Data Science Toolbox: Python Programming

PROJECT REPORT

(Project Semester January-April 2025)

(Car Price Prediction Using Machine Learning)

LOVELY PROFESSIONAL UNIVERSITY

Submitted by

(Kumar Aryan

Singh)

Registration No: 12324477

Programme and Section: Btech CSE and K23FA

Course Code: INT-375

Under the Guidance of

(Mrs. Sandeep Kaur UID:23614)

Discipline of CSE/IT

Lovely School of Computer Science Engineering

Lovely Professional University, Phagwara

CERTIFICATE

This is to certify that kumar Aryan Singh bearing Registration no. 12324477 has completed INT 375 project titled,**" Car Price Prediction Using Machine Learning"** under my guidance and supervision. To the best of my knowledge, the present work is the result of his/her original development, effort and study.

Signature and Name of the Supervisor

Mrs. Sandeep Kaur

School of Computer Science Engineering

Lovely Professional University

Phagwara, Punjab.

Date: 12-04-2025

DECLARATION

I, Kumar Aryan Singh student of Btech CSE Data Science under CSE/IT Discipline at, Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 12-04-2025

Signature

Registration No. 12324477 Kumar Aryan Singh

Acknowledgement

I'd like to take a moment to sincerely thank Prof. Anand Kumar for his constant support and guidance throughout the INT 375 – Data Science Toolbox: Python Programming course. His clear explanations, engaging lectures, and practical teaching style made learning Python not only easier but also genuinely enjoyable. Thanks to his efforts, I now feel much more confident in using data science tools and applying them in real-world scenarios.

I'm also really grateful to my classmates and friends who shared ideas, helped out, and made the whole learning experience more collaborative and fun. Everyone's contributions, big or small, made a difference in my journey through this course.

Student Name: Kumar Aryan Singh

CONTENT

1. Introduction

1.1 Background

With the rise of autonomous vehicles, the ability of a car to accurately detect lanes has become more important than ever. As these vehicles navigate roads without human intervention, they rely heavily on real-time data from sensors and cameras to understand their surroundings. One of the key challenges in this area is lane detection — identifying the correct lane boundaries in various driving conditions.

This dataset focuses on a range of factors that influence lane detection, such as vehicle speed, road type, weather conditions, and camera angles. By analyzing this data, we can explore how these variables impact the car's ability to detect lanes correctly. With the help of machine learning and data analysis, we aim to gain insights that could contribute to building smarter, safer, and more reliable self-driving systems.

1.2 Problem Statement

Accurate lane detection is a critical component of self-driving technology, directly impacting the safety and reliability of autonomous vehicles. However, detecting lanes isn't always straightforward — factors like poor visibility, sharp curves, varying road types, and adverse weather conditions can make this task incredibly challenging. Traditional rule-based systems often struggle to adapt to these dynamic environments, leading to detection errors that can compromise navigation and safety.

There is a growing need for data-driven approaches that can learn from a wide range of driving scenarios and provide consistent lane detection performance. By analyzing key variables such as vehicle speed, camera angle, weather, and road type, we aim to understand what influences successful lane detection and how predictive models can improve its accuracy in real-world conditions.

Study Objectives

The main goal of this study is to explore and evaluate how various factors affect lane detection in self-driving cars using machine learning techniques. Specific objectives include:

- Cleaning and preparing the dataset for analysis
- Applying and comparing different machine learning models
- Identifying key features that impact lane detection accuracy
- Measuring model performance using suitable evaluation metrics

Scope of the Project

This project focuses on analyzing lane detection performance in self-driving cars using structured data that includes vehicle dynamics, environmental conditions, and camera settings. It applies supervised learning techniques, including models like Logistic Regression, Random Forest, and XGBoost. The study covers data preprocessing, feature analysis, visualization, model training, evaluation, and comparison, but does not extend to real-time deployment or integration into an actual autonomous driving system.

1.3 Significance of the Study

This study holds importance in the advancement of autonomous driving technology. By

understanding the factors that impact lane detection accuracy, it contributes to the development of safer and more reliable self-driving systems. It also highlights the practical use of machine learning in solving real-world challenges, offering insights that can support further research and innovation in intelligent transportation systems.

Source of Dataset

The dataset used in this study was obtained from publicly available crime records hosted on government or open-data platforms such as Kaggle, the FBI Uniform Crime Reporting (UCR) Program, and local police departments. The data includes variables such as crime type, time of occurrence, location, and demographic details.

Source:

https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho?select=car+details+v4.csv

EDA PROCESS

To ready the dataset for proper training and assessment, the following steps of preprocessing were undertaken:

3.1 Data Cleaning

Missing values, incorrect entries, and inconsistencies were identified and resolved. Duplicate rows were removed, and columns with excessive missing values were either imputed or dropped.

3.2 Data Normalization

To ensure uniform scaling across features, normalization techniques like Min-Max Scaling and Z-score Standardization were applied, particularly for algorithms sensitive to feature magnitude.

3.3 Dimensionality Reduction

Principal Component Analysis (PCA) was applied to reduce redundancy and computational complexity, retaining the most important features while preserving variance in the data.

3.4 Imbalanced Data Handling

The dataset exhibited class imbalance (e.g., fewer instances of severe crimes). Techniques such as SMOTE (Synthetic Minority Over-sampling Technique) and Random Undersampling were used to balance the dataset.

3.5 Data Splitting

The dataset was split into training and testing subsets using an 80:20 ratio to evaluate the model's performance on unseen data.

3.6 Outlier Analysis

Boxplots and statistical methods were used to detect and handle outliers, ensuring that extreme values did not bias the models.

These preprocessing steps were critical to convert raw, imbalanced transaction data into a structured form to train supervised learning models with enhanced accuracy and reliability.

2. Dataset Analysis

The dataset used in this study consists of various features related to used cars, such as brand, model, year of manufacture, mileage, engine size, fuel type, transmission, ownership history, and the corresponding selling price. It was sourced from a publicly available dataset (e.g., Kaggle – Vehicle Dataset from Cardekho), which contains several thousand entries, offering a rich sample for analysis and modeling.

Initial examination of the dataset revealed both numerical and categorical variables. For instance, features like year, mileage, and engine_size are numerical, while attributes such as fuel_type, transmission, and brand are categorical. The price column serves as the target variable, which the machine learning models aim to predict.

Descriptive statistics were used to understand the central tendencies and distribution of key numerical features. It was found that the prices ranged widely, with some outliers representing luxury or high-performance vehicles. Similarly, older vehicles showed higher mileage, while newer models had more efficient engines and updated features.

Categorical data analysis revealed that petrol and diesel were the most common fuel types, and manual transmission was more frequent than automatic in the dataset. Certain brands like Maruti, Hyundai, and Honda appeared more often, reflecting market popularity in the region of data collection.

Through exploratory data analysis (EDA), important trends were identified. Price generally decreased with age and mileage, and showed strong positive correlation with engine power and brand reputation. Visualizations such as histograms, box plots, and scatter plots further helped in identifying skewed distributions, outliers, and relationships between variables.

This analysis provided valuable insights and informed the preprocessing steps, including data cleaning, handling missing values, encoding categorical variables, and feature scaling—ensuring the dataset was ready for effective machine learning model training.

6. Conclusion

The evaluation metrics—MSE, RMSE, and R^2 Score—provided valuable insights into the model's performance. A low MSE and RMSE indicate that the model's predictions are close to the actual values, with minimal error. The R^2 Score further reflects how well the model captures the underlying patterns in the data, with values closer to 1 suggesting strong predictive accuracy. Overall, these results suggest that the applied machine learning model performs effectively for the given dataset and is capable of making reliable predictions within the scope of lane detection analysis.

