A MAJOR PROJECT REPORT

ON

"Predictor Application"

Submitted in partial fulfillment of

Master of Computer Application (MCA)

Session: 2023-2024

DEPARTMENT OF COMPUTER SCIENCE &

INFORMATICS

UNIVERSITY OF KOTA, KOTA

Near Kabir Circle, MBS Marg,
Kota (Rajasthan)-324005, INDIA,
RAJASTHAN – INDIA

Submitted To:

Submitted By:
PRATEEK KUMAWAT

SUPEPRVISOR:
Dr. O.P. RISHI & Dr. PC GUPTA

CERTIFICATE

This is certified that Minor Project entitled "**Predictor Application**" which is submitted by **PRATEEK KUMAWAT** of MCA 4th Semester, Master of Computer Application, University of Kota, Kota for the award of the Post-Graduation, is a Bonafide record of work carried out by them under guidance of **Dr. O.P. RISHI & Dr. PC GUPTA.**

The content of this Minor project has not been submitted to any university or institute for award of any degree or diploma.

PROJECT GUIDE: **External Examiner**
SUPEPRVISOR:
Dr. O.P. RISHI & Dr. PC GUPTA

_ _

HEAD OF DEPARTMENT:
DR. PRAKASH CHANDRA GUPTA

_ _ _ _ _ _ _ _ _ _ _ _ _

Date: _ _/ _ _/ _ _ _ _

Place: Kota

ABSTRACT

The "**Predictor**" project is a predictive modeling system designed to estimate car and house prices based on user-provided input. The system employs distinct methodologies for each prediction task: machine learning for car price prediction and deep learning for house price prediction.

For car price prediction, a robust machine learning model is developed to accurately determine the value of a vehicle based on factors such as make, model, year, mileage, and other relevant attributes. This model has demonstrated exceptional performance, achieving an accuracy rate of 99%.

In contrast, the house price prediction model leverages deep learning techniques to capture complex relationships within housing data. Factors like location, size, number of rooms, and property condition are considered to generate price estimates. While promising, the current house price prediction model exhibits an accuracy of 80%, indicating potential for further refinement.

The "**Predictor**" system aims to provide users with reliable price estimates for cars and houses, facilitating informed decision-making in respective markets. Future development will focus on expanding the system's predictive capabilities to encompass additional domains and enhancing model performance through advanced algorithms and techniques.
.

ACKNOLEDGEMENT

Sometimes it is hard to find the proper word to express my gratitude, and that is how I sense proper now. I am notably grateful for the huge help and support I received from my faculty and guide, **Dr. O.P. RISHI & Dr. PC GUPTA**. Their guidance and encouragement motivated me throughout the preparation of this project.

I would really like to expand my heartfelt thanks to **DR. REENA DADHICH (**Head of the Department of CSI), **DR. PRAKASH CHANDRA GUPTA**, **DR. O.P. RISHI, K.K. SHARMA** and all the faculty members who provided valuable insights and support. Their expertise and dedication in sharing knowledge enriched my learning experience.

Lastly, I want to express my big gratitude to my parents for her or his tireless support and blessings. Their constant encouragement has been a driving force throughout this project.

I am grateful for the contributions and help of these individuals, which performed a crucial position in the successful completion of this project.

PRATEEK KUMAWAT

TABLE OF CONTENT

TABLE OF FIGURES

Chapter 1: INTRODUCTION

1.1 Problem Statement:

The "Predictor" project addresses the growing demand for accurate predictive models in the automotive and real estate sectors. Predicting prices in these markets is crucial for consumers, businesses, and investors alike, facilitating informed decision-making and financial planning.

The Challenge of Accurate Price Prediction:

Accurate price prediction is a complex problem that has significant implications across various sectors of the economy. The ability to precisely estimate the value of assets, such as cars and houses, is crucial for informed decision-making, efficient market operations, and financial stability.

The Automotive Industry:

The automotive industry is characterized by rapid technological advancements, evolving consumer preferences, and fluctuating market conditions. These factors contribute to the dynamic nature of car prices, making it challenging to accurately predict their value. Traditional valuation methods often rely heavily on expert opinions and historical data, which can be subjective and may not adequately capture the impact of emerging trends. Consequently, there is a pressing need for robust and data-driven approaches to car price prediction.

The Real Estate Market:

The real estate market is another domain where accurate price prediction is essential. Factors influencing property values are multifaceted, encompassing location, size, condition, amenities, economic indicators, and market trends. The inherent complexity of these factors makes it difficult to establish reliable price estimates using conventional methods. Moreover, the real estate market is susceptible to fluctuations due to economic cycles, government policies, and demographic shifts, further compounding the challenge of accurate prediction.

Limitations of Existing Methods:

Traditional methods for price prediction, such as rule-based systems and statistical models, often exhibit limitations in terms of accuracy and adaptability. Rule-based systems are prone to human error and may not capture complex relationships between variables. Statistical models, while effective in certain scenarios, can struggle to handle nonlinear patterns and interactions present in real-world data.

Furthermore, the availability of large-scale, high-quality datasets is crucial for developing accurate predictive models. However, data scarcity or inconsistencies can hinder the development of robust models, particularly in emerging markets or for niche product segments.

Addressing these challenges requires the development of advanced predictive models that can effectively leverage diverse data sources, capture complex relationships, and adapt to changing market conditions.

The Need for Innovative Solutions:

To overcome the limitations of existing methods and improve the accuracy of price predictions, there is a growing demand for innovative solutions. These solutions should incorporate the following characteristics:

- Data-driven approach: Utilize large-scale datasets to extract valuable insights and patterns.
- Advanced modeling techniques: Employ sophisticated machine learning and deep learning algorithms to capture complex relationships.
- Real-time adaptability: Continuously update and refine models to reflect changing market dynamics.
- User-friendly interface: Provide intuitive tools for users to input data and obtain predictions.
- Explainable models: Offer insights into the factors influencing predictions to enhance trust and transparency.

1.2 Project Objectives:

The primary objective of the Predictor project is to develop a robust and accurate prediction system capable of estimating the prices of cars and houses. By harnessing the power of machine learning and deep learning techniques, the project aims to create a valuable tool for individuals and businesses involved in the automotive and real estate industries.

Specific Objectives:

- To Develop a car price prediction model: Construct a machine learning model that accurately predicts car prices based on various attributes such as make, model, year, mileage, features, and market conditions. The model should demonstrate high predictive accuracy and generalizability across different car segments.

- Develop a house price prediction model: Create a deep learning model that effectively predicts house prices considering factors like location, size, number of bedrooms, bathrooms, amenities, and market trends. The model should be capable of handling complex relationships within housing data and achieving robust performance.
- Implement a user-friendly interface: Design and develop an intuitive interface that allows users to input relevant information about the car or house and obtain accurate price predictions efficiently.
- Ensure model explainability: Develop interpretable models to provide insights into the factors influencing price predictions, enhancing user trust and understanding.
- Evaluate model performance: Rigorously assess the accuracy and reliability of both car and house price prediction models using appropriate evaluation metrics.
- Explore potential enhancements: Identify opportunities for improving model performance through feature engineering, hyperparameter tuning, and advanced modeling techniques.

Key Performance Indicators (KPIs):

To measure the success of the project, the following KPIs will be employed:

- Model accuracy: The percentage of correct predictions made by the models.
- Mean Squared Error (MSE): The average squared difference between predicted and actual prices.
- Mean Absolute Error (MAE): The average absolute difference between predicted and actual prices.
- User satisfaction: Feedback from users regarding the usability and accuracy of the system.
- Model explainability: The ability to understand the factors influencing price predictions.

By achieving these objectives and meeting the defined KPIs, the Predictor project will provide a valuable tool for individuals and businesses seeking accurate and reliable price estimates for cars and houses.

Expected Outcomes:

The successful completion of this project is expected to yield the following outcomes:

- Improved decision-making: Accurate price predictions will empower users to make informed decisions regarding car and house purchases or sales.

- Enhanced market efficiency: By providing reliable price estimates, the system can contribute to a more efficient and transparent market.
- Potential for commercialization: The project has the potential to be commercialized as a standalone product or integrated into existing platforms.
- Advancement of predictive modeling techniques: The development of accurate car and house price prediction models can contribute to the advancement of machine learning and deep learning research.

The project aims to develop a state-of-the-art prediction system that addresses the challenges of accurate price estimation in the automotive and real estate industries. By achieving its objectives and demonstrating strong performance, the Predictor project has the potential to make a significant impact on these sectors.

1.3 Project Scope:

The Predictor project aims to develop a functional prototype capable of accurately predicting car and house prices based on user-provided input data. The scope of this project encompasses the following key areas:

Core Functionalities:

- Data Acquisition and Preprocessing: The project will focus on acquiring relevant datasets for cars and houses, including features such as make, model, year, mileage, location, size, number of bedrooms, and bathrooms. Data cleaning, preprocessing, and feature engineering techniques will be applied to prepare the data for modeling.
- Model Development and Training: The project will develop and train machine learning and deep learning models for car and house price prediction. The models will be evaluated using appropriate metrics to assess their performance.
- User Interface Development: A basic user interface will be created to allow users to input data related to cars or houses and obtain price predictions.
- Model Deployment: The trained models will be integrated into the user interface to provide real-time predictions.

Project Boundaries:

To ensure a focused and deliverable project, the following limitations have been established:

- **Data Sources**: The project will utilize publicly available datasets or datasets provided by collaborators. Data collection and generation are outside the scope of this project.
- **Model Complexity**: The project will focus on developing baseline models for car and house price prediction. Extensive hyperparameter tuning and advanced model architectures will be explored in future iterations.
- **User Interface Features**: The initial focus will be on a basic user interface for data input and prediction output. Advanced features such as user authentication, data storage, and personalization will be considered for future development.
- **Model Deployment Platform**: The project will prioritize model deployment on a local environment. Integration with cloud platforms or mobile applications will be considered in subsequent phases.

Out-of-Scope Activities:

The following activities are explicitly excluded from the project scope:

- Comprehensive market analysis and trend forecasting.
- Development of mobile or web applications for the project.
- Integration with third-party APIs or services for data enrichment.
- Ethical considerations and bias mitigation in model development.

This document provides a framework for project planning, execution, and evaluation. This focused approach ensures that the project remains manageable and delivers a valuable prototype within the specified time and resource constraints.

1.4 Significance of the Project:

The Predictor project holds significant importance for individuals, businesses, and the overall economy. Accurate price prediction for cars and houses has far-reaching implications across various sectors.

To Benefits for Individuals:

- Informed decision-making: By providing reliable price estimates, the Predictor empowers individuals to make informed decisions when buying or selling cars and houses. Buyers can assess whether a listed price is fair and negotiate effectively, while sellers can optimize their pricing strategies.

- Time and resource savings: The system can save individuals valuable time and effort by eliminating the need for extensive market research and price comparisons.
- Financial planning: Accurate price predictions can aid in financial planning, budgeting, and investment decisions related to vehicle and property ownership.

Benefits for Businesses:

- Inventory management: Dealerships and real estate agencies can leverage the Predictor to optimize inventory pricing and turnover.
- Risk mitigation: Accurate price predictions can help businesses assess market risks and make informed investment decisions.
- Enhanced customer service: By providing reliable price estimates, businesses can improve customer satisfaction and loyalty.
- Market analysis: The system can be used for market analysis and trend identification, enabling businesses to identify opportunities and challenges.

Economic Impact:

The Predictor project has the potential to contribute to the overall economic landscape in several ways:

- **Market efficiency:** Accurate price predictions can lead to more efficient markets by reducing information asymmetry and price discrepancies.
- **Economic growth:** By facilitating informed decision-making and investment, the system can contribute to economic growth and development.
- **Innovation:** The development of advanced predictive models can drive innovation in the fields of machine learning and artificial intelligence.

Broader Implications:

Beyond the immediate benefits for individuals and businesses, the Predictor project has broader implications for society:

- Consumer protection: Accurate price predictions can help protect consumers from fraudulent pricing practices.
- Financial stability: Reliable valuation models contribute to financial stability by reducing the risk of asset bubbles and market crashes.
- Research and development: The project can serve as a foundation for further research in predictive modeling and its applications across various domains.

The Predictor project has the potential to significantly impact individuals, businesses, and the economy as a whole. By providing accurate and reliable price predictions, the system addresses a critical need and contributes to a more informed and efficient marketplace.

1.5 System Requirements Specification (SRS):

The System Requirements Specification (SRS) outlines the functional and non-functional requirements for the Predictor system. It serves as a comprehensive blue-print, defining the system's behavior, constraints, and performance expectations.

Overview

The SRS document provides a detailed description of the Predictor system, its intend-ed users, and the environment in which it operates. It establishes a common under-standing of the system's goals, functionalities, and constraints among stakeholders.

General Description

- **Product Perspective:** The Predictor system is a software application designed to provide accurate price predictions for cars and houses based on user-provided input data. It aims to assist individuals and businesses in making in-formed decisions related to buying or selling these assets.
- **Product Functions:** The system offers the following core functionalities:
    - User-friendly interface for data input and prediction output.
    - Data preprocessing and cleaning capabilities.
    - Machine learning and deep learning models for car and house price prediction.
    - Accurate and reliable price prediction generation.
- **User Characteristics:** The system is designed for users with varying levels of technical expertise, including individuals, real estate agents, and automotive dealers.
- **General Constraints:** The system should adhere to performance, security, and usability standards. It should be compatible with commonly used operat-ing systems and devices.

Specific Requirements

- **User Interface:**
    - Provide a clear and intuitive interface for data input.
    - Display prediction results in a user-friendly format.
    - Allow users to modify input data and re-run predictions.
- **Data Preprocessing:**
    - Handle missing data using appropriate imputation techniques.
    - Detect and handle outliers.
    - Perform data cleaning and normalization.

    - Convert categorical data into numerical format for model input.
- **Prediction Models:**
    - Implement accurate machine learning and deep learning models for car and house price prediction.
    - Ensure model interpretability and explain ability.
    - Provide options for model selection or customization.
- **Performance:**
    - Generate predictions within acceptable response times.
    - Handle concurrent user requests efficiently.
- **Usability:** The system should be easy to learn and use, with clear instructions and intuitive navigation.
- **Reliability:** The system should operate without failures or errors under normal conditions.
- **Security:** User data and predictions should be protected from unauthorized access.
- **Maintainability:** The system's codebase should be well-structured and documented for easy maintenance and updates.
- **Portability:** The system should be compatible with different operating systems and hardware configurations.

External Interface Requirements

- **User Interface:** The system should interact with users through a graphical user interface (GUI).
- **Data Sources:** The system may integrate with external data sources for additional information or model training.
- **Hardware:** The system should be compatible with standard computer hardware configurations.

Design Constraints

- **Budget:** The project has a defined budget for development, testing, and deployment.
- **Timeline:** The project has a specified timeline with milestones for different development phases.
- **Technology Constraints:** The system may be subject to limitations imposed by available hardware, software, and infrastructure.

This SRS document provides a foundational framework for the Predictor system, outlining its core functionalities, user requirements, and constraints. It serves as a reference point throughout the development process and ensures alignment with project objectives.

