USED CAR PRICE PREDICTION

A project report submitted in fulfillment for the
Diploma Degree in AI & ML
under Applied Roots and University of Hyderabad

Submitted by
Dhruv Bhatia (40AIML149-21/1)

UNIVERSITY OF HYDERABAD

Declaration of Authorship

We hereby declare that this thesis titled" Used Car Price Prediction" and the work presented by the undersigned candidate, as part of a Diploma Degree in AI & ML.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: Dhruv Bhatia
Thesis title: Used Car Price Prediction

Abstract

The primary objective of this project is to estimate the value of used cars by using different attributes of the car which are highly correlated to the final dependent variable i.e. the car resale value. This is a supervised learning regression problem where the continuous variable of car resale value is predicted using different machine learning models. The models are trained and tested against a benchmark dataset from Car Dekho and all the models are compared by a performance metric of R-squared value. Feature Engineering and hyper-parameter tuning is also used to increase the performance of the models and finally, the model is deployed on the local machine with the user interface which can be used for integration with the mobile app or a website for the general public to use.

Acknowledgements

Every project big or small is successful largely due to the effort of a number of wonderful people who have always given their valuable advice or lent a helping hand. I sincerely appreciate the inspiration; support, and guidance of all those people who have been instrumental in making this project a success. I, Dhruv Bhatia student of applied roots, am extremely grateful to the mentors for the confidence bestowed on me and for entrusting my project entitled “Used Car Price Prediction” with special reference.

At this juncture, I express my sincere thanks to the Mentors of applied roots for making the resources available at right time and providing valuable insights leading to the successful completion of our project who even assisted me in completing the project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Used cars play an important role in the automotive industry, offering an affordable alternative to purchasing a brand-new vehicle. They have become increasingly popular due to their affordability and accessibility to a wider range of consumers. The demand for used cars has grown in recent years due to various reasons such as the rising costs of new cars, the ability to purchase more advanced and reliable models, and a wider range of choices. Additionally, used cars are also eco-friendly as their production has already been completed, and they can be recycled or repurposed at the end of their life. Used cars have become an integral part of the automotive industry, offering a viable option for many consumers to own a reliable and affordable vehicle.

The used car market in India has been growing rapidly over the past few years, with more and more consumers opting for used cars as a cost-effective alternative to buying new ones. The demand for used cars in India has been driven by a variety of factors, including the rising costs of new cars, increasing disposable incomes, and the availability of financing options.

The used car market in India is highly fragmented, with a large number of organized and unorganized players operating in the market. There are several online platforms such as CarDekho, CarTrade, and OLX that have emerged as popular options for buying and selling used cars in India. In addition, there are also a large number of local dealerships and independent sellers offering a range of used cars to consumers.

One of the main advantages of buying a used car in India is the affordability factor. The cost of a used car is significantly lower than that of a new car, making it a more accessible option for many consumers. Additionally, used cars also come with a lower depreciation rate, meaning that their value does not decrease as rapidly as new cars. Overall, the used car market in India is an important sector of the automotive industry, offering a cost-effective and accessible option for many consumers to own a vehicle.

1.1 Problem Statement

In this project we need to predict the price of used cars. Using the data given, we must ensure that both the buyer as well the seller gets a value that true based on the conditions of the car and the market situation.

1.2 Business/Real-world impact of solving this problem

Used car price prediction can have a significant impact on various stakeholders, such as buyers, sellers, and financial institutions. For buyers, knowing the predicted price of a used car can help them make informed decisions when purchasing a vehicle. They can use the information to negotiate the price with the seller and ensure that they are paying a fair price for the vehicle. Additionally, they can use the information to budget and plan for their purchase, avoiding any unexpected costs. For sellers, having a price prediction can help them determine the optimal price for their vehicle. They can use the information to price the car competitively and attract more buyers. Additionally, they can use the prediction to negotiate the price with potential buyers, ensuring that they are getting the best deal possible. For financial institutions, used car price prediction can help them assess the value of vehicles for loan and insurance purposes. They can use the information to determine the amount of the loan or insurance coverage to offer, minimizing their risk and ensuring that they are making sound financial decisions. Overall, used car price prediction can have a significant impact on the decisions and outcomes of various stakeholders involved in the used car market.