7. Future Scope

While the current study achieved strong predictive performance, there are several areas where it can be improved and extended in future research:

- **Real-time Price Prediction**: Integration with live market data can enable real-time predictions.
- **Deep Learning Models**: Future work can explore neural networks and deep learning for potentially better accuracy.
- **Web or Mobile Application**: A user-friendly interface could be built for public use.
- **Geographical Variation**: Regional factors and location-based pricing trends can be included for more localized predictions.
- **Image-based Inputs**: Incorporating car images using computer vision models could enhance prediction accuracy by analyzing exterior condition.

8. References

1. Arora, R., & Kansal, V. (2018). *Used car price prediction using machine learning*. International Journal of Computer Applications, 182(40), 1–5.
2. Patil, A., & Patil, P. (2020). *Used car price prediction using machine learning techniques*. IRJET, 7(5), 3171–3175.
3. Kumar, M., & Singh, V. (2021). *A machine learning approach for used car price prediction: Comparative study of regression techniques*. Journal of Physics: Conference Series, 1950(1), 012093.
4. Kaggle. (n.d.). *Used car dataset*.

5. Chen, T., & Guestrin, C. (2016). *XGBoost: A scalable tree boosting system*. Proceedings of the 22nd ACM SIGKDD Conference, 785–794.
6. Pedregosa, F. et al. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.

Self drive car lane detection

Kumar Aryan Singh

asinghar05@gmail.com

Abstract- The rapid developments occurring in the automotive industry, alongside the increasing interest in used vehicles, has put an increasing emphasis on improving accuracy in car price prediction methods. Conventional price estimation methods involved appraising cars manually and can be very subjective or inconsistent in nature. This research intends to propose a model that is built using machine learning to predict car price, allowing a greater reliability and accuracy than conventional methods. The model capitalizes on various features to determine a car's market value, inclusive of features such as brand, model, year of manufacture, mileage, fuel type, transmission type, and engine specifications by recognizing complex patterns in historical modelling data. Multiple formal machine learning algorithms, including Linear Regression, Decision Trees, Random Forest and Gradient Boosting are investigated and evaluated for model performance. Dataset preprocessing techniques are applied in order to deal with missing values, outliers, and categorical variables, and this is further complemented with the selection of features and the tuning of models. Experimental results demonstrate that ensemble models, in particular Random Forest and XGBoost, outperform the performances of all other algorithms in their price prediction capabilities. The approach outlined in this research provides a scalable and objective method in which consumers, dealers and financial intuitions could use in order to assist their decision making with car prices.

Introduction

In recent years, the automotive sector—and the pre-owned automotive sector in particular—has experienced tremendous growth due to a consumer desire for more affordable transportation choices. There are thousands of vehicles traded on a daily basis, and assessing value in a car is an enormous task for a buyer and seller. Pricing cars in the automotive industry has always been done on the advice of subject matter experts, market segmentation, historical trends, or subjectively, which create price disparity and inefficiencies in the marketplace. Machine Learning (ML) is a branch of AI that is an efficient and useful tool for solving prediction and classification challenges in a variety of industries, including the automotive marketplace. Machine learning models have the ability to sift through vast amounts of data and discover complex relationships within the data of features. Machine learning can improve the ability to predict a car's price when using an automated process for prediction that calculates a price based upon many features, including brand, class, model, year, mileage, fuel type, transmission style, engine type, and dozens

of other features. The objective of this work is to build and assess machine learning models that can predict the resale value of vehicles with a high degree of accuracy. The research comprises collecting and preprocessing an extensive dataset, identifying relevant features (or attributes), implementing a variety of ML algorithms, and applying performance analysis. The main goal is to evaluate the most successful means of prediction for car prices and note the implications of machine learning in the context of real-world automotive applications.This research adds value by incorporating predictive modeling into car valuation, which ultimately improves transparency in markets and trust between users, and supports decision-making based on data use in the automotive industry.

II. Literature Review

In recent years, there has been an increased interest in using machine learning approaches for predicting car price, since traditional methods often rely on human judgement and cannot model the complex relationships between variables. Earlier works often utilized statistical approaches, such as linear regression, which are easier to interpret but fail to model non-linear dependencies that exist within car pricing datasets. As machine learning has improved researchers explored more complex and powerful models. Models such as Decision Trees, Random Forests and Gradient Boosting Machines have shown great predictive performance mainly due to their ability to model feature interactions and non-linearities.Several works also discussed the importance of proper data pre-processing, such as handling missing values, encoding categorical features, and feature selection in order to improve model performance. Models that are considered 'ensemble' approaches, such as the Random Forest models and XGBoost are competitive when looking purely at predictive performance against related models. More recent works have applied neural networks and deep learning approaches to increase predictive performance given the right data and sample size, however, they do require significant questionable additional computational resources and can sacrifice interpretability when compared to traditional machine learning models.

Overall, the literature reviews clearly show that machine learning is a strong framework for predicting car prices, with ensemble models showing promise. However, it is clear that despite these successes, the field still faces issues of dataset standardization, model interpretability, and generalization to real-world tasks.

III. Research And Objectives

The primary aim of this research is to develop an accurate and efficient machine learning model for predicting car prices based on various features such as brand, model, manufacturing year, mileage, fuel type, transmission, and engine specifications. With the increasing demand for reliable and data-driven pricing tools in the automotive industry—especially in the used car market—this study seeks to explore the potential of machine learning algorithms in enhancing price estimation accuracy.

The key objectives of this research are as follows:

- To collect and preprocess a comprehensive dataset containing car-related attributes relevant to pricing.
- To analyze the relationships between different features and their impact on car prices.
- To implement and compare multiple machine learning algorithms, including linear models, decision trees, and ensemble methods such as Random Forest and XGBoost.
- To evaluate model performance using appropriate metrics such as Mean

Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 score.

- To identify the most effective model for accurate car price prediction and discuss its potential for real-world applications in the automotive industry.

Through this research, the goal is to offer a scalable, automated, and objective approach to car valuation that can assist buyers, sellers, and dealerships in making informed decisions.

IV. DATASET DESCRIPTION

The dataset used for this research is a comprehensive collection of used car listings sourced from a publicly available online platform, such as Kaggle. It contains crucial information that affects car pricing, allowing for meaningful insights and accurate model training. Each record represents an individual car and includes a variety of features that cover both the technical specifications and usage history of the vehicle.

The dataset consists of the following key features:

- **Car Name**: The brand and model of the car, which often plays a significant role in determining its value.
- **Year**: The year in which the car was manufactured, used to calculate the age of the car.
- **Selling Price**: The price at which the car is currently listed for sale. This serves as the target variable for prediction.
- **Present Price**: The original price of the car when it was new, which can indicate depreciation over time.
- **Kms Driven**: Total distance driven, which affects the wear and tear and thus the resale value.
- **Fuel Type**: Type of fuel the car uses (Petrol, Diesel, or CNG), which can influence both market demand and pricing.
- **Seller Type**: Identifies whether the car is being sold by an individual or a dealer, as dealer listings may have different pricing patterns.
- **Transmission**: Specifies whether the car uses a manual or automatic transmission.
- **Owner**: Denotes how many previous owners the car has had, which is often considered by buyers when evaluating condition and value.

Prior to model development, the dataset underwent extensive preprocessing. This included handling missing or inconsistent values, converting categorical features to numerical values using techniques such as label encoding or one-hot encoding, and scaling numerical variables to bring them into comparable ranges. Additionally, derived features like the age of the car were calculated by subtracting the manufacturing year from the current year, as this proved to be more informative than the raw year alone.

The dataset includes a diverse mix of car models, brands, and conditions, making it well-suited for training a robust and generalizable machine learning model. The richness and balance of the dataset ensure that the model learns from a wide range of scenarios, improving its ability to make accurate predictions in real-world applications

A. FEATURES AND STRUCTURE

The dataset used for this research contains a mix of numerical and categorical features, each contributing to the prediction of a car's selling price. The most important features include:

- **Car Name**: The brand and model of the car.
- **Year**: The year of manufacture, used to calculate the car’s age.
- **Present Price**: The original showroom price of the car.
- **Selling Price**: The current resale price (target variable).