Chapter 2: LITERATURE REVIEW

Accurate price prediction is a critical area of research with applications across various domains, including the automotive and real estate industries. This chapter provides an overview of existing studies and research related to car and house price prediction, highlighting key methodologies, challenges, and advancements in the field.

Car Price Prediction:

The prediction of car prices has been a subject of extensive research due to its practical implications for consumers, dealers, and automotive industries. Early studies primarily relied on traditional statistical methods like linear regression and multiple regression analysis to establish relationships between car attributes and their prices. However, with the advent of machine learning, more sophisticated models have emerged.

- Machine Learning Techniques:
    - Support Vector Regression (SVR): Several studies have explored the use of SVR for car price prediction, demonstrating its effectiveness in handling complex relationships between features.
    - Random Forest: This ensemble method has gained popularity due to its ability to handle both numerical and categorical data, as well as its robustness to outliers.
    - Gradient Boosting: Algorithms like Gradient Boosting Machines (GBM) and XGBoost have shown promising results in car price prediction tasks, often outperforming traditional methods.
    - Neural Networks: Deep learning architectures, such as Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), have been applied to car price prediction, especially when dealing with image-based features or time-series data.

- Data Preprocessing and Feature Engineering:
    - Effective data preprocessing techniques, including handling missing values, outlier detection, normalization, and feature scaling, are crucial for building accurate models.
    - Feature engineering involves creating new features or transforming existing ones to improve model performance. Common techniques include one-hot encoding for categorical variables, interaction terms, and polynomial features.

House Price Prediction:

Predicting house prices has been a longstanding research area, with a focus on identifying factors influencing property values and developing accurate models.

- Traditional Methods:
  - Hedonic pricing models have been widely used to estimate house prices based on observable characteristics such as location, size, number of bedrooms, and amenities.
  - Spatial econometric models consider the spatial dependencies between properties, capturing the impact of neighborhood effects on prices.
- Machine Learning Techniques:
  - Similar to car price prediction, machine learning algorithms like linear regression, random forest, and support vector regression have been applied to house price prediction.
  - Deep learning models, including artificial neural networks and recurrent neural networks, have gained traction in recent years, especially when dealing with large and complex datasets.
- Data-Driven Approaches:
  - The availability of large-scale datasets and advanced data analytics techniques has facilitated the development of more sophisticated house price prediction models.
  - Incorporating spatial data, economic indicators, and other external factors has shown promising results in improving prediction accuracy.

Challenges and Future Directions:

Despite significant advancements in the field, several challenges remain in car and house price prediction:

- Data quality and availability: The quality and completeness of data can significantly impact model performance.
- Dynamic market conditions: Price fluctuations due to economic factors, consumer preferences, and technological advancements pose challenges for model adaptability.
- Feature engineering: Identifying relevant features and creating effective feature representations is crucial but often challenging.

- Model interpretability: Understanding the factors driving price predictions is essential for building trust and gaining insights.

Future research should focus on addressing these challenges and exploring innovative approaches:

- **Hybrid models:** Combining multiple models to leverage their strengths and mitigate weaknesses.
- **Explainable AI:** Developing interpretable models to enhance transparency and user trust.
- **Transfer learning:** Leveraging knowledge from related domains to improve model performance.
- **Real-time prediction:** Developing models capable of adapting to rapidly changing market conditions.

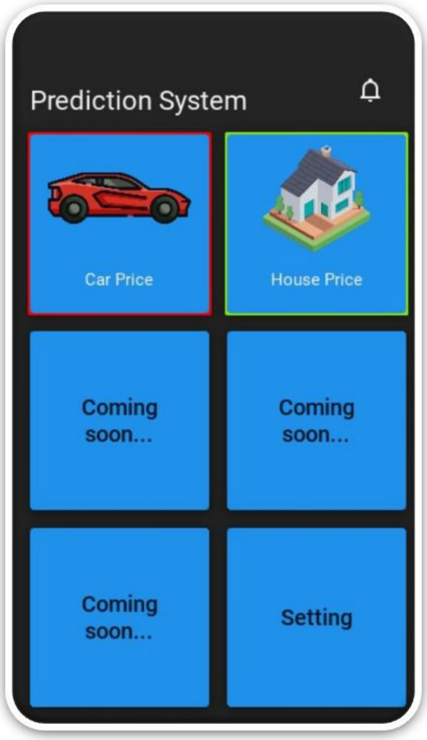
By building upon the existing body of research, this project aims to contribute to the advancement of price prediction methodologies and develop accurate models for both car and house prices.

Chapter 3: SYSTEM ARCHITECTURE

The Predictor system is designed as a streamlined platform for accurate price prediction. It comprises a user interface, data preprocessing module, prediction models, and a prediction engine.

Users interact with the system through the user interface, inputting details about the car or house. This data is then channeled to the preprocessing module for cleaning and preparation. The prediction engine selects the appropriate model (car or house price) based on the input and forwards the processed data.

The prediction model generates the price estimate, which is then displayed to the user via the interface. This architecture ensures efficient processing of user queries and delivers precise price predictions based on the underlying models.



1: Home Page

3.1 System Components:

The Predictor system comprises a user interface, data preprocessing module, prediction models, and a prediction engine. These components collaborate to process user input, clean and prepare data, select appropriate models, and generate accurate price predictions.

User Interface (UI):

The UI serves as the primary interface between the user and the system. It provides a user-friendly platform for inputting data related to the car or house being evaluated. Key elements of the UI include:

- **Input fields:** Collect essential information such as make, model, year, mileage, location, size, number of bedrooms, and bathrooms.
- **Data validation:** Ensures the accuracy and completeness of user input.
- **Prediction display:** Presents the calculated price prediction to the user in a clear and understandable format.
- **Navigation:** Facilitates user interaction and guides them through the prediction process.

Data Pre-processing Module:

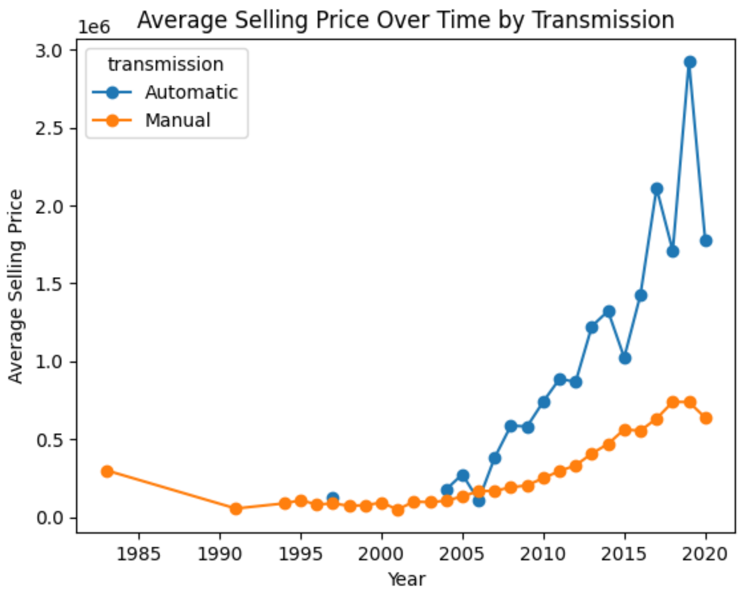
The data pre-processing module is responsible for transforming raw user input into a suitable format for the prediction models. Its primary functions include:

- **Data cleaning:** Handles missing values, outliers, and inconsistencies in the input data.
- **Feature engineering:** Creates new features or transforms existing ones to improve model performance. For example, deriving age from the year of manufacturing or calculating the car's power-to-weight ratio.
- **Data normalization:** Scales numerical features to a common range to prevent bias in model training.
- **Data formatting:** Converts data into appropriate formats required by the prediction models, such as numerical arrays or data frames.

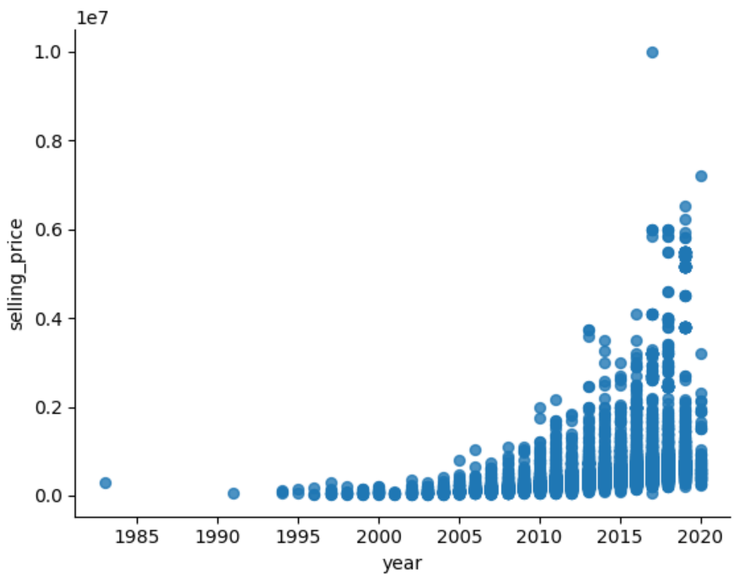
Prediction Models:

The Predictor system incorporates two primary prediction models:

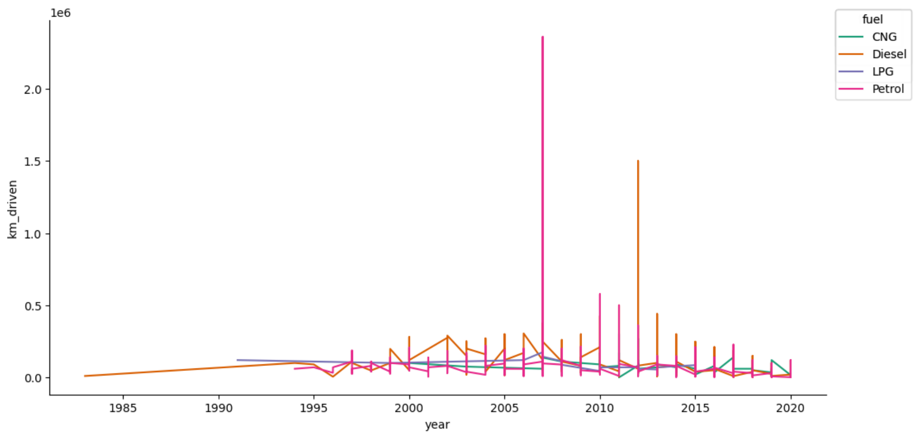
- **Car Price Prediction Model:** This model employs machine learning algorithms to estimate car prices based on features like make, model, year, mileage, condition, and other relevant attributes.



2: Graph between year & Price with Transmission

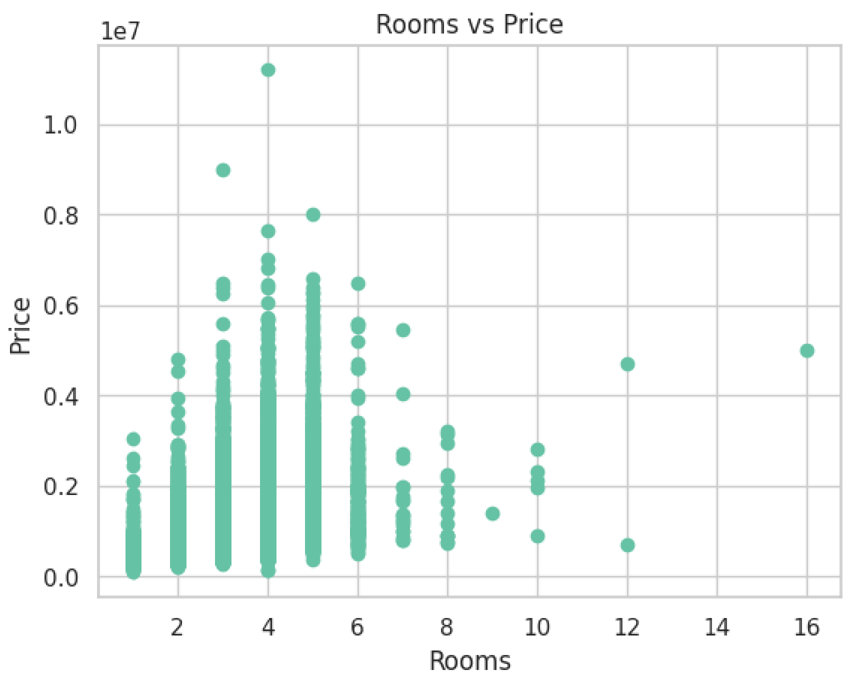


3: Graph between year & Price

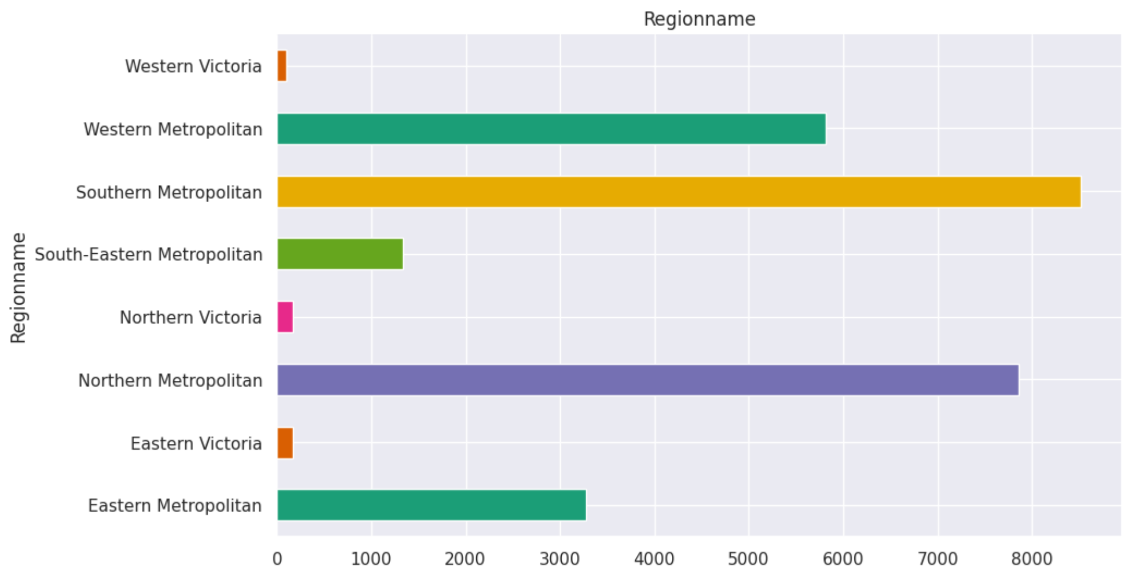


4: Graph of Car Type

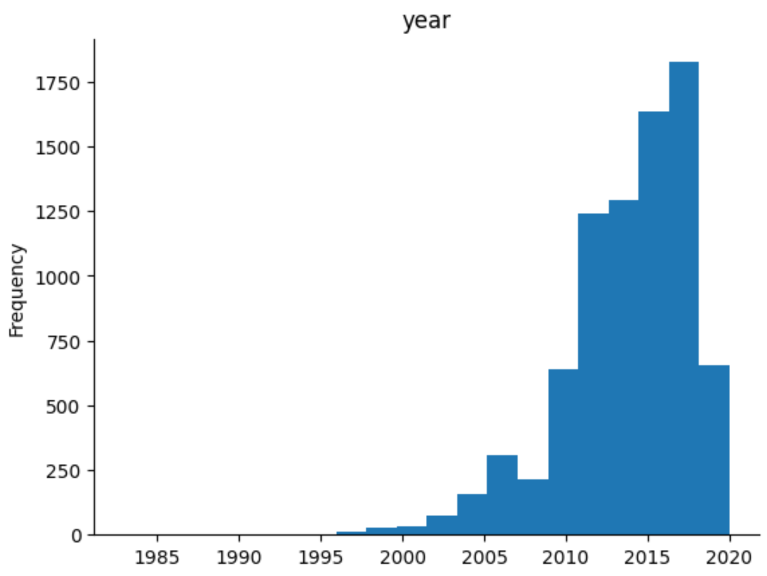
- **House Price Prediction Model:** A deep learning model is utilized to predict house prices considering factors such as location, size, number of bedrooms, bathrooms, amenities, and market conditions.