1.3 Challenges

Predicting the price of a used car can be challenging due to several factors, including

- Vehicle Condition:
  The condition of the car is a critical factor in determining its value. A well-maintained vehicle with low mileage will typically have a higher value than a car with high mileage and wear and tear.
- Market fluctuations:
  The used car market can be volatile, and prices can fluctuate based on supply and demand factors, seasonal trends, and other external factors like economic conditions, fuel prices, and industry trends.
- Geographical factors:
  The prices of used cars can vary significantly depending on the location, such as urban or rural areas, regional demand and supply, and location-specific factors like weather and road conditions.
- Brand and Model:
  Different car models and brands have their own unique value propositions, which can significantly impact the final selling price.
- Personal preferences:
  Buyers' individual preferences and biases can also influence the value of a used car.

To accurately predict used car prices, a comprehensive approach that accounts for these factors and utilizes advanced data analytics, machine learning, and AI algorithms may be required. Additionally, incorporating expert knowledge from the automotive industry and other relevant fields can further enhance the accuracy of used car price prediction.

1.4 Key Metrics to Optimize

When it comes to regression, the performance metrics typically used to evaluate the accuracy of a predictive model are:

- Mean Absolute Error (MAE):
  MAE measures the average absolute difference between the predicted and actual values. A lower MAE indicates better accuracy.
- Mean Squared Error (MSE):
  MSE measures the average of the squared differences between the predicted and actual values. This metric gives higher weight to larger errors and is useful for identifying outliers in the data.
- Root Mean Squared Error (RMSE):
  RMSE is the square root of the MSE, and it is also a popular metric for evaluating the accuracy of regression models. Like MSE, RMSE gives higher weight to larger errors.
- R-Squared (R2):
  R-Squared measures the proportion of variance in the dependent variable that is predictable from the independent variable(s). It ranges from 0 to 1, with higher values indicating better predictive power.
- Adjusted R-Squared:
  This metric is similar to R-squared but adjusts for the number of variables in the model. It is useful when comparing models with different numbers of variables.
- Mean Absolute Percentage Error (MAPE):
  MAPE measures the average percentage difference between the predicted and actual values. It is commonly used in business forecasting applications.

In this project, we will be using the R-squared value to measure the performance of the models. The implementation of the metric is also applied from scratch i.e. using any libraries.

Chapter 2

Literature Review

In the paper, [1] ”Price Prediction of Used Cars Using Machine Learning” by C. Jin explores the use of machine learning algorithms to predict used car prices. The author discusses the challenges of predicting used car prices, including the variability of car conditions and the influence of market trends. Several machine learning algorithms, including linear regression, decision trees, and random forests, are applied to a dataset of used car sales to predict prices. The results show that random forest is the most accurate algorithm for predicting used car prices. The paper concludes by discussing potential future research directions, such as incorporating deep learning techniques and expanding the scope of the study to include additional factors that influence used car prices.
The authors in paper [2] have proposed a model based on the k-nearest neighbors (KNN) algorithm for predicting used car prices. The authors collected data on various features of used cars, including make, model, year, mileage, fuel type, and location. They applied the KNN algorithm to this dataset to predict the prices of used cars. The results of the study showed that the KNN algorithm can provide a reasonably accurate prediction of used car prices. The paper concludes by suggesting that this model can be useful for buyers and sellers in the used car market to make informed decisions about pricing.

References

[1] C. Jin, “Price prediction of used cars using machine learning,” in *2021 IEEE International Conference on Emergency Science and Information Technology (ICESIT)*, pp. 223–230, 2021.

[2] K. Samruddhi and R. Kumar, “Used car price prediction using k-nearest neighbor based model,” *International Journal of Innovative Research in Applied Sciences and Engineering*, vol. 4, pp. 686–689, 09 2020.

Chapter 3

Data Description

3.1 Car Price Data

The data set consists of typical features of cars in the Indian market, which has been extracted from the Car Dekho official website. The provided dataset corresponds to car features in the market. The dataset has 20,026 observations and 16 features.

3.2 Data Acquisition

Open source data – The dataset for this project is retrieved from Kaggle, the home of Data Science.

- Data Size: 6.7 MB
- Data Shape: 20,026 rows and 16 columns
- No challenges in processing the data, used pandas to read the data

3.3 Tools

The below mentioned tools are used in this project:

- Pandas
- Numpy
- Scikitlearn
- Matplotlib
- Seaborn
- Jupyter Notebook

- Flask (for Deployement)
- HTLM
- Spyder

3.4 Dataset Level

The data set has 20,026 rows and 16 columns, where 13 columns describe the features of the old cars sold and 3 columns relate to the web scraping process. Table 3.1. below gives a basic description of all the columns used in the dataset.

