights. Traditional pricing systems often lack precision and adaptability, but the proposed system addresses these shortcomings using decision tree algorithms to deliver accurate and reliable predictions. The approach incorporates various car attributes, market trends, and data preprocessing techniques to create a model that performs with a high degree of accuracy.

The outcomes of this project can assist dealers, buyers, and sellers by providing transparent and data-backed car pricing, streamlining the decision-making process. Furthermore, the system's scalability ensures its applicability across different markets and scenarios, marking a significant improvement over conventional method.

With opportunities for future enhancements, such as real-time data integration and advanced algorithm adoption, this system lays a strong foundation for a modern, dynamic, and user-friendly car pricing solution.

* 1. **REFERENCES**
* Quinlan, J. R. (1986). *Induction of decision trees*. Machine Learning, 1(1), 81-106.
* Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques*. Elsevier.
* Scikit-learn Documentation. Retrieved from <https://scikit-learn.org>.
* Kelley Blue Book. *Car Value and Pricing Data*. Retrieved from <https://www.kbb.com>.
* Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.
* Kaggle. *Car Price Prediction Dataset*. Retrieved from <https://www.kaggle.com>.
* Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Education.