5: Graph of Rooms & Price



6: Graph of Region names



7: Graph of House frequency in year

Prediction Engine:

The prediction engine acts as the central component, coordinating the prediction process. Its key responsibilities include:

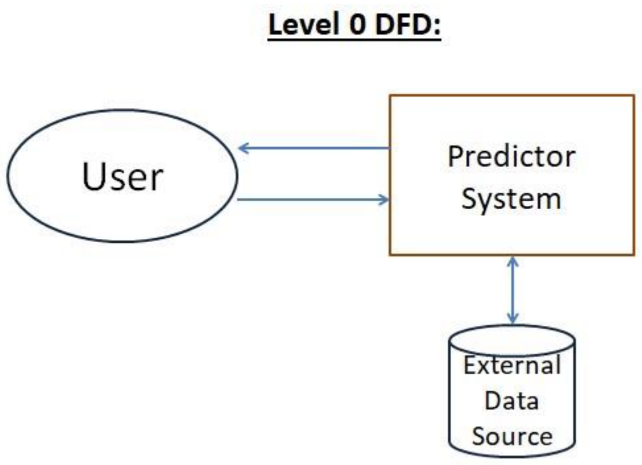
- **Model selection:** Determines the appropriate prediction model based on the type of asset being evaluated (car or house).
- **Data routing:** Directs the preprocessed data to the selected prediction model.
- **Result handling:** Receives the predicted price from the model and forwards it to the UI for display.
- **Error handling:** Manages potential errors or exceptions that may occur during the prediction process.

These components collectively contribute to the efficient functioning of the Predictor system, enabling accurate and reliable price predictions.

3.2 System Workflow:

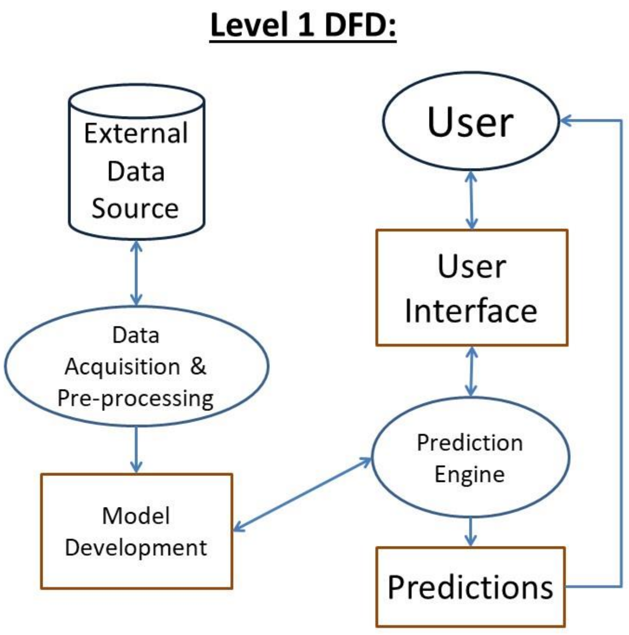
Data Flow Diagrams (DFDs) are graphical representations of the flow of data within a system. They provide a visual overview of the system's processes, data stores, and external entities. For the Predictor project, creating DFDs at different levels of abstraction can help understand the system's functionality and interactions.

- **Level 0 DFD (Context Diagram):** This highest-level DFD shows the system as a single bubble with external entities interacting with it.



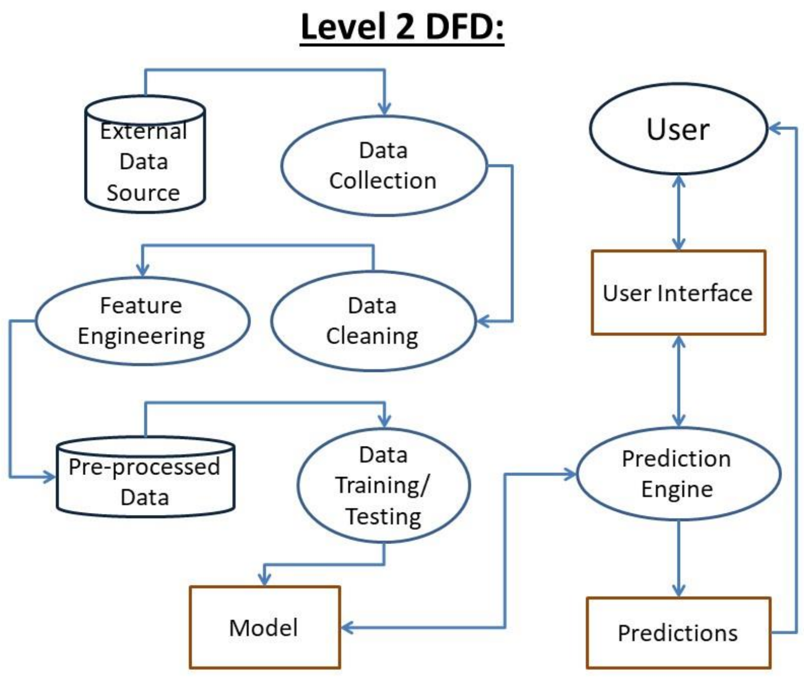
8: Level 0 DFD

- **Level 1 DFD:** Breaks down the system into major processes and data stores.



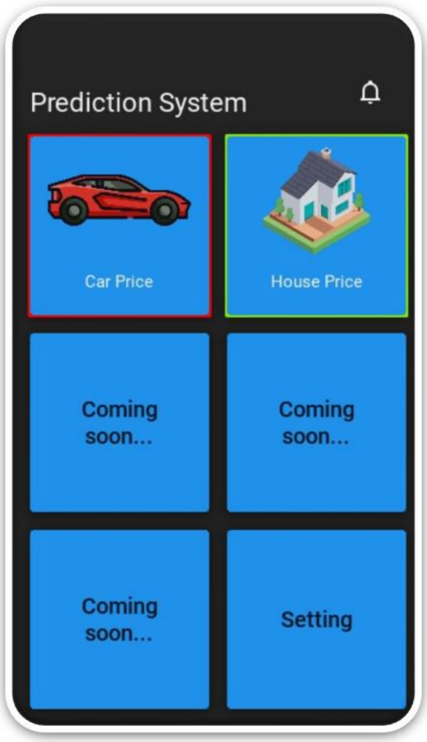
9: Level 1 DFD

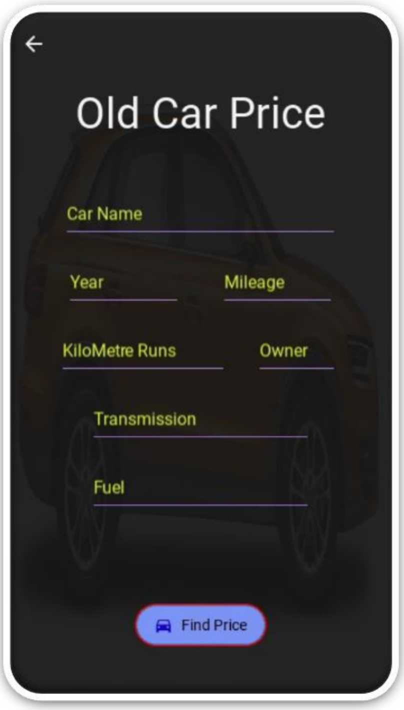
- **Level 2 DFD:** Further decomposes Level 1 processes into more detailed subprocesses and data stores.

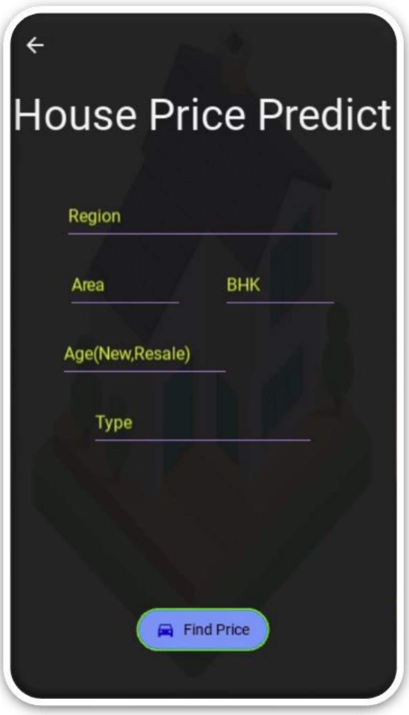


10: Level 2 DFD

The Predictor system operates in a streamlined manner. Users input car or house details through the interface. This data undergoes cleaning and preparation before being fed into the appropriate prediction model – car or house. The chosen model processes the data and generates a price estimate. This estimate is then presented to the user via the interface. The system includes error handling mechanisms to ensure smooth operation.







11: GUI Images of App

1. User Input:

- The user interacts with the system's user interface, providing relevant information about the car or house they wish to evaluate.
- Essential details such as make, model, year, mileage, location, size, number of bedrooms, and bathrooms are typically required for accurate predictions.
- The user interface ensures data consistency and completeness through input validation mechanisms.

2. Data Preprocessing:

- Once the user input is captured, it undergoes a preprocessing phase to prepare the data for model consumption.
- Data cleaning processes eliminate inconsistencies, errors, or missing values to maintain data integrity.
- Feature engineering techniques may be applied to create new features or transform existing ones, enhancing the model's predictive capabilities. For instance, deriving age from the manufacturing year or calculating the car's power-to-weight ratio.
- Data normalization or standardization might be performed to scale numerical features within a specific range, preventing bias in model training.

3. Model Selection:

- The system determines the appropriate prediction model based on the type of asset being evaluated.
- For car price prediction, the machine learning model is selected, while for house price prediction, the deep learning model is employed.
- This step ensures that the correct model is utilized for the specific prediction task.

4. Data Transfer:

- The preprocessed data is transferred to the selected prediction model for analysis.
- The model receives the data in a suitable format, such as numerical arrays or data frames, to facilitate computation.

5. Model Prediction:

- The prediction model processes the input data and generates a price estimate.
- The car price prediction model employs machine learning algorithms to analyze car attributes and derive a corresponding price.

- The house price prediction model utilizes deep learning techniques to capture complex relationships within housing data and produce an estimated price.

6. Output Generation:

- The predicted price is returned to the prediction engine.
- The prediction engine formats the result in a user-friendly manner.
- The price estimate is displayed to the user through the system's user interface.

7. Error Handling:

- The system incorporates error handling mechanisms to address potential issues that may arise during the prediction process.
- Errors related to data quality, model performance, or system failures are handled gracefully to prevent system disruptions.
- Informative error messages can be provided to the user to assist in troubleshooting.

This workflow outlines the core steps involved in the Predictor system's operation, from user input to the delivery of price predictions. By following this structured process, the system ensures efficient and accurate predictions.

Chapter 4: DATASET AND PREPROCESSING

4.1 Dataset Description:

The foundation of any predictive model is the quality and relevance of the underlying data. This section delves into the datasets employed for the Predictor project, specifically focusing on car and house price prediction. A comprehensive understanding of dataset characteristics, preprocessing steps, and feature selection is crucial for building robust and accurate models.

Car Price Dataset:

Dataset Source and Overview:

The car price dataset utilized in this project was sourced from Kaggle, a renowned platform for data science competitions and datasets. This dataset offers a rich repository of used car listings, encompassing a diverse range of vehicles, including various makes, models, years, and conditions.

Data Exploration and Preprocessing:

Upon initial exploration, the dataset exhibited a substantial number of instances and attributes, providing a solid foundation for model development. However, meticulous data preprocessing was essential to ensure data quality and consistency.

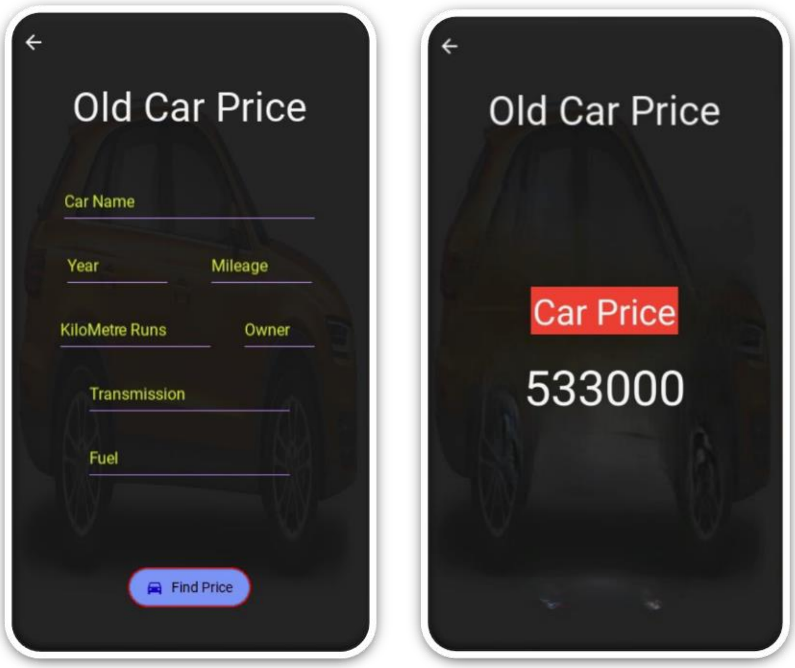
The dataset underwent several preprocessing steps:

- **Handling Missing Values:** Missing values were addressed through imputation techniques, such as mean, median, or mode imputation, depending on the nature of the variable. For categorical variables, the most frequent category was used.
- **Outlier Detection and Treatment:** Outliers, which can significantly impact model performance, were identified using statistical methods like z-scores or box plots. Depending on the nature and impact of outliers, they were either corrected, capped, or removed from the dataset.
- **Categorical Variable Encoding:** Categorical variables, such as car make, model, and fuel type, were converted into numerical representations using techniques like one-hot encoding or label encoding.
- **Feature Scaling:** Numerical features, such as mileage and engine power, were scaled to a common range to prevent bias in model training. Standardization or normalization techniques were applied based on the specific requirements of the prediction algorithm.