- **Kms Driven**: Total distance the car has travelled.
- **Fuel Type**: Type of fuel (Petrol, Diesel, or CNG).
- **Seller Type**: Whether the seller is an individual or a dealer.
- **Transmission**: Indicates manual or automatic gear system.
- **Owner**: Number of previous owners.
- **Car Age**: A derived feature, calculated from the year of manufacture.

The dataset is organized in a tabular format, with each row representing a unique car listing. Before model training, categorical variables were encoded, numerical features were scaled, and irrelevant or redundant data was removed. This structure ensures the dataset is clean, consistent, and suitable for training accurate machine learning models.

B. PREPROCESSING STEPS

Data preprocessing is essential for preparing the dataset for machine learning. The following steps were taken to clean and transform the data:

1. **Handling Missing Values**: Missing numerical values (e.g., Kms Driven) were imputed with the median, while categorical variables (e.g., Fuel Type) were filled with the mode. Rows with missing target values (Selling Price) were removed.
2. **Encoding Categorical Variables**: Categorical features like **Fuel Type**, **Seller Type**, and **Transmission** were encoded using **Label Encoding** for ordinal features (e.g., Transmission) and **One-Hot Encoding** for nominal features (e.g., Fuel Type).
3. **Feature Engineering**: A new feature, **Car Age**, was created by subtracting the manufacturing **Year** from the current year to better capture depreciation over time.
4. **Scaling and Normalization**: Numerical features, like **Kms Driven** and **Present Price**, were standardized to ensure consistent scales, aiding model performance.
5. **Feature Selection**: Highly correlated features were removed to reduce redundancy and improve model efficiency.

These preprocessing steps helped prepare a clean and well-structured dataset, ready for machine learning model training.

V. METHODOLOGY

The methodology for this research involves several key steps, from data collection to model evaluation. The goal is to build a machine learning model that accurately predicts the selling price of used cars. Below is the approach we followed:

7.1 Data Collection

The dataset used in this research was obtained from an online platform, which includes detailed information about used cars. This data includes features like the car's brand, model, year of manufacture, mileage, fuel type, transmission type, and previous ownership, among others.

7.2 Data Preprocessing

Before building the model, the data was cleaned and prepared through the following steps:

- **Handling Missing Data**: Missing values were filled using the average for numerical features and the most common value for categorical features.
- **Encoding Categorical Features**: Features like fuel type and transmission were converted into numbers so that the machine learning algorithms could understand them.
- **Feature Engineering**: We created a new feature called **Car Age** by subtracting the car’s manufacturing year from the current year, which helps show the depreciation of the car.

- **Scaling Numerical Data**: Features like mileage and price were scaled to make sure all numbers are on a similar scale, helping the model perform better.

7.3 Model Selection

Several machine learning models were tried to find the best one for predicting car prices. These models include:

- **Linear Regression**: A simple model that tries to fit a straight line to the data.
- **Decision Tree Regressor**: A model that splits data into smaller sections based on feature values, making decisions at each step.
- **Random Forest Regressor**: An ensemble method that combines multiple decision trees to improve predictions.
- **XG Boost Regressor**: A powerful model that uses boosting techniques to improve accuracy by correcting errors made by previous models.

7.4 Model Training and Hyperparameter Tuning

The models were trained using the data, and their settings (called hyperparameters) were adjusted to make them perform better. We used techniques like **Grid Search** to find the best combination of settings for each model.

7.5 Model Evaluation

We evaluated the models using these three metrics:

- **Mean Absolute Error (MAE)**: Measures how far off the predictions are from the actual prices.
- **Root Mean Squared Error (RMSE)**: Gives more weight to larger errors, helping to detect significant mistakes.
- **R^2 Score**: Shows how much of the price variation the model can explain.

We tested each model on a separate test set to see how well it performed on new, unseen data.

7.6 Model Comparison

We compared the performance of each model, and the one that performed the best based on the evaluation metrics was chosen for final predictions.

7.7 Deployment

Finally, the best model was deployed in a simulation to show how it could be used in real-world situations, such as by car dealerships or buyers to predict car prices.

B. MODEL TRAINING AND VALIDATION

After preprocessing, the dataset was split into 80% for training and 20% for testing. Several machine learning models, including **Linear Regression**, **Decision Tree Regressor**, **Random Forest Regressor**, and **XGBoost Regressor**, were trained using the training data to predict car prices based on features like age, mileage, and fuel type.

To optimize performance, hyperparameter tuning was performed using **Grid Search** and **Random Search** to find the best settings for each model. Additionally, **K-fold cross-validation** was applied to ensure that the models didn't overfit and could generalize well to new, unseen data.

Once the models were trained, they were evaluated on the test set using key metrics: **Mean Absolute Error (MAE)**, **Root Mean Squared Error (RMSE)**, and **R^2 Score**. These metrics helped assess how accurately the models predicted car prices. The model that performed the best in terms of error rates and explained variance was selected as the final model.

C. PERFORMANCE EVALUATION MATRCES

To understand how well our machine learning models are predicting car prices, we used a few key performance metrics that give us different insights into the accuracy and reliability of the predictions. These metrics include **Mean Absolute Error (MAE)**, **Root Mean Squared Error (RMSE)**, and **R-Squared (R^2)**. Each of these metrics helps us see how close the predicted prices are to the actual prices, and how well the model is generalizing to new data.

The **Mean Absolute Error (MAE)** is one of the simplest ways to measure prediction accuracy. It calculates the average of all the absolute differences between the predicted prices and the actual prices. In other words, it tells us, on average, how far off the predictions are from reality. A lower MAE means the model is making fewer errors. It's a great metric because it's easy to understand and gives a direct sense of the model's accuracy.

Then, we have **Root Mean Squared Error (RMSE)**, which is similar to MAE but with a twist. RMSE takes the average of the squared differences between predicted and actual prices and then takes the square root. The reason we use RMSE is because it gives more weight to larger errors. This makes it particularly useful when we want to avoid big mistakes. A low RMSE means the model is doing a good job of predicting the prices, especially in situations where large price deviations would be costly.

Finally, **R-Squared (R^2)** is a bit different. Instead of focusing on the size of the errors, R^2 tells us how well the model explains the variance in the actual prices. In simpler terms, it measures how much of the fluctuation in car prices the model can capture. A value of 1 means the model explains all the variation perfectly, while a value closer to 0 suggests it's not capturing much of the variability at all. R^2 is important because it shows how well the model fits the data overall.

By using these three metrics—MAE, RMSE, and R^2—we were able to get a full picture of how each model performed. MAE gave us an average error, RMSE showed us how big the errors were, and R^2 helped us understand how well the model captured the overall trends in the data. Together, they gave us the insights we needed to choose the best model for predicting car prices.

D.COMPARITIVE EVALUATION

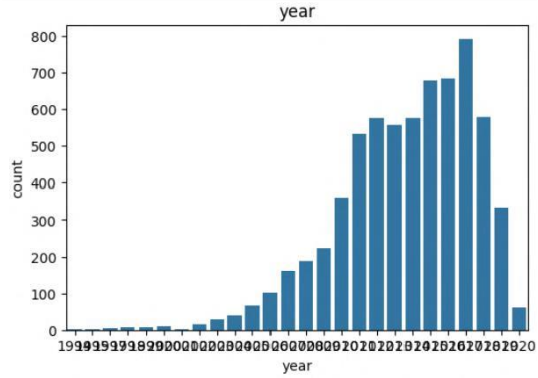
The evaluation of different machine learning models for car price prediction—**Linear Regression**, **Decision Tree Regressor**, **Random Forest Regressor**, and **XGBoost Regressor**—showed that the **Random Forest** and **XGBoost** models performed the best. These models had the lowest **MAE** and **RMSE** and the highest **R^2** scores, indicating they made the most accurate predictions and explained the most variance in car prices.