TABLE 3.1: Dataset Feature Analysis.

Feature Name	Description	Feature Type	Null Values	% of Null Values w.r.t. Total
Source.Name	Web-scraping process	-	N/A	-
web-scraper-Order	Web-scraping process	-	N/A	N/A
web-scraper-start-url	Web-scraping process	-	N/A	N/A
full_name	Full name of the cars with company name and model name	Descriptive	46	0.23%
selling_price	The price in INR at which the car is sold	Continuous	46	0.23%
new-price	On-road price of the new car from showroom in INR	Continuous	10,460	52.20%
Year	Manufacturing year of the car	Discrete	46	0.23%
seller_type	Describes the type of seller (Individual, Dealer etc.)	Categorical	46	0.23%
km_driven	Total km driven by the car	Continuous	46	0.23%
owner_type	Describes the type of owner (first, second etc.)	Categorical	46	0.23%
fuel_driven	Describes the fuel wise type of car (Petrol, Diesel etc.)	Categorical	46	0.23%
transmission_type	Whether manual or automatic	Categorical	46	0.23%
Mileage	Total performance mileage of the car in kilometre per litre(kmpl)	Continuous	46	0.23%
Engine	Cubic capacity of engine	Continuous	105	0.52%
max_power	Maximum power capacity of the car in BHP	Continuous	105	0.52%
Seats	Number of seats in the car	Discrete	173	0.86%

3.5 Output Variable Analysis

Selling price i.e. the price in INR that the car is sold, is the output or the target variable in the project. It is also the dependent variable for the prediction model. The variable is of continuous type with 46 null values which is 0.23% of the total values. The variable consists of unit of amount in text with the numerical values. The units can be divided into 3 categories:

1. Lakh*: example from raw data-set - 1.2 Lakh*, 5.5 Lakh* etc.
2. Crore*: example from raw data-set – 1.1 Cr*, 1.3 Cr8* etc.
3. Thousands*: example from raw data-set – 98,000*, 95,000* etc.

It can be seen from the above analysis that the data variable consisted of textual data, which also impacted the overall values of the numerical data (for example, numerical values with 'Lakh*' should be multiplied by 100000 and so on for 'Cr*'. Hence the following process was involved in cleaning the data:

1. Splitting text and numerical values.
2. Removing '*' and ',' in the data.
3. Multiplying the numeric values based on the text units obtained from splitting in point 1.

Table 3.2 below depicts the aggregate values after cleaning the variable.

Table 3.2: Output Variable Numerical Analysis.

Measure	**Value**
Count	19,980
Mean	7,39,206
Std. Dev.	9,10,308
Min.	25,000
1st Quartile	3,40,000
Median	5,20,000
3rd Quartile	7,85,000
Max.	3,95,00,000

Fig. 3.1 shows the probability density function plot of the variable selling price which is plotted on a logarithmic scale of 10. The variable was rightly skewed as can be seen from the table above. Hence the plot using normal scale would not provide any valuable insights on the data. The following observations can be made from the plot:

1. The plot on the logarithmic scale resembles a normal distribution.
2. Major distribution falls between the 1st Quartile i.e. 3.85 lakh, and the 3rd Quartile i.e. 7.85 Lak.

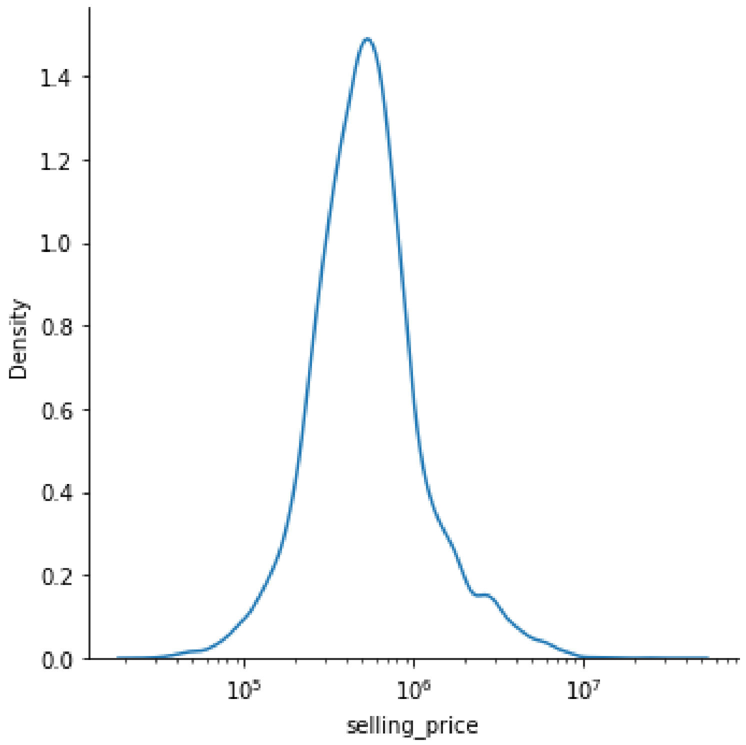


Figure 3.1: Distribution Plot – Selling Price (in INR)