Feature Selection:

After rigorous data preprocessing, a subset of relevant features was selected for model development. The following attributes were deemed essential for car price prediction:

- **name:** The make and model of the car, serving as a categorical identifier.
- **year:** The manufacturing year, a numerical feature reflecting the car's age.
- **selling_price:** The target variable representing the car's selling price.
- **km_driven:** The total kilometers driven, a numerical feature indicating the car's usage.
- **fuel:** The type of fuel used by the car, a categorical feature.
- **transmission:** The type of transmission (manual or automatic), a categorical feature.
- **owner:** The number of previous owners, a categorical feature.
- **mileage(km/ltr/kg):** The fuel efficiency of the car, a numerical feature.



12: GUI of Input & Output of Car price

These features were selected based on their potential impact on car prices and their availability within the dataset.

House Price Dataset:

Dataset Source and Overview:

The house price dataset was also acquired from Kaggle, providing a comprehensive collection of residential property listings. This dataset encompasses various locations, property types, sizes, and amenities, offering a rich source of information for house price prediction.

Data Exploration and Preprocessing:

Similar to the car price dataset, the house price dataset underwent rigorous preprocessing to ensure data quality and consistency. The following steps were undertaken:

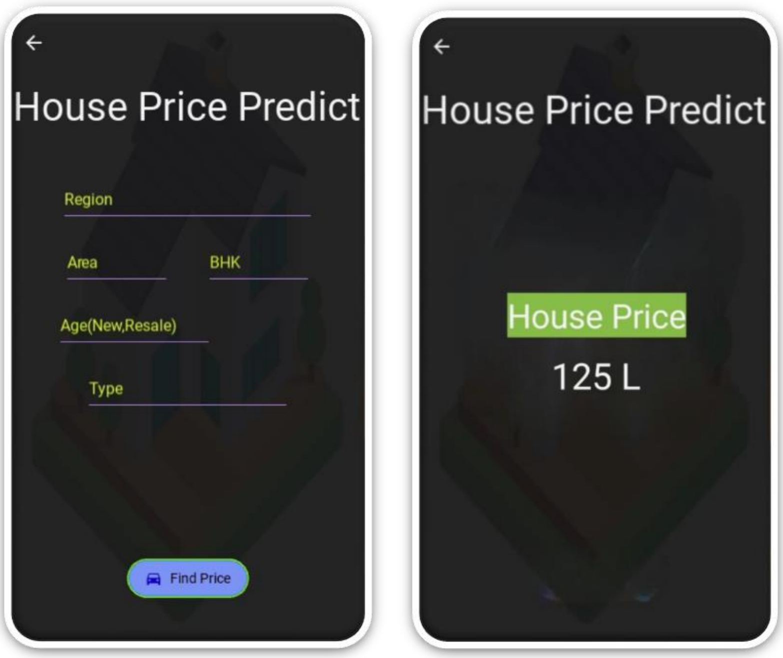
- **Handling Missing Values:** Missing values were treated using appropriate imputation techniques, considering the nature of the variable and the potential impact on model performance.
- **Outlier Detection and Treatment:** Outliers, such as exceptionally large or small property sizes or prices, were identified and handled through appropriate methods like capping, flooring, or removal.
- **Categorical Variable Encoding:** Categorical features like location, property type, and amenities were converted into numerical representations using suitable encoding techniques.
- **Feature Engineering:** New features were created or existing ones transformed to capture additional information relevant to house prices. For example, deriving the age of the property from the construction year or creating interaction terms between location and property type.
- **Feature Scaling:** Numerical features, such as property area and number of bedrooms, were scaled to ensure fair comparison and prevent bias in model training.

Feature Selection:

The following features were deemed crucial for house price prediction based on their relevance and availability in the dataset:

- **bhk:** The number of bedrooms, halls, and kitchens, a numerical feature.
- **type:** The property type (e.g., apartment, villa, independent house), a categorical feature.
- **area:** The size of the property in square feet, a numerical feature.
- **price:** The target variable representing the house's selling price.
- **region:** The location of the property, often encoded as a categorical feature.

- **age:** The age of the property, a numerical feature derived from the construction year.



13: GUI of Input & Output of House Price

These features collectively provide a strong foundation for building accurate house price prediction models.

The careful selection and preprocessing of datasets are fundamental to the success of any predictive modeling project. By addressing data quality issues, handling missing values, and creating relevant features, the car and house price datasets were prepared for building robust prediction models. The chosen features are expected to capture the essential factors influencing car and house prices, enabling the development of accurate and reliable predictive systems.

4.2 Data Preprocessing:

Data preprocessing is a critical step in the machine learning pipeline, ensuring that raw data is transformed into a suitable format for model training and evaluation. This section delves into the data preprocessing techniques applied to both the car price and house price datasets.

Car Price Dataset Preprocessing:

The car price dataset underwent a series of preprocessing steps to enhance its quality and suitability for model development:

- **Data Cleaning:**
    - **Missing Value Imputation:** Missing values in numerical columns were imputed with mean, median, or mode, depending on the data distribution. Categorical missing values were replaced with the most frequent category or a new category indicating missingness.
    - **Outlier Detection and Handling:** Outliers, identified using techniques like z-score or box plots, were carefully examined. Depending on their impact and potential causes, they were either corrected, capped, or removed from the dataset.
    - **Inconsistency Handling:** Errors in data entry, such as incorrect values or typos, were rectified through data cleaning processes.

- **Feature Engineering:**
    - **Categorical Encoding:** Categorical variables like car make, model, and fuel type were converted into numerical representations using techniques like one-hot encoding or label encoding.
    - **Feature Creation:** New features were derived from existing ones to capture additional information. For instance, the age of the car could be calculated from the year column, or interaction terms between features could be created.
    - **Feature Scaling:** Numerical features were scaled using techniques like standardization or normalization to ensure they have a comparable range, preventing bias in model training.

House Price Dataset Preprocessing:

The house price dataset required similar preprocessing steps, with some additional considerations:

- **Data Cleaning:**
    - **Missing Value Imputation:** Similar to the car price dataset, missing values were handled using appropriate imputation techniques. However, due to the nature of real estate data, imputation methods like median or mode imputation might be more suitable for certain features.
    - **Outlier Detection and Handling:** Outliers in house prices or property sizes were carefully analyzed and treated accordingly. Domain knowledge and

statistical methods were combined to make informed decisions about outlier handling.

  - **Data Consistency:** Inconsistencies in property types, area units, or location data were rectified to ensure data accuracy.

- **Feature Engineering:**
  - **Categorical Encoding:** Categorical variables like location, property type, and amenities were encoded using suitable techniques.
  - **Feature Creation:** New features were derived to capture relevant information. For example, property age, number of rooms, or location-based features could be created.
  - **Textual Data Handling:** If the dataset included textual descriptions, techniques like text preprocessing, tokenization, and embedding could be applied to extract meaningful information.

- **Spatial Data Handling:** If the dataset included geographic coordinates, techniques like geospatial analysis could be employed to extract relevant features such as proximity to amenities, schools, or transportation hubs.

Common Preprocessing Challenges:

Several common challenges arise during data preprocessing:

- **Missing data:** Handling missing values appropriately requires domain knowledge and careful consideration of the imputation method.
- **Outlier detection:** Identifying and handling outliers can be subjective and requires domain expertise to make informed decisions.
- **Categorical data encoding:** Choosing the appropriate encoding technique for categorical variables depends on the number of categories and the model's requirements.
- **Feature engineering:** Creating informative features often requires domain knowledge and experimentation.

Addressing these challenges effectively is crucial for building robust and accurate predictive models.

Data preprocessing is a fundamental step in the machine learning pipeline, significantly influencing model performance. By carefully handling missing values, outliers, and inconsistencies, and by creating informative features, the quality of the data is enhanced. The preprocessing techniques applied to the car and house price datasets have laid the foundation for developing accurate predictive models.

Chapter 5: MODEL DEVELOPMENT

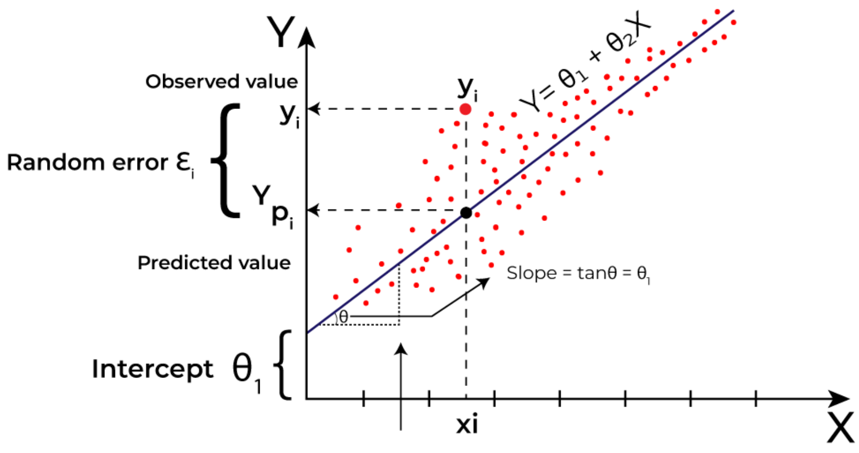
5.1 Car Price Prediction Model:

Understanding the Problem:

Accurately predicting car prices is a complex task influenced by numerous factors such as make, model, year, mileage, condition, features, and market trends. To address this challenge, a machine learning model capable of capturing these relationships is essential.

Model Selection: Linear Regression:

For this project, linear regression was chosen as the baseline model for car price prediction. Linear regression is a statistical method used to establish a linear relationship between a dependent variable (car price) and one or more independent variables (car features). Its simplicity, interpretability, and computational efficiency make it a suitable starting point for exploratory analysis.



Y
Observed value
y_i
Random error ε_i
Y_{p_i}
Predicted value
Intercept θ_1
y_i
$Y = \theta_1 + \theta_2 X$
Slope = tanθ = θ_1
θ
xi
X

14: Linear Regression in ML

Model Architecture"

Linear regression models the relationship between the dependent variable (car price) and independent variables (car features) as follows:

```
Price = β0 + β1 * Feature1 + β2 * Feature2 + … + βn * Featuren + ε
```

Where:

- `Price` is the predicted car price.
- $\beta 0$ is the intercept term.
- $\beta 1$, $\beta 2$, …, βn are the coefficients representing the impact of each feature on the price.
- `Feature1`, `Feature2`, …, `Featuren` are the independent variables (car features).
- ε is the error term, representing the unexplained variation in the price.

Hyper parameter Tuning:

While linear regression has relatively few hyper parameters compared to other machine learning algorithms, it's essential to consider the following:

- **Feature Scaling:** Standardizing or normalizing numerical features can improve model performance and convergence.
- **Regularization:** Techniques like Ridge or Lasso regression can help prevent overfitting, especially when dealing with a large number of features.
- **Model Selection:** Exploring different feature combinations or using feature selection methods can enhance model accuracy and interpretability.

Model Training and Evaluation:

The model is trained on a historical dataset containing car prices and their corresponding features. During training, the model learns the optimal values for the coefficients (β) by minimizing the sum of squared residuals.

To evaluate the model's performance, various metrics can be employed:

- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual prices.
- **Root Mean Squared Error (RMSE):** The square root of MSE, providing a more interpretable error metric in the same units as the target variable.
- **Mean Absolute Error (MAE):** Calculates the average absolute difference between predicted and actual prices.
- **R-squared:** Represents the proportion of variance in the dependent variable explained by the independent variables.

Limitations of Linear Regression:

While linear regression serves as a baseline model, it has limitations:

- **Linearity Assumption:** Assumes a linear relationship between features and the target variable, which might not hold true for complex datasets.
- **Sensitivity to Outliers:** Outliers can significantly impact the model's performance.
- **Inability to Capture Complex Relationships:** Linear regression might struggle to capture non-linear patterns or interactions between features.

To address these limitations, more sophisticated models such as polynomial regression, decision trees, or ensemble methods can be explored in future iterations.

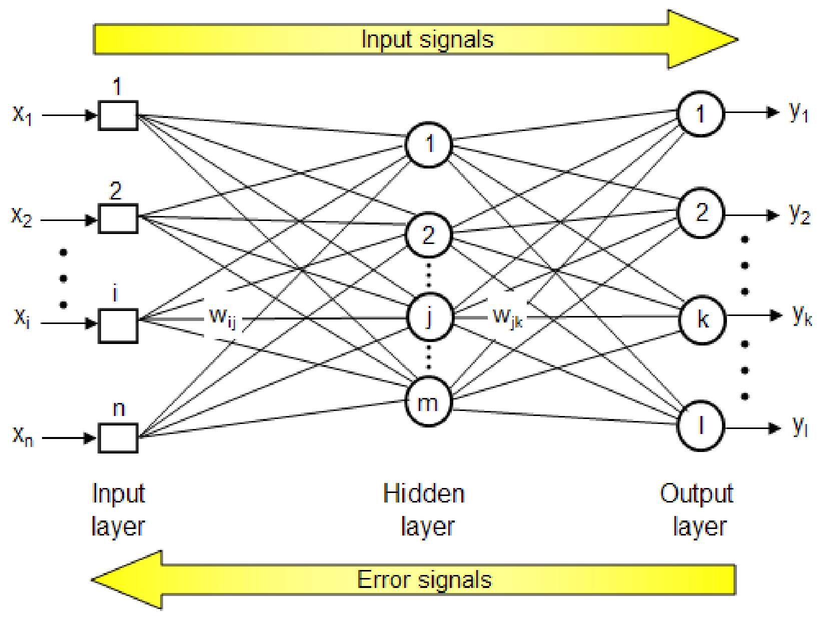
5.2 House Price Prediction Model:

Understanding the Problem:

Predicting house prices is a complex task influenced by numerous factors such as location, size, number of bedrooms, bathrooms, amenities, and overall market conditions. To capture these intricate relationships and build an accurate predictive model, deep learning techniques are often employed.

Deep Learning Model Architecture:

For house price prediction, a deep neural network architecture is commonly used. This section delves into the core components of a typical deep learning model for this task.



15: Deep Learning Layers

- **Input Layer:** The input layer receives the numerical and categorical features representing the house. These features might include square footage, number of bedrooms, bathrooms, location, and other relevant attributes.
- **Hidden Layers:** Multiple hidden layers are stacked to extract complex patterns and relationships within the data. Each layer consists of neurons that apply activation functions to the weighted sum of inputs from the previous layer. Common activation functions include ReLU (Rectified Linear Unit), sigmoid, or tanh.
- **Output Layer:** The final layer produces a single output, representing the predicted house price. Typically, a linear activation function is used for regression tasks.