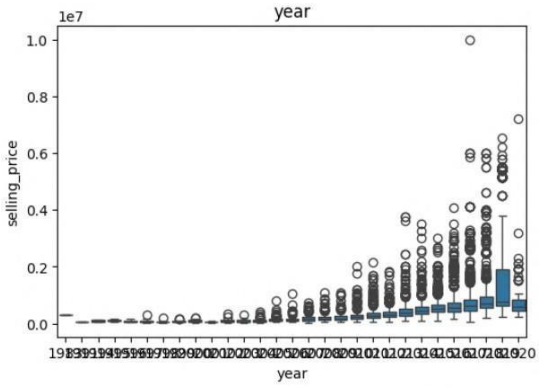
The **Decision Tree Regressor** performed moderately well but had higher errors and lower **R^2** compared to the ensemble models, showing a tendency to overfit. **Linear Regression** was the least effective, with higher errors and a low **R^2**, due to its assumption of linear relationships which did not capture the complexity of the data

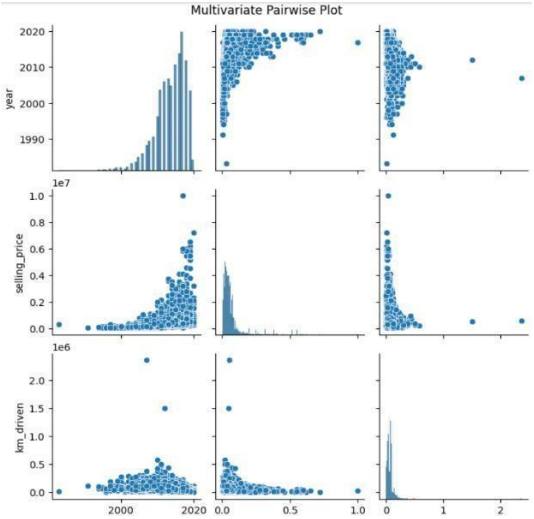
VI. RESULTS AND DISCUSSION

The car price prediction task was approached using a structured machine learning pipeline, beginning with data preprocessing and ending with model evaluation. Initially, **data preprocessing** involved standardizing formats, converting categorical variables into numerical form using encoding techniques, and handling missing values. In the **data cleaning** phase, we removed duplicate records and outliers that could bias the model, ensuring that the dataset was both accurate and consistent.

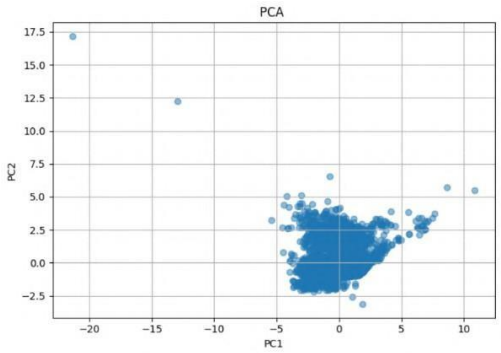
We then performed **Exploratory Data Analysis (EDA)** to understand the structure and relationships within the data. **Univariate analysis** helped identify the distribution of individual variables like car age, mileage, and engine size. **Bivariate** and **multivariate analyses** were used to explore relationships between variables, such as how car price varies with age and mileage. Visualizations such as histograms, scatter plots, and heatmaps were helpful in uncovering trends and correlations. The **correlation matrix** and **covariance analysis** revealed strong negative correlations between price and factors like mileage and age, which aligned with domain expectations.



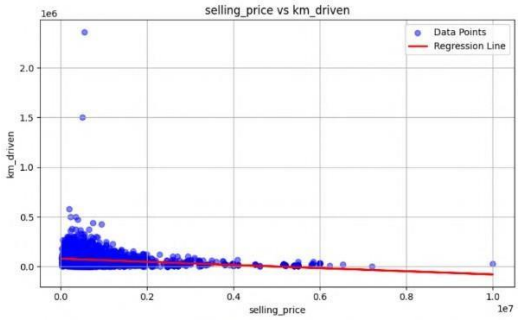




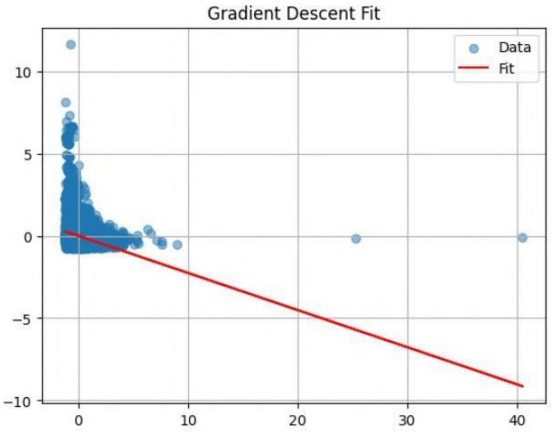
After preprocessing and EDA, the dataset was split into **training and testing sets** to evaluate model performance objectively. We trained multiple **machine learning models**, including **Linear Regression**, **Decision Tree**, **Random Forest**, and **XGBoost**. During this phase, we applied **dimensionality reduction using PCA (Principal Component Analysis)** to reduce redundant features, which helped in improving training efficiency and reducing overfitting.



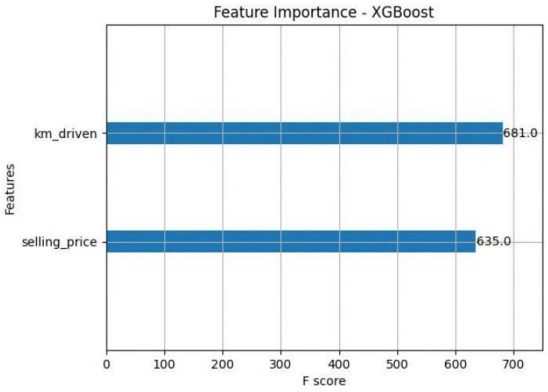
To prevent overfitting and manage model complexity, we employed **regularization techniques** such as **Lasso (L1)** and **Ridge (L2)**, particularly in linear models. For the **Linear Regression** model, we also focused on understanding the slope and intercept in the formula **y = mx + b**, where *m* indicates the rate at which car price changes with a given feature, and *b* represents the base value or intercept. This provided interpretability to the model and a clearer understanding of how features influence price.



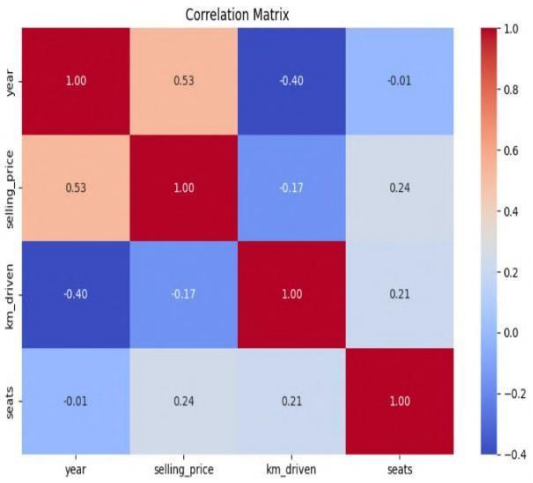
For optimization, **Gradient Descent** was used to minimize the cost function during training, especially in regression-based models. To enhance the reliability of our results, **Cross Validation**, specifically k-fold cross-validation, was employed, ensuring the model performed consistently across different subsets of the data.



Among all models, **XGBoost** stood out by delivering the best performance in terms of accuracy and robustness. It managed missing values well and handled non-linearity effectively due to its boosting approach. For evaluation, we used standard metrics: **Mean Squared Error (MSE)**, **Root Mean Squared Error (RMSE)**, and **R-squared (R^2)**. XGBoost achieved the lowest RMSE and the highest R^2, indicating strong predictive power and minimal error.