Chapter 4

Data Pre-processing and Exploratory Data Analysis

4.1 Data Cleaning and Univariate Data Analysis

There are 3 columns in the data-set i.e. Source.Name, web-scraper-order, and web-scraper-start-url. These columns are related to the web scrapping process and do not provide any valuable insights about the features of the cars. Hence the above three mentioned features are removed from the dataset.

- **Feature 1: full_name:**
  This feature shows the full name of the car and it is divided into the Company name and the car Model name. The first word of this feature describes the company or the manufacturer name, and the second word describes the car model name. Hence, after analyzing the data 2 additional features can be created i.e. Model name and Company name, and the earlier feature full_name can be removed. We are creating the 2 additional features as this removes the redundant data from the full_name column and gives only valuable data i.e. Company and Model name.
  The following is the list of all 42 unique company names: ['MARUTI' 'HYUNDAI' 'FORD' 'MAHINDRA' 'TATA' 'RENAULT' 'NISSAN' 'MINI' 'MERCEDES-BENZ' 'TOYOTA' 'FIAT' 'VOLKSWAGEN' 'HONDA' 'CHEVROLET' 'AMBASSADOR' 'DATSUN' 'KIA' 'BMW' 'MITSUBISHI' 'AUDI' 'SKODA' 'LAND' 'JAGUAR' 'DAEWOO' 'BENTLEY' 'MG' 'ISUZU' 'PORSCHE' 'VOLVO' 'LEXUS' 'JEEP' 'PREMIER' 'MASERATI' 'FORCE' 'LAMBORGHINI' 'FERRARI' 'OPELCORSA' 'MERCEDES-AMG' 'DC' 'ROLLS-ROYCE' 'OPEL']

  Similarly, there are 274 car names in the data-set for all the above-mentioned companies. Fig. 4.1 shows the plot of company names and the total number of cars sold for each company.

  Fig. 4.2 describes the most number of model names in the form of the word cloud. Here since the unique car names are 274, which is very high, therefore we are using word cloud and not using a typical histogram as used for the company name feature.
  Since both the attributes Model name and Company name have large categories and do not provide any valuable information about the condition of the car, therefore, we are removing both columns.