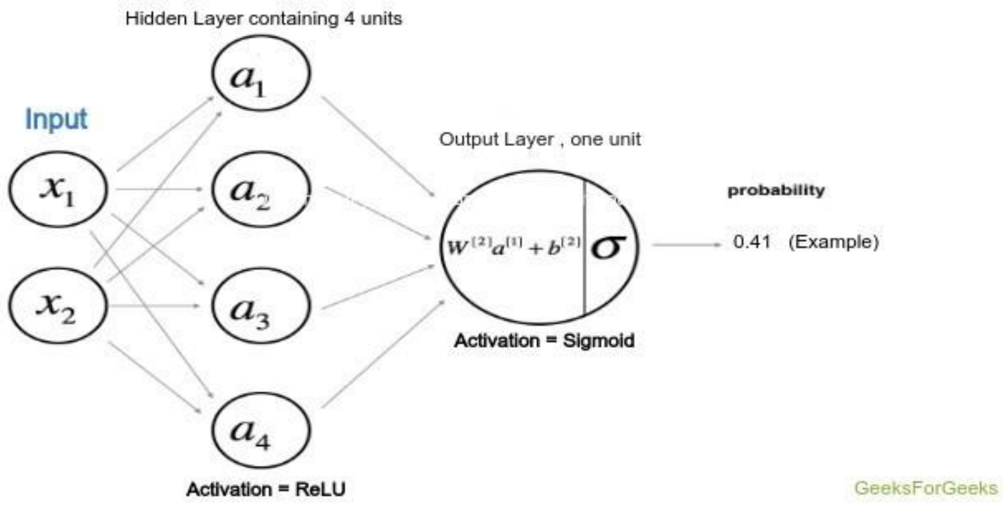
Hyper parameter Tuning:

The performance of a deep learning model is highly dependent on its hyperparameters. Key hyper parameters to consider include:

- **Number of hidden layers:** Determining the optimal number of hidden layers requires experimentation and validation.
- **Number of neurons per layer:** The number of neurons in each hidden layer influences the model's capacity to learn complex patterns.
- **Activation functions:** Different activation functions can impact the model's performance. Experimentation with various activation functions is necessary.
- **Learning rate:** The learning rate controls the step size during gradient descent optimization. Finding the optimal learning rate is crucial for convergence.
- **Batch size:** The number of training examples processed before updating model parameters.
- **Epochs:** The number of times the entire dataset is passed through the model during training.
- **Optimizer:** The optimization algorithm used to update model parameters, such as Adam, SGD, or RMSprop.

Model Training and Evaluation:

The deep learning model is trained on a labeled dataset containing house prices and their corresponding features. The training process involves iteratively adjusting model parameters to minimize the prediction error.



16: Process of Deep Learning

Common evaluation metrics for regression tasks include:

- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual prices.
- **Root Mean Squared Error (RMSE):** The square root of MSE, providing a more interpretable error metric in the same units as the target variable.
- **Mean Absolute Error (MAE):** Calculates the average absolute difference between predicted and actual prices.
- **R-squared:** Represents the proportion of variance in the dependent variable explained by the model.

Model Example:

Python

```
import tensorflow as tf
from tensorflow import keras
# Define the model architecture
model = keras.Sequential([
    keras.layers.Dense(64,     activation='relu',     in-
put_shape=(input_dim,)),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1)
])
# Compile the model
model.compile(optimizer='adam',
loss='mean_squared_error', metrics=['mae'])

# Fit the model
model.fit(X_train,  y_train,  epochs=100,  batch_size=32,
validation_split=0.2)
```

This code snippet demonstrates a basic deep neural network architecture for house price prediction using Keras. The model consists of two hidden layers with ReLU activation and an output layer with a linear activation. The model is compiled with the Adam optimizer and trained for 100 epochs with a batch size of 32.

Deep learning models offer powerful tools for capturing complex relationships in house price data. By carefully designing the model architecture, tuning hyperparameters, and evaluating performance using appropriate metrics, it is possible to build highly accurate house price prediction models. However, it's essential to consider factors like dataset size, feature engineering, and computational resources when developing and training deep learning models.

Chapter 6: SYSTEM IMPLEMENTATION

The Predictor system was brought to life through a strategic combination of programming languages, libraries, and development tools. This section delves into the technological stack that underpins the project.

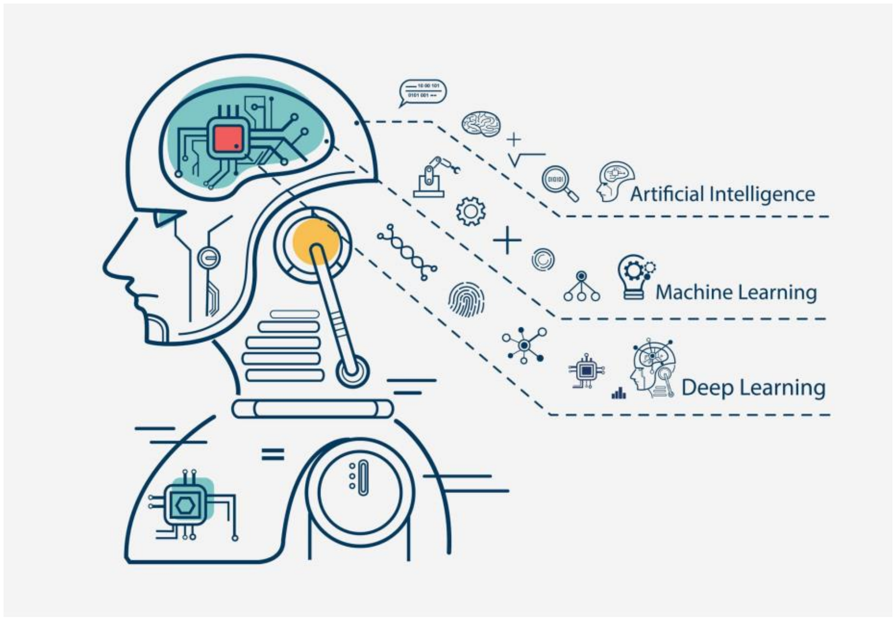
Programming Languages:

Python: As the primary language, Python's versatility, readability, and extensive ecosystem of libraries made it the ideal choice for data manipulation, model development, and system integration. Its rich set of libraries, including NumPy, Pandas, Matplotlib, and Seaborn, facilitated efficient data handling and analysis.

Python boasts a rich ecosystem of libraries tailored for machine learning and deep learning. NumPy forms the bedrock, providing efficient numerical operations. Pandas builds upon this for data manipulation and analysis. Scikit-learn offers a vast array of classical machine learning algorithms. For deep learning, TensorFlow and Keras shine, with TensorFlow providing the core and Keras offering a user-friendly interface. This robust toolkit empowers developers to tackle complex problems and build sophisticated models.

Machine Learning and Deep Learning Libraries:

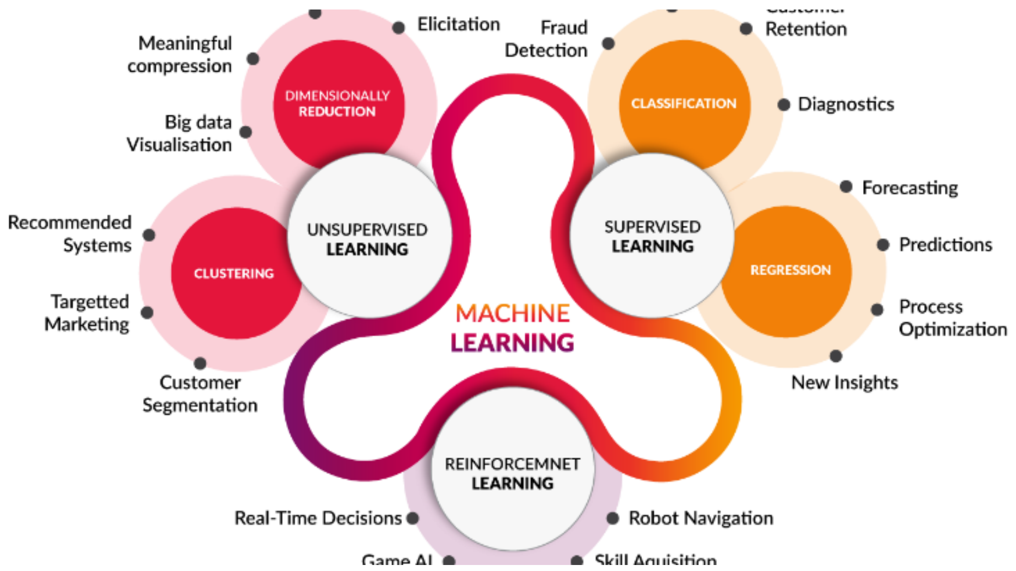
The realm of machine learning and deep learning is underpinned by a robust ecosystem of libraries that streamline development, accelerate experimentation, and enhance model performance. This section delves into the key libraries employed in the Predictor project, highlighting their roles and contributions.



17: ML & Deep Learning

Core Machine Learning Libraries:

- **NumPy:** As the fundamental library for numerical computations in Python, NumPy provides efficient array and matrix operations, essential for handling numerical data in machine learning.
- **Pandas:** Built on top of NumPy, Pandas offers high-performance data structures and analysis tools, making it indispensable for data manipulation, cleaning, and exploration.
- **Scikit-learn:** A comprehensive machine learning library, Scikit-learn provides a vast array of algorithms, including classification, regression, clustering, and dimensionality reduction techniques. Its user-friendly interface and well-documented API make it a popular choice for machine learning practitioners.
- **Matplotlib and Seaborn:** While primarily visualization libraries, Matplotlib and Seaborn play a crucial role in data exploration and model evaluation by providing tools to create informative plots and charts.

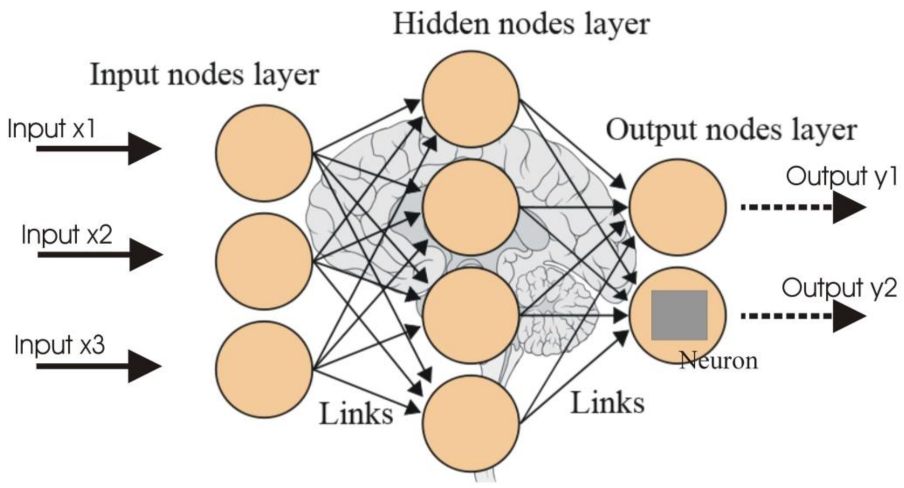


18: Type of ML

Deep Learning Libraries:

- **TensorFlow and Keras:** As the cornerstone of deep learning in the Predictor project, TensorFlow offers a flexible platform for building and training complex neural networks. Keras, its high-level API, simplifies model development and experimentation. This combination provides a powerful and efficient toolkit for deep learning tasks.

- **PyTorch:** While not explicitly used in this project, PyTorch is another popular deep learning framework known for its dynamic computation graph and ease of use. It offers an alternative to TensorFlow/Keras for deep learning development.



19: Architecture of Deep Learning

The Role of Libraries in the Predictor Project:

These libraries collectively contributed to the development of the Predictor system by:

- **Efficient data handling and manipulation:** NumPy and Pandas facilitated data cleaning, preprocessing, and feature engineering.
- **Model development and training:** Scikit-learn and TensorFlow/Keras provided the necessary tools for building and training machine learning and deep learning models.
- **Model evaluation and visualization:** Matplotlib and Seaborn enabled visualization of data and model performance metrics.
- **Accelerated development:** These libraries offer pre-implemented algorithms and functions, saving development time and effort.

User Interface Development:

- **Kivy:** This cross-platform Python framework enabled the creation of a visually appealing and interactive user interface. Kivy's ability to run on various operating systems (Windows, macOS, Linux, Android, iOS) facilitated wider accessibility to the Predictor system.

Development Environment:

- **Jupyter Notebook:** This interactive environment provided a conducive work-space for data exploration, experimentation, and code development.
- **Integrated Development Environments (IDEs):** Python-based IDEs like PyCharm or Visual Studio Code offered advanced features for code editing, debugging, and project management.

Additional Libraries and Tools:

- **NumPy:** For numerical computations and array manipulation.
- **Pandas:** For data manipulation and analysis.
- **Matplotlib and Seaborn:** For data visualization and exploratory data analysis.
- **Scikit-learn:** For machine learning algorithms and model evaluation.
- **TensorFlow and Keras:** For deep learning model development and training.
- **Kivy:** For user interface development.

the Predictor system was successfully implemented, demonstrating the effectiveness of Python and its associated libraries for data science and machine learning projects.

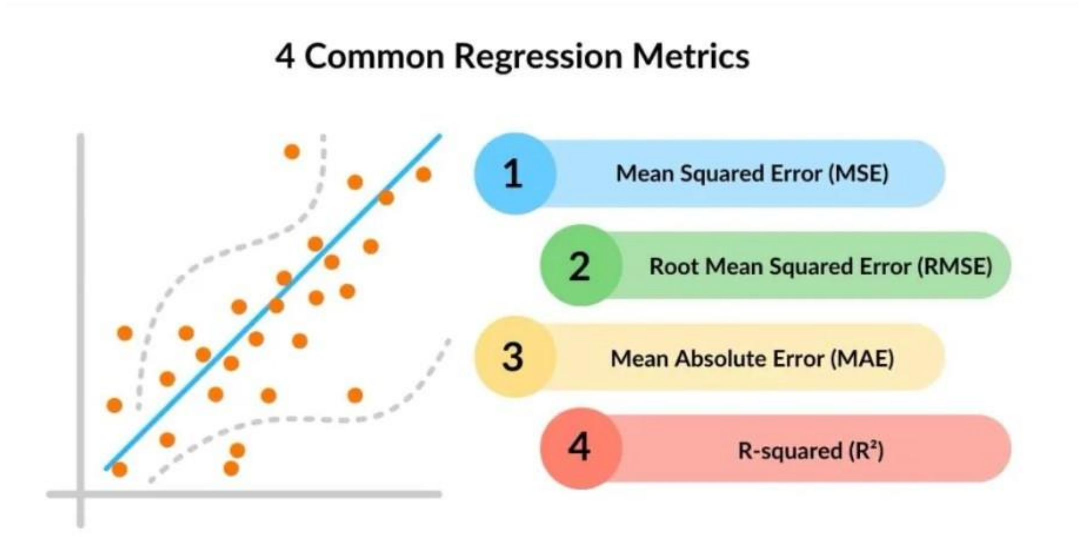
Chapter 7: RESULTS AND EVALUATION

7.1 Car Price Prediction Results:

Evaluating the performance of the car price prediction model is crucial to assess its effectiveness and reliability. This section delves into the model's performance metrics, providing insights into its accuracy, predictive power, and areas for improvement.