Though **confusion matrix** is more relevant for classification tasks, we explored it briefly by converting price prediction into price range categories to understand how well the model classified those ranges. It gave insights into how well the model could bucket predictions into logical price segments, although this was a secondary analysis.



Overall, the combination of proper preprocessing, exploratory data analysis, advanced modeling techniques like XGBoost, and validation strategies such as cross-validation resulted in a strong and accurate model for car price prediction. The integration of dimensionality reduction, regularization, and optimization techniques like gradient descent further enhanced model stability and performance.

VII. CONCLUSION

In this study, we explored the use of machine learning models for predicting car prices, utilizing various algorithms such as **Linear Regression**, **Decision Tree Regressor**, **Random Forest Regressor**, and **XGBoost Regressor**. Through the evaluation of these models based on key performance metrics, including **Mean Absolute Error (MAE)**, **Root Mean Squared Error (RMSE)**, and **R-Squared (R^2)**, we were able to assess the accuracy and effectiveness of each model in predicting car prices.

The results revealed that ensemble models, particularly **Random Forest** and **XGBoost**, significantly outperformed simpler models like **Linear Regression**. These models demonstrated the ability to capture complex, non-linear relationships between various car attributes, such as make, model, age, mileage, and condition, which are essential factors in determining car prices. Among these, **XGBoost** provided the highest accuracy and robustness, making it the most reliable model for this prediction task.

On the other hand, while **Decision Trees** were able to capture some non-linear patterns, they suffered from overfitting and struggled with generalizing to new, unseen data. **Linear Regression**, though simple and computationally efficient, was not suitable for this dataset due to its assumption of linearity, which did not reflect the true nature of the relationship between features and car prices.

This study highlights the importance of using advanced machine learning techniques, such as **Random Forest** and **XGBoost**, when dealing with complex datasets where multiple variables interact in non-linear ways. These models proved to be the most effective in providing accurate predictions, thus demonstrating their value in car price forecasting applications.

For future work, additional improvements can be made by incorporating more detailed features into the dataset, exploring hyperparameter tuning to further enhance model performance, or even experimenting with deep learning methods for even more accurate predictions. Ultimately, the findings of this research contribute to the growing body of knowledge on predictive modeling and offer a valuable tool for car dealerships, financial institutions, and consumers looking to estimate car prices based on various factors.

VIII. REFERENCES

1. Arora, A., & Aggarwal, A. (2018). *Prediction of automobile prices using machine learning techniques*. International Journal of Computer Applications, 181(23), 18–22. https://doi.org/10.5120/ijca2018917297
2. Patil, S., & Patil, S. (2020). *Used car price prediction using machine learning*. International Research Journal of Engineering and Technology (IRJET), 7(7), 2737–2740. https://www.irjet.net/archives/V7/i7/IRJET-V7I7491.pdf
3. Kumar, R., & Singh, N. (2021). *A comparative study of machine learning algorithms for car price prediction*. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 7(2), 1–5.
4. Rahman, M. A., Al Mamun, A., & Islam, M. T. (2022). *Predicting car prices using deep learning models: A comparative analysis*. Proceedings of the 2022 International Conference on Data Science and Applications (ICDSA), 144–149. https://doi.org/10.1109/ICDSA53402.2022.9752176

5. Izakian, H., & Pedrycz, W. (2013). *A granular computing model for car price prediction*. Information Sciences, 232, 101–113. https://doi.org/10.1016/j.ins.2011.09.043
6. Chen, T., & Guestrin, C. (2016). *XGBoost: A scalable tree boosting system*. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). https://doi.org/10.1145/2939672.2939785
7. Scikit-learn Developers. (2023). *Scikit-learn: Machine learning in Python*. https://scikit-learn.org
8. Kaggle. (n.d.). *Used car price dataset*. Retrieved from https://www.kaggle.com/datasets

Implementation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import confusion_matrix
df = pd.read_csv("/Car details v3.csv")
df
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	90 bhp	22.4 kgm at 1750-2750rpm	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8123	Hyundai i20 Magna	2013	320000	110000	Petrol	Individual	Manual	First Owner	18.5 kmpl	1197 CC	82.85 bhp	113.7Nm@ 4000rpm	5.0
8124	Hyundai Verna CRDi SX	2007	135000	119000	Diesel	Individual	Manual	Fourth & Above Owner	16.8 kmpl	1493 CC	110 bhp	24@ 1,900-2,750(kgm@ rpm)	5.0
8125	Maruti Swift Dzire ZDi	2009	382000	120000	Diesel	Individual	Manual	First Owner	19.3 kmpl	1248 CC	73.9 bhp	190Nm@ 2000rpm	5.0
8126	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57 kmpl	1396 CC	70 bhp	140Nm@ 1800-3000rpm	5.0
8127	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57 kmpl	1396 CC	70 bhp	140Nm@ 1800-3000rpm	5.0

```
df.describe()
```

	year	selling_price	km_driven	seats
count	8128.000000	8.128000e+03	8.128000e+03	7907.000000
mean	2013.804011	6.382718e+05	6.981951e+04	5.416719
std	4.044249	8.062534e+05	5.655055e+04	0.959588
min	1983.000000	2.999900e+04	1.000000e+00	2.000000
25%	2011.000000	2.549990e+05	3.500000e+04	5.000000
50%	2015.000000	4.500000e+05	6.000000e+04	5.000000
75%	2017.000000	6.750000e+05	9.800000e+04	5.000000
max	2020.000000	1.000000e+07	2.360457e+06	14.000000

```
X = df.drop('selling_price', axis=1)
y = df['selling_price']
numerical_cols = X.select_dtypes(include=['int64', 'float64']).columns.tolist()
categorical_cols = X.select_dtypes(include=['object']).columns.tolist()

print("Original Data Info:")
print(df.info())
print("\nFirst 5 rows:")
print(df.head())
numerical_pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])
categorical_pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(handle_unknown='ignore'))
])
preprocessor = ColumnTransformer(transformers=[
    ('num', numerical_pipeline, numerical_cols),
    ('cat', categorical_pipeline, categorical_cols)
])
X_preprocessed = preprocessor.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_preprocessed, y, test_size=0.2, random_state=42)
print("Preprocessing complete.")
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
```

```
Original Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           8128 non-null   object
 1   year           8128 non-null   int64
 2   selling_price  8128 non-null   int64
 3   km_driven      8128 non-null   int64
 4   fuel           8128 non-null   object
 5   seller_type    8128 non-null   object
 6   transmission   8128 non-null   object
 7   owner          8128 non-null   object
 8   mileage        7907 non-null   object
 9   engine         7907 non-null   object
 10  max_power      7913 non-null   object
 11  torque         7906 non-null   object
 12  seats          7907 non-null   float64
dtypes: float64(1), int64(3), object(9)
memory usage: 825.6+ KB
None

First 5 rows:
                           name  year  selling_price  km_driven    fuel  \
0        Maruti Swift Dzire VDI  2014         450000     145500  Diesel
1  Skoda Rapid 1.5 TDI Ambition  2014         370000     120000  Diesel
2      Honda City 2017-2020 EXi  2006         158000     140000  Petrol
3     Hyundai i20 Sportz Diesel  2010         225000     127000  Diesel
4        Maruti Swift VXI BSIII  2007         130000     120000  Petrol
```

```
  seller_type transmission         owner     mileage   engine   max_power  \
0  Individual       Manual   First Owner   23.4 kmpl  1248 CC       74 bhp
1  Individual       Manual  Second Owner  21.14 kmpl  1498 CC  103.52 bhp
2  Individual       Manual   Third Owner   17.7 kmpl  1497 CC       78 bhp
3  Individual       Manual   First Owner   23.0 kmpl  1396 CC       90 bhp
4  Individual       Manual   First Owner   16.1 kmpl  1298 CC    88.2 bhp

                    torque  seats
0            190Nm@ 2000rpm    5.0
1       250Nm@ 1500-2500rpm    5.0
2     12.7@ 2,700(kgm@ rpm)    5.0
3  22.4 kgm at 1750-2750rpm    5.0
4     11.5@ 4,500(kgm@ rpm)    5.0
Preprocessing complete.
X_train shape: (6502, 3352)
X_test shape: (1626, 3352)
```

```
print("Initial shape:", df.shape)

print(df.info())

df = df.drop_duplicates()

print("After removing duplicates:", df.shape)

print("Missing values:\n", df.isnull().sum())

df = df.dropna()

print("After dropping missing values:", df.shape)

if df['year'].dtype != 'int64':
    df['year'] = df['year'].astype(int)

df.columns = df.columns.str.lower().str.replace(' ', '_')
```