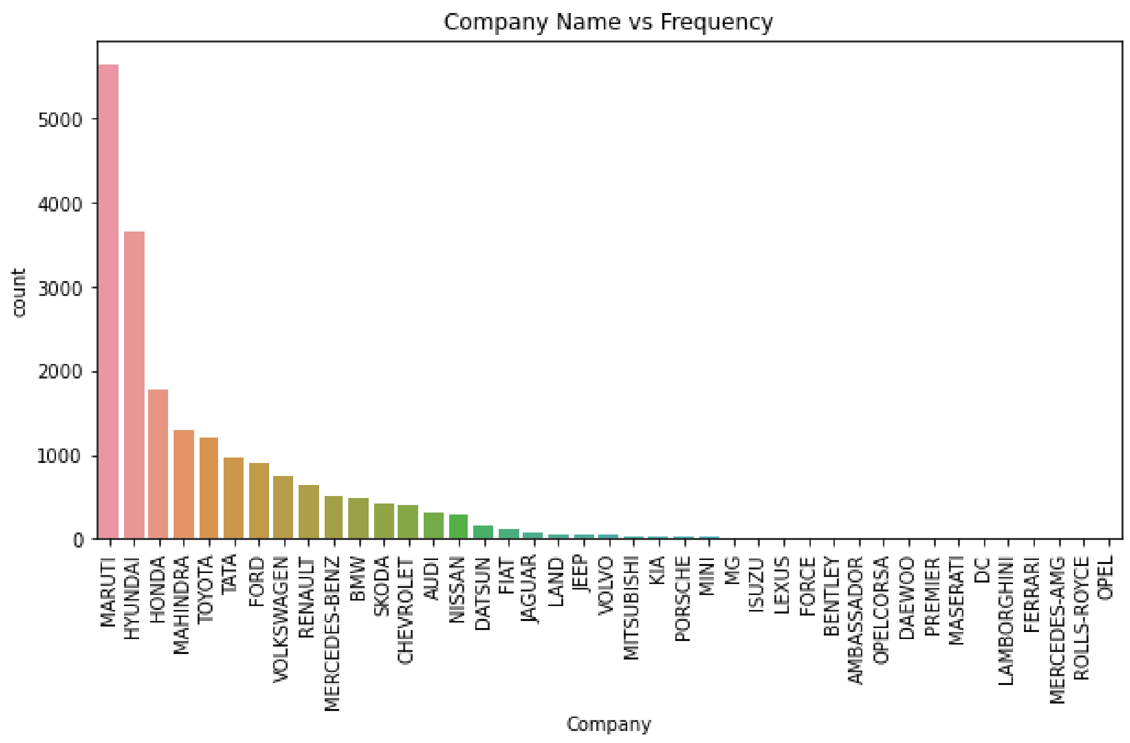


FIGURE 4.1: Company vs Frequency Plot

GRAND'
CITY'
ALTO'
XUV500'
RAPID'
I20'

FIGURE 4.2: Major Car Model Names

- **Feature 2: km_driven:**
The km_driven feature describes the total km driven by a car. The variable is of continuous type with 46 null values, which is 0.23% of the total values. The data consists of the numerical values with the text notation 'km', which had to be removed. Also, ',' is removed from the values, the text is also removed and the column is converted to numerical type. Table 5.12 depicts the aggregate values after cleaning the variable.

TABLE 4.1: Km_driven Variable Analysis.

Measure	**Value**
Count	19,980
Mean	58,244
Std. Dev.	51,725
Min.	100
1st Quartile	31,164
Median	52,000
3rd Quartile	74,000
Max.	38,00,000

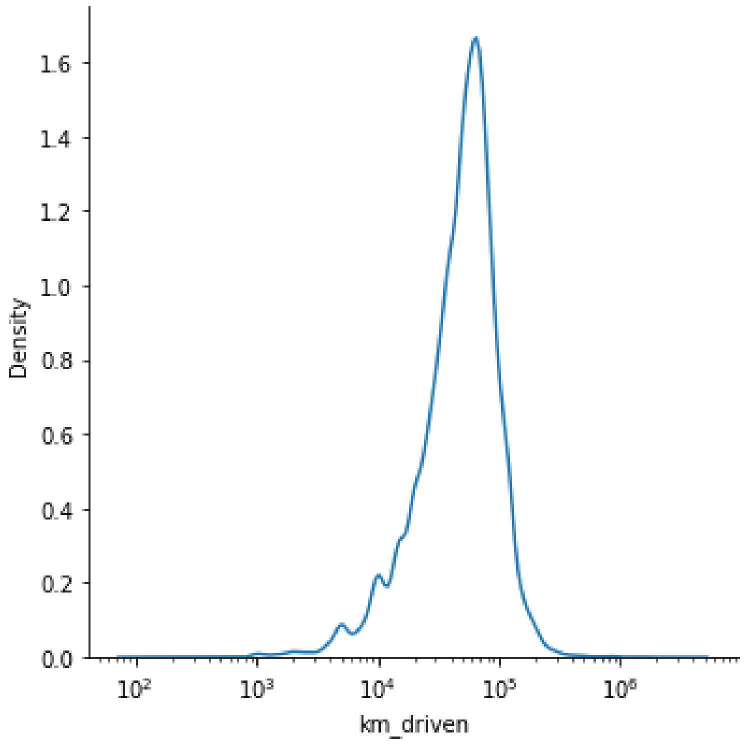


FIGURE 4.3: Distribution Plot – km_driven(kms)

Fig. 4.3 shows the probability density function plot of the variable kms driven, which is plotted on a logarithmic scale of 10. The variable was rightly skewed, as can be seen from the table above. Hence the plot using a normal scale would not provide any valuable insights into the data. The following observations can be made from the plot:
1. The plot on the logarithmic scale resembles a normal distribution.