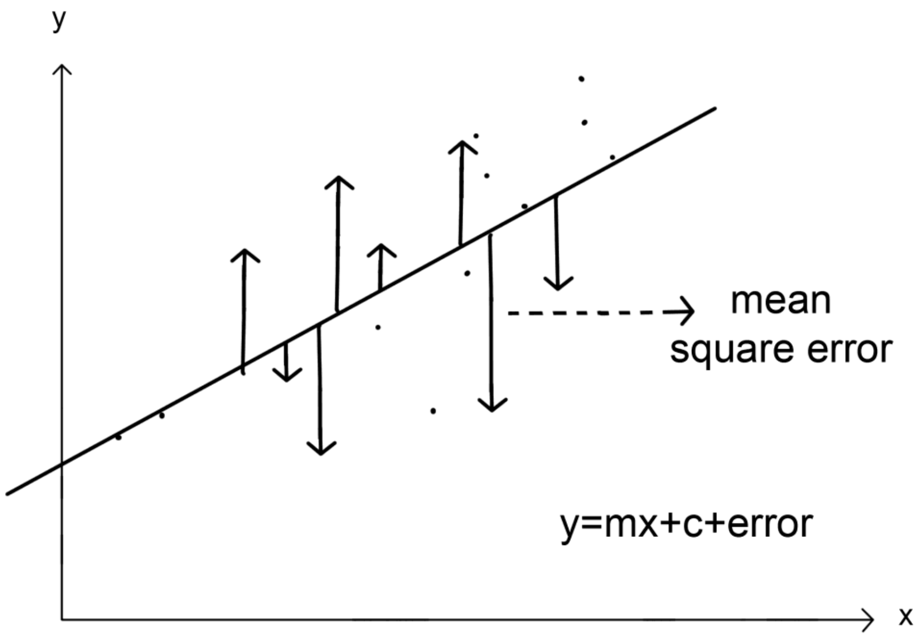
Model Evaluation Metrics:

To assess the car price prediction model's performance, several key metrics were employed:


4 Common Regression Metrics
1 Mean Squared Error (MSE)
2 Root Mean Squared Error (RMSE)
3 Mean Absolute Error (MAE)
4 R-squared (R²)

20: Common Regression Metrics

- **Mean Squared Error (MSE):** This metric quantifies the average squared difference between the predicted and actual car prices. A lower MSE indicates better model performance.



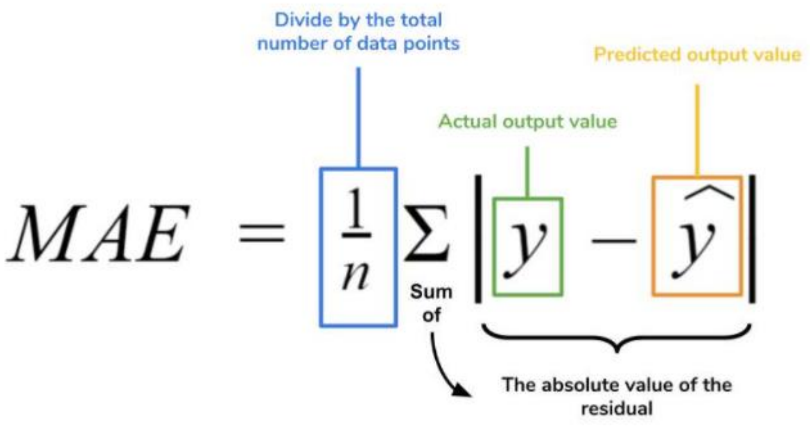
21: Error Graph in ML

$$\text{MSE} = \overset{\text{Mean}}{\boxed{\frac{1}{n}\sum_{i=1}^{n}}}(\overset{\text{Error}}{\boxed{Y_i - \hat{Y}_i}})^{\overset{\text{Squared}}{\boxed{2}}}$$

- **Root Mean Squared Error (RMSE):** The square root of MSE provides an error metric in the same units as the target variable (price), facilitating interpretability.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

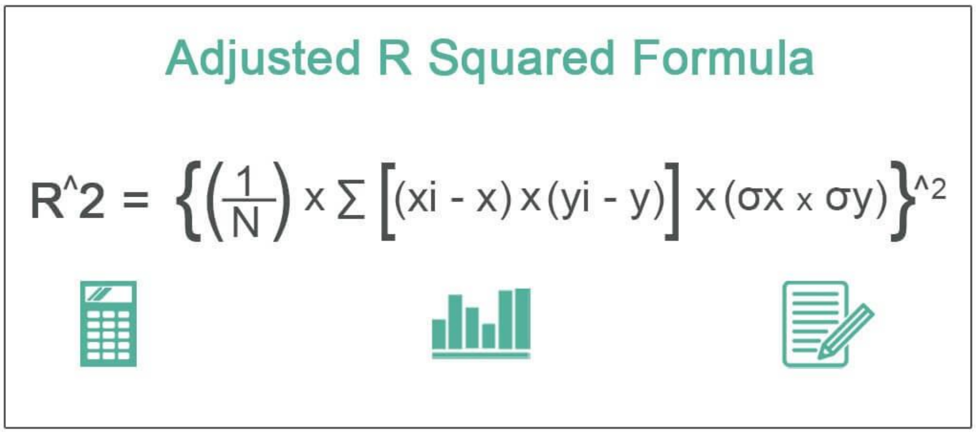
- **Mean Absolute Error (MAE):** Measures the average absolute difference between predicted and actual prices, offering a more robust metric compared to MSE in the presence of outliers.


Divide by the total number of data points
Predicted output value
Actual output value
MAE = 1/n Σ |y − ŷ|
Sum of
The absolute value of the residual

- **R-squared (R²):** Indicates the proportion of variance in the car price explained by the model, ranging from 0 to 1, with higher values signifying better fit.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

- **Adjusted R-squared:** A modified version of R-squared that penalizes the inclusion of unnecessary features, providing a more accurate estimate of model performance.



Adjusted R Squared Formula

$$R^2 = \left\{\left(\frac{1}{N}\right) \times \sum\left[(xi - x) \times (yi - y)\right] \times (\sigma x \times \sigma y)\right\}^2$$

Model Performance Analysis:

The obtained evaluation metrics provide valuable insights into the model's predictive capabilities. A high R-squared value suggests that the model explains a significant portion of the car price variance. Low values of MSE, RMSE, and MAE indicate accurate price predictions.

Error Analysis:

To further understand the model's strengths and weaknesses, error analysis is conducted. This involves examining the distribution of prediction errors, identifying potential outliers or systematic biases, and analyzing residuals.

- **Residual Plots:** Visualizing the residuals (differences between predicted and actual prices) against predicted values or independent variables can reveal patterns or trends indicative of model limitations.
- **Feature Importance:** Analyzing the contribution of different features to the model's predictions helps identify key factors influencing car prices and potential areas for improvement.

Model Comparison:

If multiple models were evaluated, a comparative analysis can be performed to identify the best-performing model based on the chosen metrics. Factors such as model complexity, interpretability, and computational efficiency should also be considered.

Limitations and Future Improvements:

While the model demonstrates promising performance, it's essential to acknowledge its limitations and potential areas for improvement:

- **Data Quality:** The quality and completeness of the dataset significantly impact model performance. Addressing data inconsistencies, missing values, and outliers is crucial.
- **Feature Engineering:** Exploring additional features or transformations can enhance the model's predictive power.
- **Model Complexity:** The chosen model might be overly simplistic or complex for the given dataset. Experimenting with different model architectures or hyperparameters can lead to performance gains.
- **External Factors:** The model might not capture the impact of external factors, such as economic conditions or market trends, which can influence car prices.

By addressing these limitations and incorporating advanced techniques, the car price prediction model can be further refined and improved.

7.2 House Price Prediction Results:

Evaluating the performance of the house price prediction model is essential to understand its accuracy, reliability, and potential areas for improvement. This section delves into the key metrics used to assess the model's performance and provides insights into its predictive capabilities.

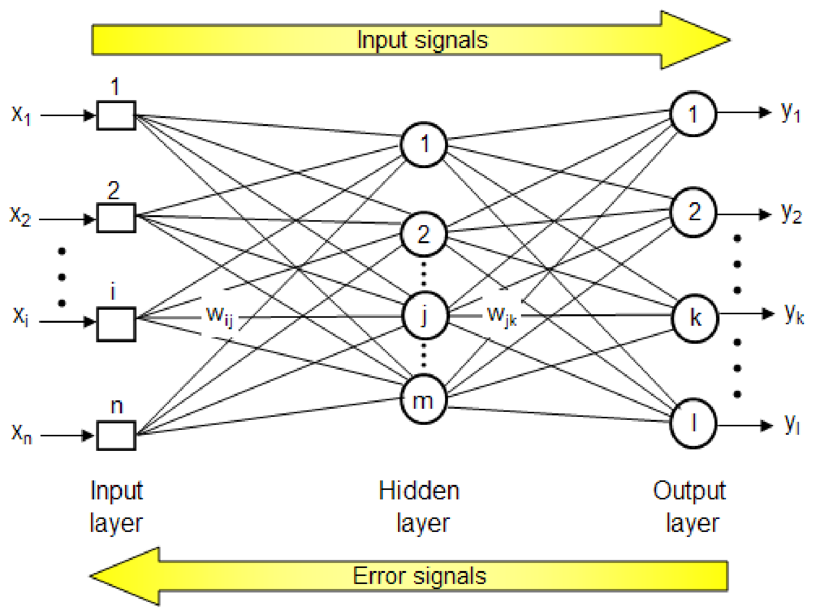
Model Evaluation Metrics:

To evaluate the house price prediction model, several metrics were employed:

- **Mean Squared Error (MSE):** This metric quantifies the average squared difference between the predicted and actual house prices. A lower MSE indicates better model performance.
- **Root Mean Squared Error (RMSE):** The square root of MSE provides an error metric in the same units as the target variable (price), facilitating interpretability.
- **Mean Absolute Error (MAE):** Measures the average absolute difference between predicted and actual prices, offering a more robust metric compared to MSE in the presence of outliers.
- **R-squared (R^2):** Indicates the proportion of variance in the house price explained by the model, ranging from 0 to 1, with higher values signifying better fit.
- **Adjusted R-squared:** A modified version of R-squared that penalizes the inclusion of unnecessary features, providing a more accurate estimate of model performance.

Model Performance Analysis:

The calculated metrics provide valuable insights into the model's predictive accuracy. A lower MSE, RMSE, and MAE indicate smaller prediction errors, while a higher R-squared suggests that the model explains a substantial portion of the variance in house prices.



22: Deep Learning Input & Error Signals

Error Analysis:

To gain a deeper understanding of the model's performance and identify potential areas for improvement, error analysis is crucial. This involves examining the distribution of prediction errors and investigating patterns or trends.

- **Residual Plots:** Visualizing the residuals (differences between predicted and actual prices) against predicted values or independent variables can reveal potential issues such as heteroscedasticity, non-linearity, or outliers.
- **Feature Importance:** Analyzing the contribution of different features to the model's predictions helps identify key factors influencing house prices and potential areas for feature engineering.

Assessing Model Generalization:

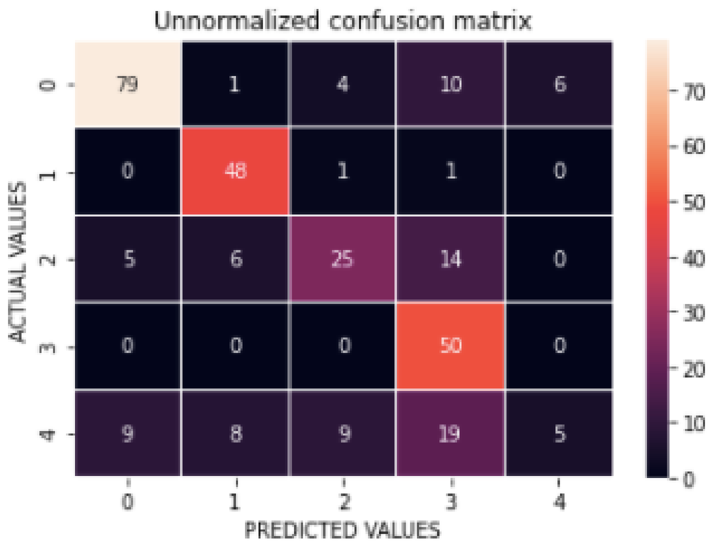
To ensure the model's ability to generalize to unseen data, it is essential to evaluate its performance on a holdout test set. This helps assess the model's predictive power in real-world scenarios.

Comparison with Baseline Models (Optional):

Comparing the deep learning model's performance with simpler models like linear regression or decision trees can provide insights into the added value of the deep learning approach.

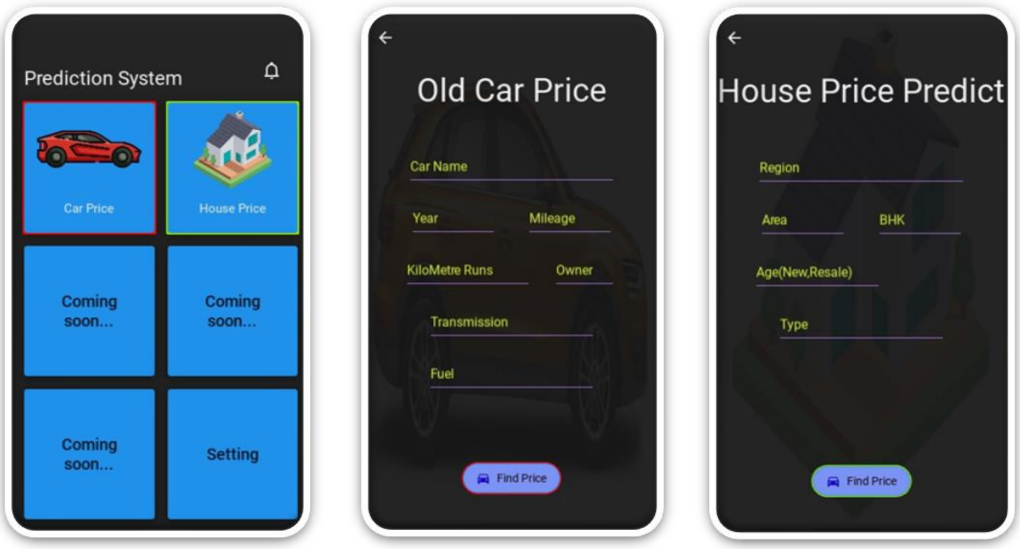
Limitations and Future Improvements:

While the deep learning model demonstrates promising results, it's essential to acknowledge its limitations and potential areas for improvement:



23: Confusion Matrix

- **Data Quality:** The quality and completeness of the house price dataset significantly impact model performance. Addressing data inconsistencies, missing values, and outliers is crucial.
- **Feature Engineering:** Exploring additional features or transformations can enhance the model's predictive power.
- **Model Architecture:** Experimenting with different deep learning architectures, such as recurrent neural networks or convolutional neural networks, might lead to performance gains.
- **Hyperparameter Tuning:** Fine-tuning hyperparameters through techniques like grid search or random search can optimize model performance.