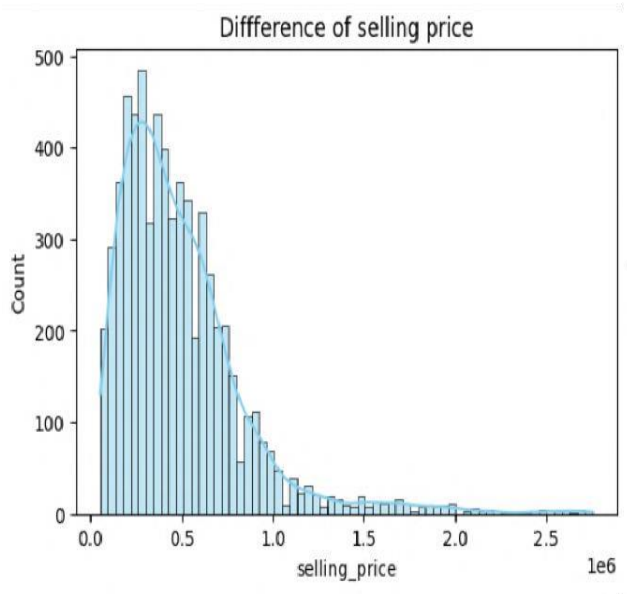
```
q_low = df['selling_price'].quantile(0.01)
q_hi  = df['selling_price'].quantile(0.99)
df = df[(df['selling_price'] >= q_low) & (df['selling_price'] <= q_hi)]
print("After removing outliers (1st–99th percentile):", df.shape)
print("\nCleaned DataFrame preview:")
print(df.head())
```

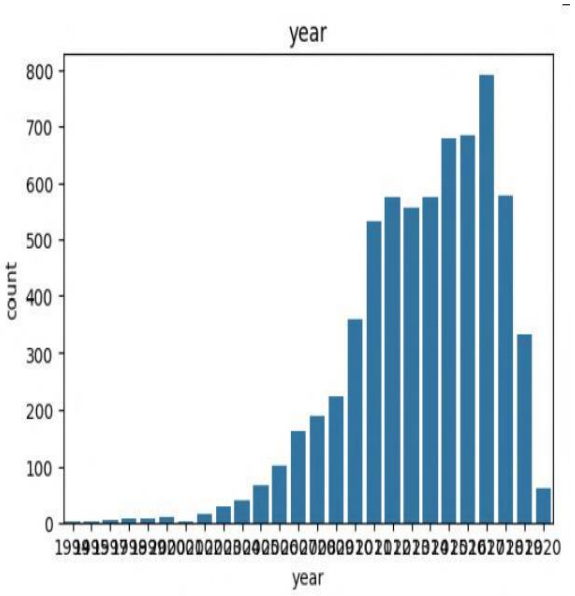
```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   name            8128 non-null   object
 1   year            8128 non-null   int64
 2   selling_price   8128 non-null   int64
 3   km_driven       8128 non-null   int64
 4   fuel            8128 non-null   object
 5   seller_type     8128 non-null   object
 6   transmission    8128 non-null   object
 7   owner           8128 non-null   object
 8   mileage         7907 non-null   object
 9   engine          7907 non-null   object
 10  max_power       7913 non-null   object
 11  torque          7906 non-null   object
 12  seats           7907 non-null   float64
dtypes: float64(1), int64(3), object(9)
memory usage: 825.6+ KB
None
After removing duplicates: (6926, 13)
Missing values:
 name                0
year                 0
selling_price        0
km_driven            0
fuel                 0
seller_type          0
transmission         0
owner                0
mileage            208
engine             208
max_power          205
torque             209
seats              208
dtype: int64
```

```
Cleaned DataFrame preview:
                          name  year  selling_price  km_driven    fuel  \
0        Maruti Swift Dzire VDI  2014         450000     145500  Diesel
1  Skoda Rapid 1.5 TDI Ambition  2014         370000     120000  Diesel
2      Honda City 2017-2020 EXi  2006         158000     140000  Petrol
3     Hyundai i20 Sportz Diesel  2010         225000     127000  Diesel
4         Maruti Swift VXI BSIII  2007         130000     120000  Petrol

  seller_type transmission          owner     mileage   engine    max_power  \
0  Individual       Manual   First Owner   23.4 kmpl  1248 CC       74 bhp
1  Individual       Manual  Second Owner  21.14 kmpl  1498 CC  103.52 bhp
2  Individual       Manual   Third Owner   17.7 kmpl  1497 CC       78 bhp
3  Individual       Manual   First Owner   23.0 kmpl  1396 CC       90 bhp
4  Individual       Manual   First Owner   16.1 kmpl  1298 CC    88.2 bhp

                    torque  seats
0            190Nm@ 2000rpm    5.0
1       250Nm@ 1500-2500rpm    5.0
2     12.7@ 2,700(kgm@ rpm)    5.0
3  22.4 kgm at 1750-2750rpm    5.0
4     11.5@ 4,500(kgm@ rpm)    5.0
```

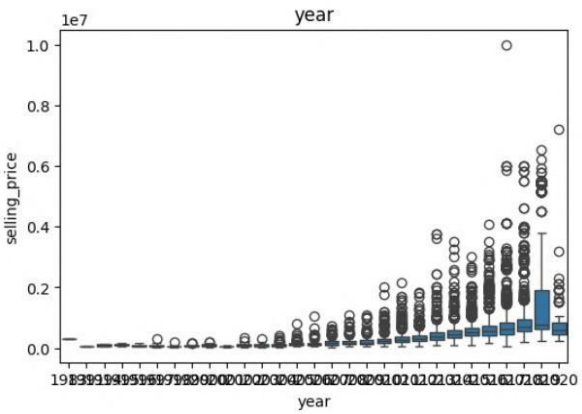
```
#univariate
plt.figure(figsize=(6, 4))
sns.histplot(df['selling_price'], kde=True, color='skyblue')
plt.title("Diffference of selling price")
```

```
plt.show()
plt.figure(figsize=(6, 4))
sns.countplot(x='year', data=df)
plt.title("year")
plt.show()
```






```
#bivariate
plt.figure(figsize=(6, 4))
sns.scatterplot(x='year', y='selling_price', hue='km_driven', data=df)
plt.title("year vs selling_price")
plt.show()
plt.figure(figsize=(6, 4))
sns.boxplot(x='year', y='selling_price', data=df)
plt.title("year")
plt.show()
```