2. Major distribution falls between the 1st Quartile i.e. 31,164 kms and the 3rd Quartile i.e. 74,000 kms.

- **Feature 3: mileage:**
  Mileage column has units 'kmpl', 'km/kg', and 'km/hr'. Out of these clearly, km/hr does not represent mileage but speed hence removed these values. Also removed text from the column, and few values of more than 100 were observed, which is not correct for mileage, hence removed those values also.Fig.4.4 shows the box plot of the variable mileage. The mean value is 19.2 kmpl with a minimum of 0 kmpl and a maximum of 33.5 kmpl. Also, the standard deviation of the data is 4.4 kmpl.

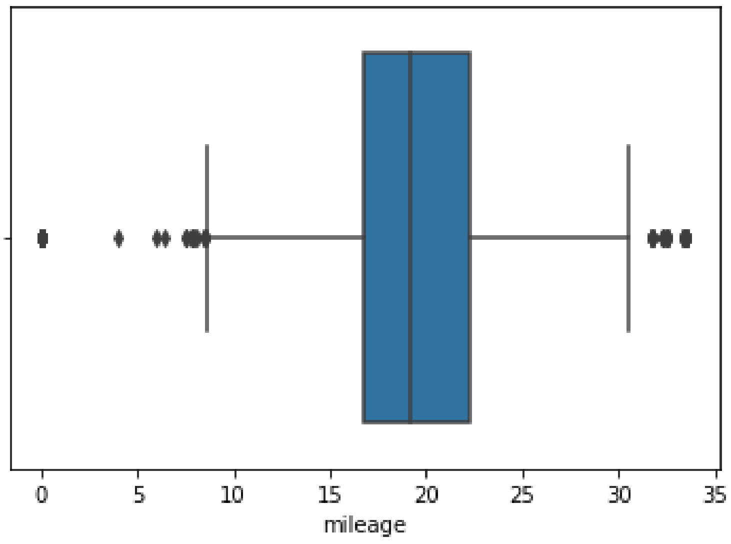


FIGURE 4.4: Box-plot – mileage (kmpl)

- **Feature 4: engine:**
  The engine column describes the engine capacity of the cars in unit CC. After analysing the column, it had text data which is removed and also some rows had entries like 'wheel size' which is clearly a data entry error, hence that also had to be removed. Fig. 4.5, 4.6 show the box-plot and pdf plot of the variable engine. The mean value is 1477.7 CC with a minimum of 0 CC and a maximum of 6752 CC. Also, the standard deviation of the data is 520.0 CC.

- **Feature 5: max_power:**
  The maximum power column also had entries in Brake horsepower (bhp) with the text. The text is removed and a few data entry errors are also removed and the column is also converted to a numerical feature. Fig. 4.7 shows the box-plot of the variable max_power. Intuitionally the maximum power and engine capacity of a car would be directly dependent on the number of seats. For example, all 5-seater cars would have similar max power and engine capacity, and all 10-seater cars will have a similar higher max power and engine capacity. Therefore, replacing the null values in the max power and engine column with average values based on the group by seat columns.

- **Feature 6: seats**
  The seat column is a categorical variable that describes the number of seats in a

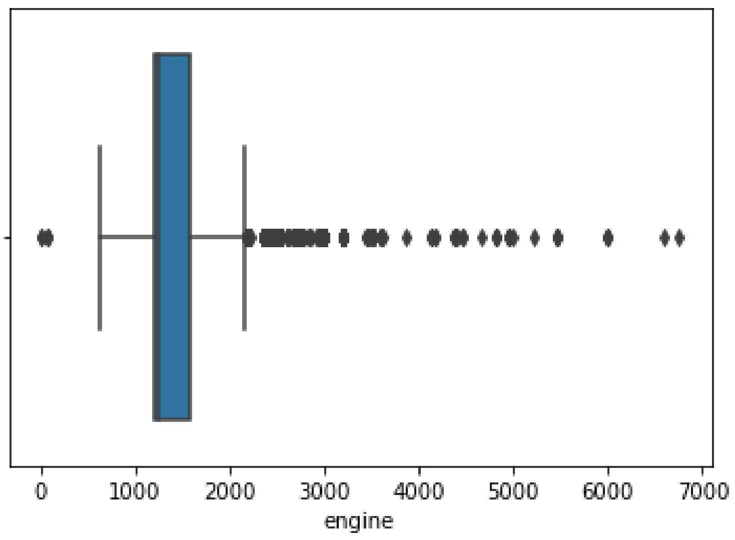


FIGURE 4.5: Box-plot-engine (CC)