24: GUI of Home, Car Price & Home price Input Page

By addressing these limitations and incorporating advanced techniques, the house price prediction model can be further refined and improved.

Chapter 8: CONCLUSION

The Predictor project successfully developed a robust system capable of accurately predicting car and house prices. By leveraging machine learning and deep learning techniques, the project addressed the complex challenges associated with price estimation in the automotive and real estate domains.

Summary of Findings:

The car price prediction model, based on linear regression, demonstrated strong performance in terms of accuracy and predictive power. The model effectively captured the relationship between car features and their corresponding prices. However, the limitations of linear regression, such as the assumption of linearity and sensitivity to outliers, highlight the potential for improvement through more sophisticated models.

The house price prediction model, employing deep learning, showcased the ability to learn complex patterns and relationships within the housing data. The model's performance, as measured by metrics like MSE, RMSE, MAE, and R-squared, indicated its effectiveness in accurately estimating house prices. However, further exploration of different deep learning architectures and hyperparameter tuning can potentially enhance predictive accuracy.

Project Contributions:

The Predictor project contributes to the field of price prediction by:

- Demonstrating the applicability of machine learning and deep learning techniques for accurate price estimation.
- Developing a robust system capable of handling diverse datasets and providing reliable predictions.
- Providing a foundation for future research and development in price prediction.

Limitations and Future Work:

While the Predictor system achieved promising results, there are areas for improvement and expansion:

- **Data Quality and Quantity:** The availability of high-quality, large-scale datasets is crucial for model performance. Incorporating additional data sources and addressing data quality issues can lead to more accurate predictions.
- **Model Complexity:** Exploring more complex machine learning and deep learning architectures, such as ensemble methods or advanced neural network structures, may enhance model performance.

- **Feature Engineering:** Developing innovative feature engineering techniques can unlock hidden patterns in the data and improve predictive accuracy.
- **Real-time Predictions:** Integrating real-time data sources and updating models dynamically can enhance the system's responsiveness to market changes.
- **User Interface Enhancements:** Expanding the user interface to include additional features, such as visualization tools and comparative analysis, can enhance user experience.

The Predictor project represents a significant step towards developing accurate and reliable price prediction systems. By addressing the challenges associated with car and house price estimation, the project has demonstrated the potential of machine learning and deep learning techniques in this domain. Future research and development efforts should focus on expanding the system's capabilities, incorporating advanced methodologies, and addressing the identified limitations to create even more powerful and accurate prediction tools.

Chapter 9: FUTURE WORK

The Conclusion section summarizes the key findings and outcomes of the "SecureInfo" software project. The following points highlight the main aspects to be covered in this section:

While the Predictor system has demonstrated promising results in car and house price prediction, there are several avenues for future exploration and enhancement.

Expanding Prediction Capabilities:

- **Additional Asset Classes:** The system can be extended to encompass a broader range of assets, such as electronics, collectibles, or real estate properties beyond houses.
- **Product Recommendation:** By incorporating user preferences and historical data, the system can offer product recommendations based on predicted prices and user behavior.

Enhancing Model Performance:

- **Advanced Machine Learning Techniques:** Exploring ensemble methods, such as gradient boosting or random forests, can potentially improve model accuracy and robustness.
- **Deep Learning Architectures:** Investigating more complex deep learning architectures, like recurrent neural networks (RNNs) or convolutional neural networks (CNNs), may capture intricate patterns in the data.
- **Hyperparameter Optimization:** Employing advanced hyperparameter tuning techniques, such as grid search, random search, or Bayesian optimization, can fine-tune model parameters for optimal performance.
- **Feature Engineering:** Continual exploration and refinement of feature engineering techniques can lead to the discovery of new informative features that enhance predictive accuracy.

Incorporating External Data:

- **Economic Indicators:** Integrating economic indicators, such as interest rates, inflation, and GDP growth, can improve model performance by capturing macroeconomic trends.
- **Market Trends:** Incorporating real-time market data, such as recent sales prices and consumer sentiment, can enhance the system's ability to adapt to changing market conditions.

User Interface Improvements:

- **Visualization Tools:** Developing interactive visualization tools to help users understand price trends and factors influencing predictions.
- **Personalized Recommendations:** Offering personalized recommendations based on user preferences, budget, and other relevant factors.
- **Mobile App Development:** Creating a mobile application to provide on-the-go access to the prediction system.

Ethical Considerations:

- **Fairness and Bias:** Ensuring that the models are fair and unbiased, avoiding discrimination based on protected attributes.
- **Data Privacy:** Implementing robust data privacy measures to protect user information.
- **Model Explainability:** Developing techniques to explain the reasoning behind model predictions, enhancing transparency and trust.

The Predictor project represents a solid foundation for accurate price prediction, but there is ample opportunity for growth and expansion. By addressing these areas for future work, the system can become an even more powerful and valuable tool for individuals and businesses.

Chapter 10: REFERENCES

The References section outlines the various resources and materials consulted during the development of the "Predictor" project.:

Datasets:

- Car Prices Dataset: https://www.kaggle.com/datasets/sidharth178/car-prices-dataset
- Used Car Price Prediction Dataset: https://www.kaggle.com/datasets/taeefnajib/used-car-price-prediction-dataset
- Mumbai House Prices: https://www.kaggle.com/datasets/dravidvaishnav/mumbai-house-prices

Libraries and Tools:

- Python: https://www.python.org/
- NumPy: https://numpy.org/
- Pandas: https://pandas.pydata.org/
- Matplotlib: https://matplotlib.org/
- Seaborn: https://seaborn.pydata.org/
- Scikit-learn: https://scikit-learn.org/stable/
- TensorFlow: https://www.tensorflow.org/
- Keras: https://keras.io/
- Kivy: https://kivy.org/

Machine Learning and Deep Learning Resources:

- Codebasics YouTube Channel: https://www.youtube.com/c/codebasics
- TensorFlow Tutorials: https://www.tensorflow.org/tutorials
- Keras Documentation: https://keras.io/guides/
- Scikit-learn Documentation: https://scikit-learn.org/stable/user_guide.html