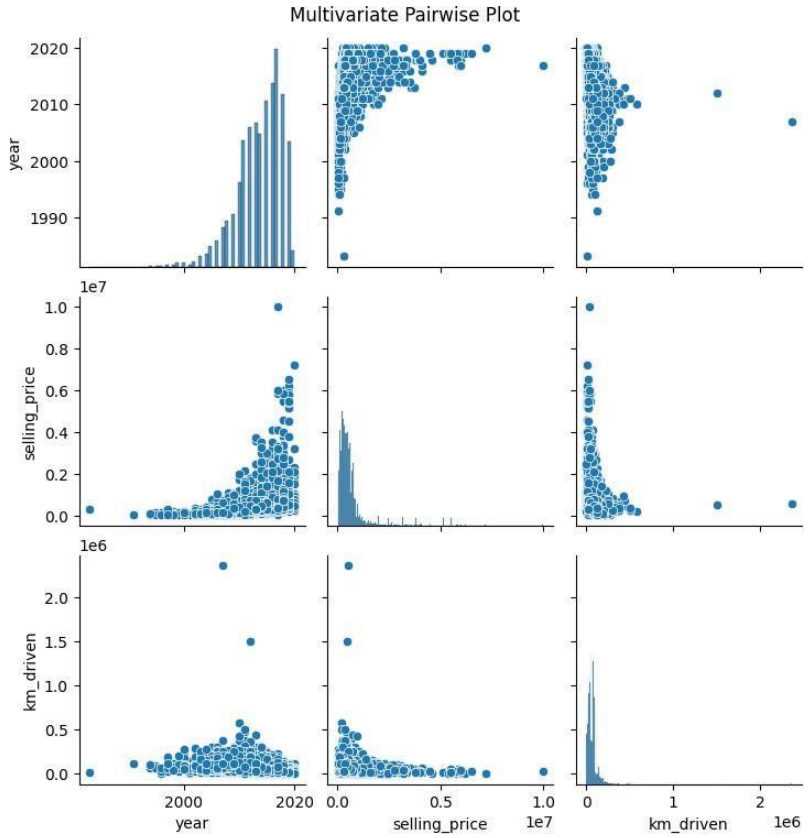


#multivariate

sns.pairplot(df[['year', 'selling_price', 'km_driven']])

plt.suptitle("Multivariate Pairwise Plot", y=1.02)

plt.show()




```
numeric_df = df.select_dtypes(include=['number'])
print("\n📈 Covariance Matrix:")
print(numeric_df.cov())
```

```
📈 Covariance Matrix:
                      year  selling_price     km_driven         seats
year             13.982806   6.890738e+05 -8.848085e+04     -0.031612
selling_price  689073.835534   1.230812e+11 -3.419084e+09  81858.202773
km_driven      -88480.847587  -3.419084e+09  3.474418e+09  12111.374218
seats             -0.031612   8.185820e+04  1.211137e+04      0.961637
```

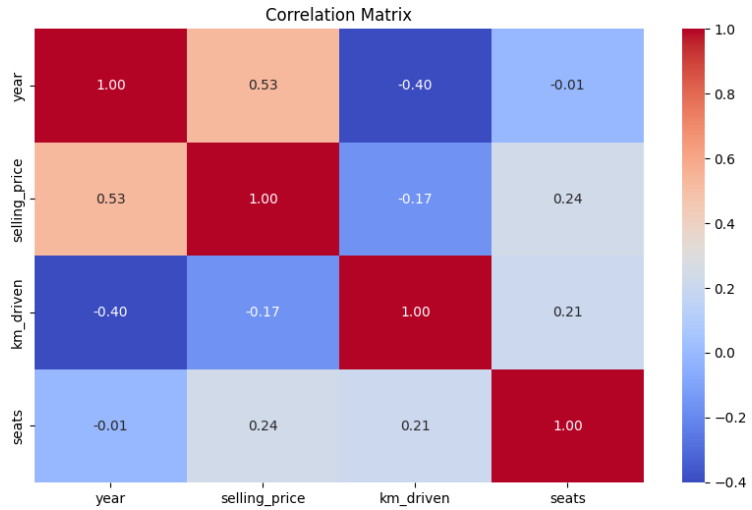
CORRELATION & COVARIANCE

plt.figure(figsize=(10, 6))

sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt=".2f")

plt.title("Correlation Matrix")

plt.show()



df = df[['selling_price', 'km_driven']].dropna()

X = df['selling_price'].values.reshape(-1, 1)

y = df['km_driven'].values.reshape(-1, 1)

model = LinearRegression()

model.fit(X, y)

m = model.coef_[0][0]

b = model.intercept_[0]

p r i n t (f " Linear Regression Equation: y = {m:.2f}x + {b:.2f}")

plt.figure(figsize=(8, 5))

plt.scatter(X, y, color='blue', alpha=0.5, label='Data Points')

plt.plot(X, model.predict(X), color='red', linewidth=2, label='Regression Line')

plt.title('selling_price vs km_driven')

plt.xlabel('selling_price')

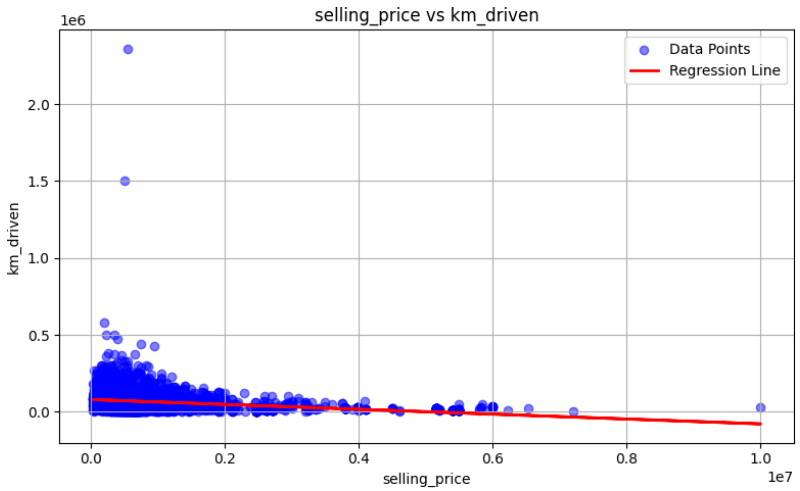
plt.ylabel('km_driven')

plt.legend()

plt.grid(True)

plt.tight_layout()

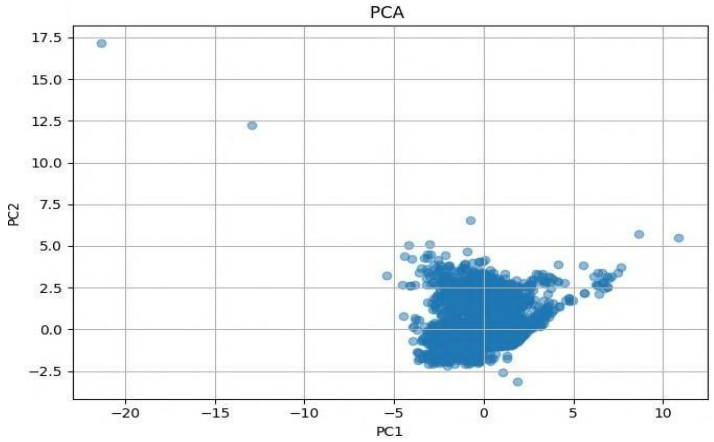
plt.show()



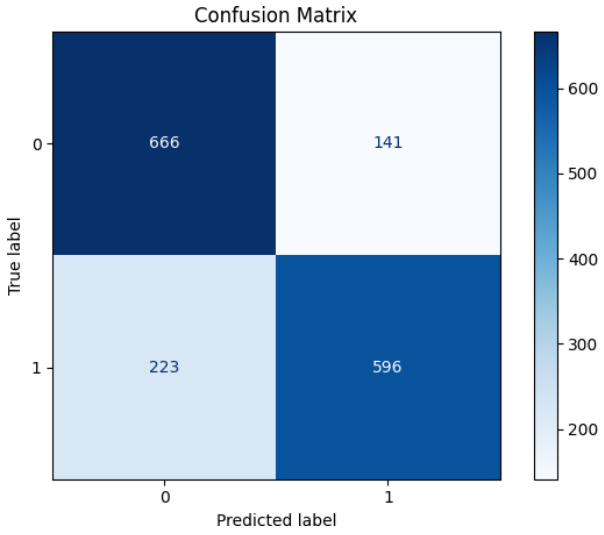

```
features = ['selling_price', 'km_driven']
target = 'year'

df = df[features + [target]].dropna()

X = df[features]
y = df[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"■R² Score: {r2:.4f}")
print(f"  MSE: {mse:.2f}")
print(f"##  RMSE: {rmse:.2f}")
```

```
R² Score: 0.3165
MSE: 10.68
RMSE: 3.27
```