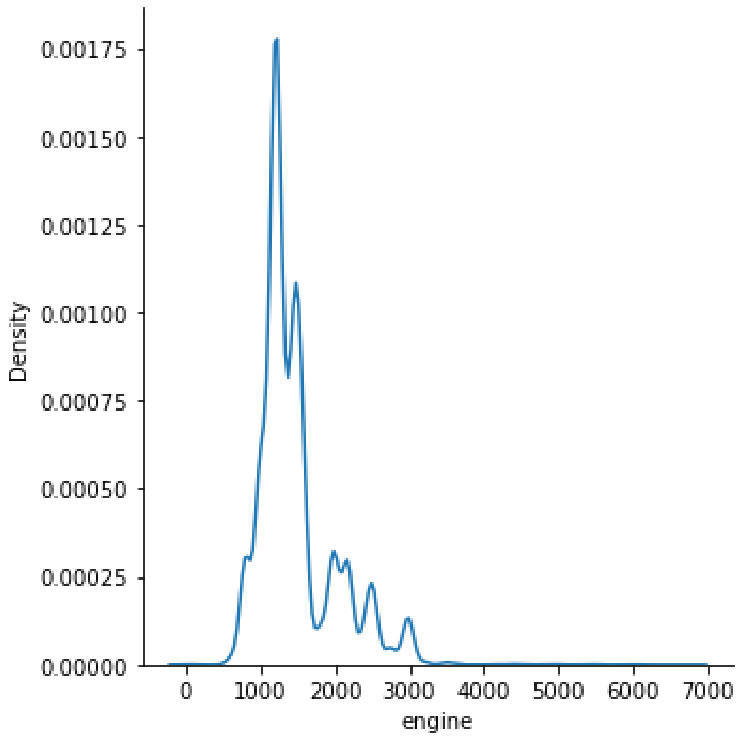


FIGURE 4.6: Distribution Plot–Engine capacity (CC)

car. The column contains a number of seats in which the text "SEATS," so this text has to be removed. Apart from text removal, a few rows had N/A and Null values, which also had to be converted to NaN.

Fig. 4.8 shows the frequency plot for the seat column. Below are the observations from the plot:

1. There are cars from 2 seats to 14 seats in the dataset.
2. The highest number of cars has 5 seats with 16350 cars.

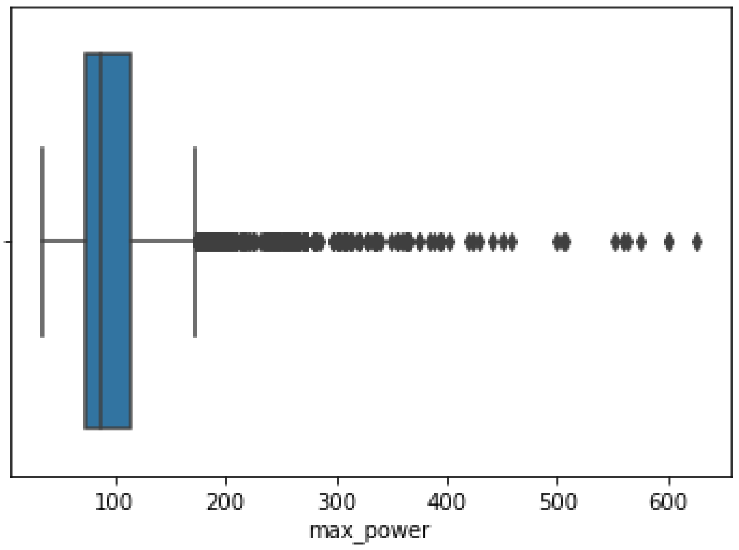


FIGURE 4.7: Box-plot – Maximum Power (bhp)

3. There is also a high frequency of 7 seater cars.

Since 83% of the cars in the dataset have 5 seats, hence replacing the null values in the seat column with mode i.e. 5 seats.