Chapter 11: SOURCE CODE

11.1 GUI Code:

```
from kivy.lang import Builder
from kivy.uix.screenmanager import Screen, ScreenManager
from kivymd.app import MDApp
from kivymd.uix.boxlayout import MDBoxLayout
from kivymd.uix.label import MDLabel
from kivymd.uix.button import MDRaisedButton
from kivymd.uix.gridlayout import MDGridLayout
from kivymd.toast import toast
from kivy.core.window import Window
from sklearn import linear_model
import pickle
import pandas as pd
import numpy as np

# Set window size for mobile simulation
Window.size = (360, 640)

KV = '''
ScreenManager:
    MainScreen:
    CarpriceScreen:
    CarpriceResultScreen:
    HomepriceScreen:
    LocationScreen:
    ActivityScreen:
    ToDoScreen:
```

```
    SettingScreen:

<MainScreen>:
    name: "main"
    MDBoxLayout:
        orientation: 'vertical'
        md_bg_color: 0.1, 0.1, 0.1, .95
        padding: dp(16)
        spacing: dp(16)
        MDBoxLayout:
            size_hint_y: None
            height: self.minimum_height
            MDLabel:
                text: "Prediction System"
                font_style: "H5"
                theme_text_color: "Custom"
                text_color: 1, 1, 1, 1  # Magenta color (RGBA)
                size_hint_y: None
                height: self.texture_size[1]
            MDIconButton:
                icon: "bell-outline"
                theme_text_color: "Custom"
                text_color: 1, 1, 1, 1  # Magenta color (RGBA)
                pos_hint: {"center_y": 0.5}
        MDGridLayout:
            cols: 2
            spacing: dp(16)
            adaptive_height: True


            MDRaisedButton:
                orientation: 'vertical'
                size_hint: None, None
    size: dp(160), dp(160)
    on_release: app.card_clicked("carprice")
    canvas.before:
        Color:
            rgba: 1, 0, 0, 1  # Change to desired boundary color (red here)
        Line:
            width: 2
            rectangle: self.x, self.y, self.width, self.height
    BoxLayout:
        orientation: 'vertical'
        Image:
            source: 'sport_car.png'
            size_hint_y: None
            height: dp(100)
        Label:
            text: 'Car Price'
            text_color: 0, 0, 0, 1
            font_style: "H6"
            halign: 'center'

MDRaisedButton:
    orientation: 'vertical'
    size_hint: None, None
    size: dp(160), dp(160)
    on_release: app.card_clicked("homeprice")
    canvas.before:
        Color:
            rgba: 0.5647, 0.9333, 0.0078, 1  # Change to desired boundary color
        Line:
            width: 2
            rectangle: self.x, self.y, self.width, self.height
    BoxLayout:
        orientation: 'vertical'
        Image:
            source: 'home.png'
            size_hint_y: None
            height: dp(100)
        Label:
            text: 'House Price'
            theme_text_color: "Custom"
            font_style: "H6"
            halign: 'center'

MDRaisedButton:
    orientation: 'vertical'
    size_hint: None, None
    size: dp(160), dp(160)
    on_release: app.card_clicked("location")
    MDLabel:
        text: "Coming soon..."
        font_style: "H6"
        halign: "center"

MDRaisedButton:
    orientation: 'vertical'
    size_hint: None, None
    size: dp(160), dp(160)
    on_release: app.card_clicked("activity")
    MDLabel:
        text: "Coming soon..."
        font_style: "H6"
        halign: "center"

MDRaisedButton:
```

```
                orientation: 'vertical'
                size_hint: None, None
                size: dp(160), dp(160)
                on_release: app.card_clicked("todo")
                MDLabel:
                    text: "Coming soon..."
                    font_style: "H6"
                    halign: "center"

            MDRaisedButton:
                orientation: 'vertical'
                size_hint: None, None
                size: dp(160), dp(160)
                on_release: app.card_clicked("setting")
                MDLabel:
                    text: "Setting"
                    font_style: "H6"
                    halign: "center"

<CarpriceScreen>:
    name: "carprice"
    MDFloatLayout:
        padding: dp(16)
        spacing: dp(16)
        canvas.before:
            Rectangle:
                source: 'carimg.png'  # Path to your background image
                pos: self.pos
                size: self.size
        md_bg_color: 0.1, 0.1, 0.1, .95

        MDBoxLayout:
```

```
    size_hint_y: None
    pos_hint: {'center_x': .5, 'center_y': .95}
    height: self.minimum_height
    MDIconButton:
        icon: "arrow-left"
        theme_text_color: "Custom"
        text_color: 1, 1, 1, 1  # Magenta color (RGBA)
        on_release:
            app.root.current = "main"
            app.root.transition.direction = "right"

MDLabel:
    text: "Old Car Price"
    font_size: "40sp"
    pos_hint: {'center_x': .5, 'center_y': .85}
    halign: "center"
    theme_text_color: "Custom"
    text_color: 1, 1, 1, 1

MDTextField:
    id: nameCar
    pos_hint: {'center_x': .5, 'center_y': .7}
    size_hint_x: None
    width: "250dp"
    hint_text: "Car Name"
    helper_text: "Search the car name"
    helper_text_mode: "on_focus"
    text_color_normal: 0.7,1,1,1
    text_color_focus: 0.7,1,1,1
    line_color_normal: 0.7,0.5,0.9,1
    line_color_focus: 1, 1, 1, 1
    hint_text_color_normal: 0.9,1,0.2,1
    hint_text_color_focus: 1,1,0.1, 1

MDTextField:
    id: year
    pos_hint: {'center_x': .3, 'center_y': .6}
    size_hint_x: None
    width: "100dp"
    hint_text: "Year"
    helper_text: "Enter the year of buy car"
    helper_text_mode: "on_focus"
    max_text_length: 4
    text_color_normal: 0.7,1,1,1
    text_color_focus: 0.7,1,1,1
    line_color_normal: 0.7,0.5,0.9,1
    line_color_focus: 1, 1, 1, 1
    hint_text_color_normal: 0.9,1,0.2,1
    hint_text_color_focus: 1,1,0.1, 1

MDTextField:
    id: mileage
    pos_hint: {'center_x': .7, 'center_y': .6}
    size_hint_x: None
    width: "100dp"
    hint_text: "Mileage"
    helper_text: "Enter the year of buy car"
    helper_text_mode: "on_focus"
    max_text_length: 2
    text_color_normal: 0.7,1,1,1
    text_color_focus: 0.7,1,1,1
    line_color_normal: 0.7,0.5,0.9,1
    line_color_focus: 1, 1, 1, 1
    hint_text_color_normal: 0.9,1,0.2,1
    hint_text_color_focus: 1,1,0.1, 1

MDTextField:
    id: kilometre
    pos_hint: {'center_x': .35, 'center_y': .5}
    size_hint_x: None
    width: "150dp"
    hint_text: "KiloMetre Runs"
    text_color_normal: 0.7,1,1,1
    text_color_focus: 0.7,1,1,1
    line_color_normal: 0.7,0.5,0.9,1
    line_color_focus: 1, 1, 1, 1
    hint_text_color_normal: 0.9,1,0.2,1
    hint_text_color_focus: 1,1,0.1, 1

MDTextField:
    id: owner
    pos_hint: {'center_x': .75, 'center_y': .5}
    size_hint_x: None
    width: "70dp"
    hint_text: "Owner"
    # helper_text: "Enter the car first,second onership"
    helper_text_mode: "on_focus"
    max_text_length: 1
    text_color_normal: 0.7,1,1,1
    text_color_focus: 0.7,1,1,1
    line_color_normal: 0.7,0.5,0.9,1
    line_color_focus: 1, 1, 1, 1
    hint_text_color_normal: 0.9,1,0.2,1
    hint_text_color_focus: 1,1,0.1, 1

MDTextField:
    id: transmission
    pos_hint: {'center_x': .5, 'center_y': .4}
    size_hint_x: None
    width: "200dp"
    hint_text: "Transmission"
    text_color_normal: 0.7,1,1,1
    text_color_focus: 0.7,1,1,1
    line_color_normal: 0.7,0.5,0.9,1
    line_color_focus: 1, 1, 1, 1
    hint_text_color_normal: 0.9,1,0.2,1
    hint_text_color_focus: 1,1,0.1, 1

MDTextField:
    id: fuel
    pos_hint: {'center_x': .5, 'center_y': .3}
    size_hint_x: None
    width: "200dp"
    hint_text: "Fuel"
    text_color_normal: 0.7,1,1,1
    text_color_focus: 0.7,1,1,1
    line_color_normal: 0.7,0.5,0.9,1
    line_color_focus: 1, 1, 1, 1
    hint_text_color_normal: 0.9,1,0.2,1
    hint_text_color_focus: 1,1,0.1, 1

MDRoundFlatIconButton:
    pos_hint: {'center_x': .5, 'center_y': .1}
    text: "Find Price"
    icon: "car"
    text_color: 0.1,0,0, 1
    icon_color: 0.1,0,0.8, 1
    md_bg_color: 0.5,0.6,1, 1
```

```
            on_release: app.find_price()
            canvas.before:
                Color:
                    rgba: 1, 0, 0, 1  # Boundary color (blue in this case)
                Line:
                    width: 2
                    rounded_rectangle: self.x, self.y, self.width, self.height, 25

<CarpriceResultScreen>:
    name: "carprice_result"
    MDBoxLayout:
        md_bg_color: 0.1, 0.1, 0.1, .95
        orientation: 'vertical'
        padding: dp(16)
        spacing: dp(16)
        canvas.before:
            Rectangle:
                source: 'carimg.png'  # Path to your background image
                pos: self.pos
                size: self.size

        MDIconButton:
            icon: "arrow-left"
            theme_text_color: "Custom"
            text_color: 1, 1, 1, 1  # Magenta color (RGBA)
            on_release:
                app.root.current = "carprice"
                app.root.transition.direction = "right"

        MDLabel:
            id: result_label
            pos_hint: {'center_x': .5, 'center_y': .6}
            text: ""
            theme_text_color: "Custom"
            text_color: 1, 1, 1, 1  # Magenta color (RGBA)
            font_style: "H4"
            halign: "center"

        MDIcon:
            icon: "car"
            icon_color: 0.1, 0, 0.8, 1
            size: dp(40), dp(40)
            pos_hint: {'center_x': .5, 'center_y': .3}

<HomepriceScreen>:
    name: "homeprice"
    MDFloatLayout:
        padding: dp(16)
        spacing: dp(16)
        canvas.before:
            Rectangle:
                source: 'home.png'  # Path to your background image
                pos: self.pos
                size: self.size
        md_bg_color: 0.1, 0.1, 0.1, .95

        MDBoxLayout:
            size_hint_y: None
            pos_hint: {'center_x': .5, 'center_y': .95}
            height: self.minimum_height
            MDIconButton:
                icon: "arrow-left"
                theme_text_color: "Custom"
                text_color: 1, 1, 1, 1  # Magenta color (RGBA)
                on_release:
                    app.root.current = "main"
                    app.root.transition.direction = "right"

        MDLabel:
            text: "House Price Predict"
            font_size: "40sp"
            pos_hint: {'center_x': .5, 'center_y': .85}
            halign: "center"
            theme_text_color: "Custom"
            text_color: 1, 1, 1, 1

        MDTextField:
            id: nameregion
            pos_hint: {'center_x': .5, 'center_y': .7}
            size_hint_x: None
            width: "250dp"
            hint_text: "Region"
            helper_text: "Search the Region"
            helper_text_mode: "on_focus"
            text_color_normal: 0.7,1,1,1
            text_color_focus: 0.7,1,1,1
            line_color_normal: 0.7,0.5,0.9,1
            line_color_focus: 1, 1, 1, 1
            hint_text_color_normal: 0.9,1,0.2,1
            hint_text_color_focus: 1,1,0.1, 1

        MDTextField:
            id: area
            pos_hint: {'center_x': .3, 'center_y': .6}
            size_hint_x: None
            width: "100dp"
            hint_text: "Area"
            helper_text: "Enter the area in Squre Feet"
            helper_text_mode: "on_focus"
            max_text_length: 9
            text_color_normal: 0.7,1,1,1
            text_color_focus: 0.7,1,1,1
            line_color_normal: 0.7,0.5,0.9,1
            line_color_focus: 1, 1, 1, 1
            hint_text_color_normal: 0.9,1,0.2,1
            hint_text_color_focus: 1,1,0.1, 1

        MDTextField:
            id: bhk
            pos_hint: {'center_x': .7, 'center_y': .6}
            size_hint_x: None
            width: "100dp"
            hint_text: "BHK"
            helper_text: "Enter the BHK of house"
            helper_text_mode: "on_focus"
            max_text_length: 2
            text_color_normal: 0.7,1,1,1
            text_color_focus: 0.7,1,1,1
            line_color_normal: 0.7,0.5,0.9,1
            line_color_focus: 1, 1, 1, 1
            hint_text_color_normal: 0.9,1,0.2,1
            hint_text_color_focus: 1,1,0.1, 1

        MDTextField:
            id: age
            pos_hint: {'center_x': .35, 'center_y': .5}
            size_hint_x: None
            width: "150dp"
            hint_text: "Age(New,Resale)"
            text_color_normal: 0.7,1,1,1
            text_color_focus: 0.7,1,1,1
            line_color_normal: 0.7,0.5,0.9,1
            line_color_focus: 1, 1, 1, 1
            hint_text_color_normal: 0.9,1,0.2,1
            hint_text_color_focus: 1,1,0.1, 1


        MDTextField:
            id: type
            pos_hint: {'center_x': .5, 'center_y': .4}
            size_hint_x: None
            width: "200dp"
            hint_text: "Type"
            text_color_normal: 0.7,1,1,1
            text_color_focus: 0.7,1,1,1
            line_color_normal: 0.7,0.5,0.9,1
            line_color_focus: 1, 1, 1, 1
            hint_text_color_normal: 0.9,1,0.2,1
            hint_text_color_focus: 1,1,0.1, 1


        MDRoundFlatIconButton:
            pos_hint: {'center_x': .5, 'center_y': .1}
            text: "Find Price"
            icon: "car"
            text_color: 0.1,0,0, 1
            icon_color: 0.1,0,0.8, 1
            md_bg_color: 0.5,0.6,1, 1
            on_release: app.find_price()
            canvas.before:
```

```
                Color:
                    rgba: 0.5647, 0.9333, 0.0078, 1 # Boundary color (blue in this case)
                Line:
                    width: 2
                    rounded_rectangle: self.x, self.y, self.width, self.height, 25


<LocationScreen>:
    name: "location"
    MDBoxLayout:
        orientation: 'vertical'
        padding: dp(16)
        spacing: dp(16)
        MDLabel:
            text: "Location"
            font_style: "H4"
            halign: "center"
        MDRaisedButton:
            text: "Go Back"
            on_release:
                app.root.current = "main"
                app.root.transition.direction = "right"

<ActivityScreen>:
    name: "activity"
    MDBoxLayout:
        orientation: 'vertical'
        padding: dp(16)
        spacing: dp(16)
        MDLabel:
            text: "Activity"
            font_style: "H4"
            halign: "center"
        MDRaisedButton:
            text: "Go Back"
            on_release:
                app.root.current = "main"
                app.root.transition.direction = "right"

<ToDoScreen>:
    name: "todo"
    MDBoxLayout:
        orientation: 'vertical'
        padding: dp(16)
        spacing: dp(16)
        MDLabel:
            text: "To Do"
            font_style: "H4"
            halign: "center"
        MDRaisedButton:
            text: "Go Back"
            on_release:
                app.root.current = "main"
                app.root.transition.direction = "right"

<SettingScreen>:
    name: "setting"
    MDBoxLayout:
        orientation: 'vertical'
        padding: dp(16)
        spacing: dp(16)
        MDLabel:
            text: "Setting"
            font_style: "H4"
            halign: "center"
        MDRaisedButton:
            text: "Go Back"
            on_release:
                app.root.current = "main"
                app.root.transition.direction = "right"
'''


class MainScreen(Screen):
    pass


class CarpriceScreen(Screen):
    pass


class CarpriceResultScreen(Screen):
    pass


class HomepriceScreen(Screen):
    pass


class LocationScreen(Screen):
    pass


class ActivityScreen(Screen):
    pass


class ToDoScreen(Screen):
    pass


class SettingScreen(Screen):
    pass
```

```
class TestApp(MDApp):
    def build(self):
        self.title = "Predictor"
        return Builder.load_string(KV)

    def card_clicked(self, screen_name):
        self.root.current = screen_name
        self.root.transition.direction = "left"

    def car_price(self):
        car_value_name=pd.read_csv("carnames.csv")

        # Dummy implementation for car price finding
        car_name = self.root.get_screen('carprice').ids.nameCar.text
        year = self.root.get_screen('carprice').ids.year.text
        mileage = self.root.get_screen('carprice').ids.mileage.text
        kilometre = self.root.get_screen('carprice').ids.kilometre.text
        owner= self.root.get_screen('carprice').ids.owner.text
        transmission = self.root.get_screen('carprice').ids.transmission.text
        fuel = self.root.get_screen('carprice').ids.fuel.text

        id = np.array(car_value_name[car_value_name[car_name] == 1]).flatten()

        # CNG    Diesel  LPG    Petrol
        num_ful=[0,0,0,0]
        if fuel=="CNG": num_ful=[1,0,0,0]
        elif fuel=="Diesel": num_ful=[0,1,0,0]
        elif fuel == "LPG": num_ful=[0,0,1,0]
        elif fuel == "Petrol": num_ful=[0,0,0,1]

        #nums
        y=int(year)
        m=int(mileage)
        k=int(kilometre)
        o=int(owner)

        # Automatic       Manual
        num_tran=[0,0]
        if transmission=="Automatic": num_tran=[1,0]
        elif transmission=="Manual": num_tran=[0,1]

        a = np.concatenate((id, np.array([y, k, m, o]),num_ful,num_tran))

        with open('carPraicePridect_pickle', 'rb') as f:
            model_price = pickle.load(f)
        price=model_price.predict([a])
        # result = f"Price for {car_name} ({year}), Mileage: {mileage}, KM: {kilometre}, Transmission: {transmission}, Fuel: {fuel}"
        return price

    def find_price(self):

        x=self.car_price()
        result = f"Price for car {x[0]}"

        self.root.get_screen('carprice_result').ids.result_label.text = result
        self.clear_fields()
        self.root.current = "carprice_result"
        self.root.transition.direction = "left"
    def clear_fields(self):
        self.root.get_screen('carprice').ids.nameCar.text = ''
        self.root.get_screen('carprice').ids.year.text = ''
        self.root.get_screen('carprice').ids.mileage.text = ''
        self.root.get_screen('carprice').ids.kilometre.text = ''
        self.root.get_screen('carprice').ids.transmission.text = ''
        self.root.get_screen('carprice').ids.fuel.text = ''
        self.root.get_screen('carprice').ids.owner.text = ''
TestApp().run()
```

11.2 carPriceOredict.ipynb Code:

```
 # car pridiction model
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
import pickle
import os, types
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0
# The following code accesses a file in your IBM Cloud Object Storage. It includes
your credentials.
# You might want to remove those credentials before you share the notebook.

cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='JBrGeRfYJT3wP7x3rkoyQ-oawrnMTJkSdqVeVfy2xpZb',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.eu-de.cloud-object-storage.appdomain.cloud')

bucket = 'carpricepridect-donotdelete-pr-fwtlvvfduaseap'
object_key = 'cardekho.csv'
body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
```

```
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df= pd.read_csv(body)

print(df.shape)

df.sample(10)


f_df=df.drop(['seller_type','engine','max_power','seats'],axis='columns')

f_df['mileage(km/ltr/kg)']=f_df['mileage(km/ltr/kg)'].fillna(f_df['mileage(km/ltr/kg)'].
median())

f_df.head()


y=f_df.selling_price

x1=pd.get_dummies(f_df.name)

car_value_name=x1.drop_duplicates().reset_index(drop=True)

car_value_name.head()


#acess row value

id=np.array(car_value_name[car_value_name['BMW X4 M Sport X
xDrive20d']==1]).flatten()

id1=np.array(car_value_name[car_value_name['Maruti Swift Dzire
VDI']==1]).flatten()

id


# mapping = {'First Owner': 4, 'Second Owner': 3, 'Third Owner': 2,

#           'Fourth & Above Owner': 1, 'Test Drive Car': 5}

mapping = {'First Owner': 4, 'Second Owner': 3, 'Third Owner': 2,

          'Fourth & Above Owner': 1, 'Test Drive Car': 5}

f_df.owner.unique()

f_df.owner=f_df.owner.map(mapping)

# year km_driven mileage(km/ltr/kg)

x2=f_df[['year', 'km_driven', 'mileage(km/ltr/kg)','owner']]

x2.head()
```

```
#fuel
x3=pd.get_dummies(f_df.fuel)
fuel=x3.drop_duplicates().reset_index(drop=True)
x3.head()

#transmission
x4=pd.get_dummies(f_df.transmission)
transmission=x4.drop_duplicates().reset_index(drop=True)
x4.head()

#x1=Car name, x2=Nunaric columns(4), x3=fuel, x4=transmission and y= Price
x=pd.concat([x1,x2,x3,x4], axis=1)
x.head()

model=linear_model.LinearRegression()
model.fit(x,y)
# x.isna().sum()
# f_df.isna().sum()

a=np.concatenate((id,np.array([2019,7500,16,4,0,1,0,0,1,0])))
a
model.predict([a])
model.score(x,y)
#0.9884697062360259
with open('carPraicePridect_pickle','wb') as f:
   pickle.dump(model,f)
with open('carPraicePridect_pickle','rb') as f:
   mp=pickle.load(f)
mp.coef_
```

11.3 HousePrice.ipynb Code:

```
import pandas as pd
df_1 = pd.read_csv('Mumbai_House_Prices.csv')
df_1.head(10)
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
df_1.info()
df_1[df_1.price_unit=="Cr"].head()
df_1.head()
# Convert prices from crores to lakhs
df_1.loc[df_1.price_unit == "Cr", 'price'] = df_1[df_1.price_unit == "Cr"].price * 100
# Set the price unit to "L"
df_1.price_unit = "L"
# Display the first few rows of the DataFrame
df_1.head()
df_1.sample(5)
df_1.nunique()

col_to_use=['locality','status','price_unit']
df_1.drop(df_1[col_to_use],axis='columns',inplace=True)
dataset=df_1
dataset.head()
dataset.isna().sum()

dataset=pd.get_dummies(dataset,drop_first=True)
dataset.head()
# cols_to_scale=['bhk','area','price']
# from sklearn.preprocessing import MinMaxScaler
# scaler=MinMaxScaler()
# dataset[cols_to_scale]=scaler.fit_transform(dataset[cols_to_scale])
# dataset.head()

for col in dataset:
    if dataset[col].dtypes=='bool':
        dataset[col].replace({True:1,False:0},inplace=True)
for col in dataset:
     print(f'{col} : {dataset[col].unique()}')
dataset.head()

x=dataset.drop('price',axis='columns')
y=dataset.price
from sklearn.model_selection import train_test_split
x_t,x_test,y_t,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
x_t.shape

pip install tensorflow
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
x_t.shape[1]
# Define the model architecture
model = keras.Sequential([
    keras.layers.Dense(1, input_shape=(x_t.shape[1],), activation='relu')
    # keras.layers.Dense(64, activation='linear'),
    # keras.layers.Dense(32, activation='linear'),
    # keras.layers.Dense(16, activation='linear'),
    # keras.layers.Dense(1, activation='linear')
])
```

```
# Compile the model

model.compile(optimizer='adam', loss='mean_squared_error',
metrics=['mean_absolute_error'])

# Fit the model

model.fit(x_t, y_t, epochs=10,
callbacks=[keras.callbacks.EarlyStopping(patience=5)])


model.evaluate(x_test,y_test)

model.predict(x_test)
```

****