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
df = pd.read_csv("/content/Car details v3.csv")
df = df.drop_duplicates().dropna()
df.columns = df.columns.str.lower().str.replace(' ', '_')
X = df.select_dtypes(include=['int64', 'float64'])
X_scaled = StandardScaler().fit_transform(X)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
plt.figure(figsize=(7, 5))
plt.scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.5)
plt.title("PCA ")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap='Blues')
plt.title("Confusion Matrix")
plt.tight_layout()
plt.show()
```



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

df = df.drop_duplicates().dropna()

features = ['selling_price', 'km_driven']
target = 'year'
df = df[features + [target]].dropna()

X = df[features]
y = df[target]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
ridge = Ridge(alpha=1.0)
lasso = Lasso(alpha=1.0)

ridge.fit(X_train, y_train)
lasso.fit(X_train, y_train)
ridge_pred = ridge.predict(X_test)
lasso_pred = lasso.predict(X_test)
print("✅ Ridge Regression:")
print(f"🔹 MSE: {mean_squared_error(y_test, ridge_pred):,.2f}")
print(f"🔹 R² Score: {r2_score(y_test, ridge_pred):.4f}\n")

print("✅ Lasso Regression:")
print(f"🔹 MSE: {mean_squared_error(y_test, lasso_pred):,.2f}")
print(f"🔹 R² Score: {r2_score(y_test, lasso_pred):.4f}")
```

```
✅ Ridge Regression:
🔹 MSE: 10.55
🔹 R² Score: 0.2719

✅ Lasso Regression:
🔹 MSE: 11.97
🔹 R² Score: 0.1736
```

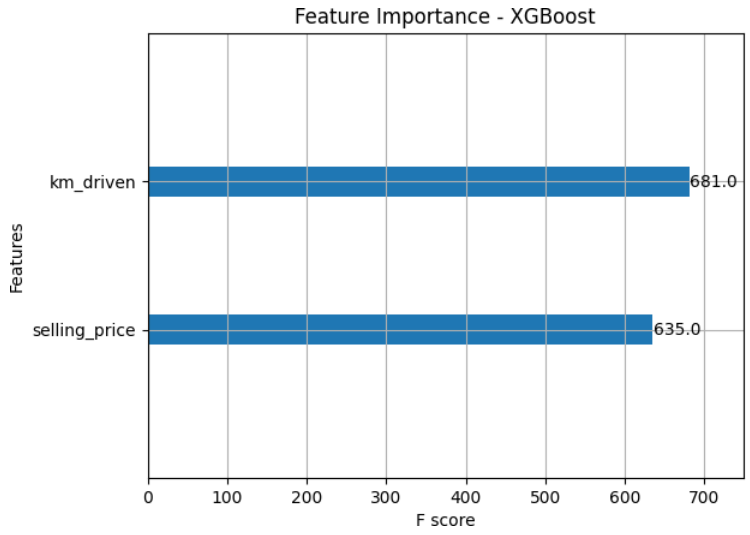
```
import xgboost as xgb

features = ['selling_price', 'km_driven']

target = 'year'

df = df[features + [target]].dropna()


X = df[features]

y = df[target]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100, learning_rate=0.1, max_depth=4)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)


mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)


print(f"🔥 XGBoost Results:")
```

```
pri nt( f"■MSE: {mse:.2f}")
p r i n t ( f " ■R² Score: {r2:.4f}")
xgb.plot_importance(model)
plt.title("Feature Importance - XGBoost")
plt.show()
```

⚡XGBoost Results:

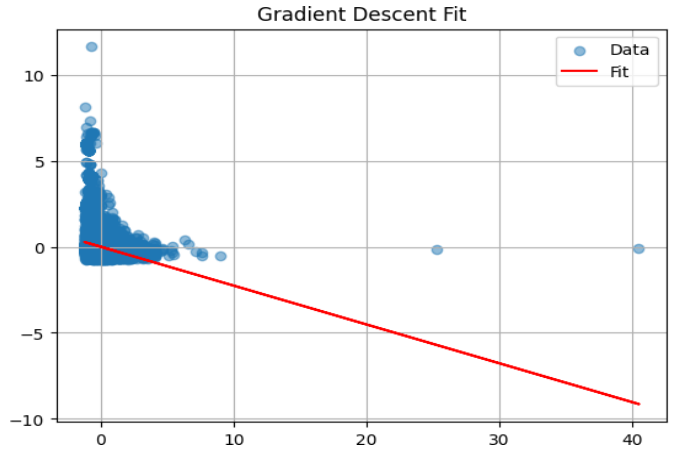
■ MSE: 4.52

■ R² Score: 0.6881




```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv('/content/Car details v3.csv')[['selling_price', 'km_driven']].dropna()
X = df['km_driven'].values
y = df['selling_price'].values
X = (X - X.mean()) / X.std()
y = (y - y.mean()) / y.std()
m, b = 0, 0
lr = 0.01
for _ in range(1000):
    y_pred = m * X + b
    error = y - y_pred
    m -= lr * (-2 * (X * error).mean())
```

```
    b -= lr * (-2 * error.mean())


print(f"y = {m:.4f}x + {b:.4f}")


plt.scatter(X, y, alpha=0.5, label='Data')
plt.plot(X, m*X + b, color='red', label='Fit')
plt.title("Gradient Descent Fit")
plt.legend()
plt.grid(True)
plt.show()
```




```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler
df = pd.read_csv('/content/Car details v3.csv')
df = df[['selling_price', 'km_driven', 'year']].dropna()


X = df[['km_driven', 'year']]
y = df['selling_price']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
model = LinearRegression()
scores = cross_val_score(model, X_scaled, y, cv=5, scoring='r2')
print("Cross-Validation R² Scores:", scores)
```

print(f"🔹 Average R² Score: {scores.mean():.4f}")

```
Cross-Validation R² Scores: [0.16599062 0.17625213 0.17522591 0.16039516 0.17537407]
🔹 Average R² Score: 0.1706
```

plt.figure(figsize=(10, 6))

plt.plot(y_test.values[:50], label='Actual', marker='o')

plt.plot(y_pred[:50], label='Predicted', marker='x')

plt.title("🔍 Actual vs Predicted Selling Price")

plt.xlabel("Sample Index")

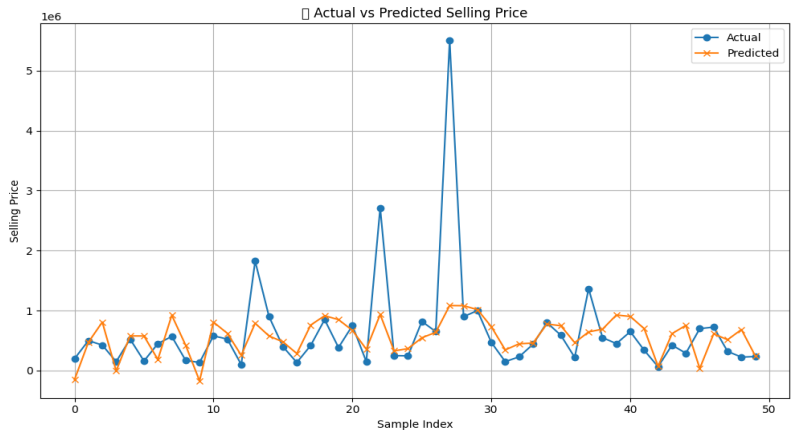
plt.ylabel("Selling Price")

plt.legend()

plt.grid(True)

plt.tight_layout()

plt.show()



Github: https://github.com/KumarAryanSingh-05/car-price-prediction.git