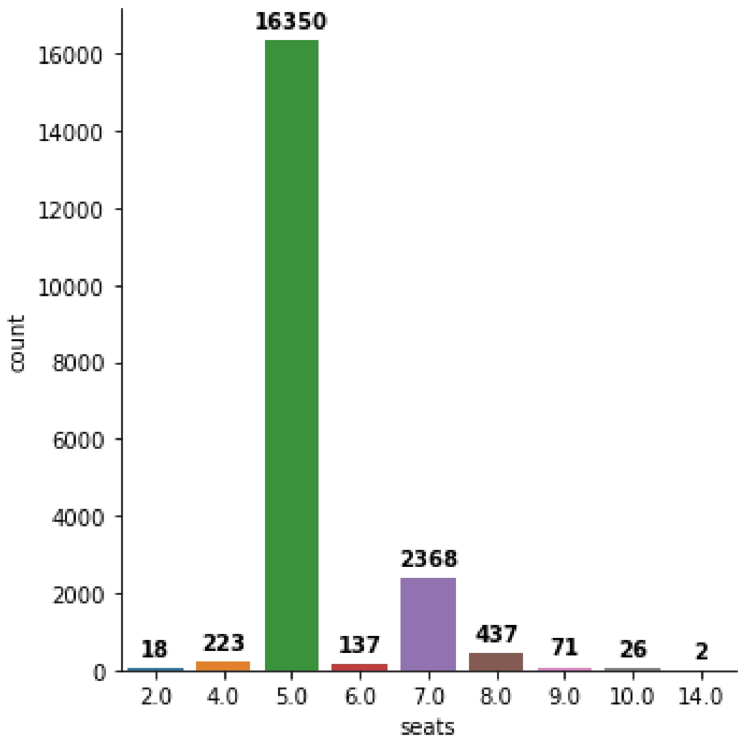


FIGURE 4.8: Seats vs Frequency Plot

- **Feature 7: new-price**
  Dropping new-price column since it has more than 50% null values and does not provide any valuable information in the dataset.

- **Feature 8: year**
  The year column consists of the year the cars were manufactured in. This can also describe the age of the car if it is subtracted from the current year. Fig. 4.9 shows the frequency plot for the year column.

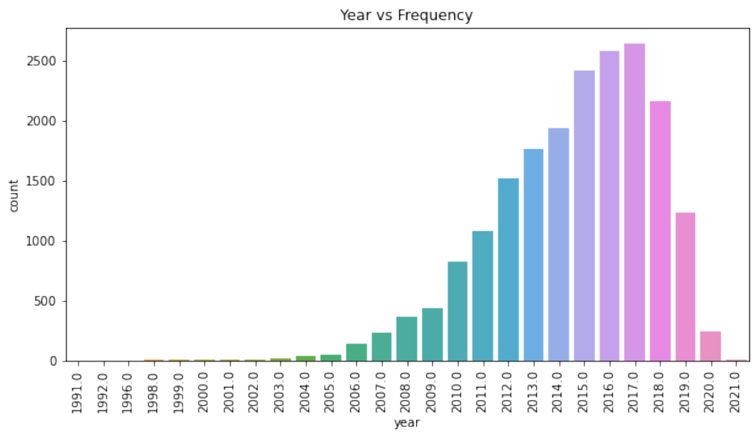


FIGURE 4.9: Year vs Frequency Plot

- **Feature 9: seller_type**
  The seller type column has 2 major entries with 'Individual' and 'Dealer' based cars. The third category, 'Trustmark dealer,' has only 190 entries which are negligible. Fig. 4.10 gives the distribution of data based on the above-mentioned three categories.

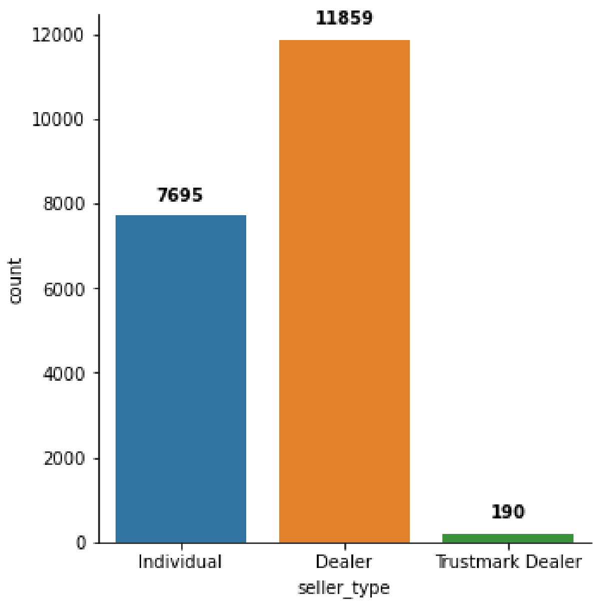


FIGURE 4.10: Seller Type vs Frequency Plot

- **Feature 10: owner_type**
  The owner type column describes the owner of the car, whether it is the first owner, second owner, or third owner. More than 99% of the data has first owner seller type. Only 6 rows have second-owner or third-owner entries. Hence these can be considered outliers. Therefore dropping the owner type column.

- **Feature 11: transmission_type**
  The transmission type column has 2 entries with manual or automatic cars.

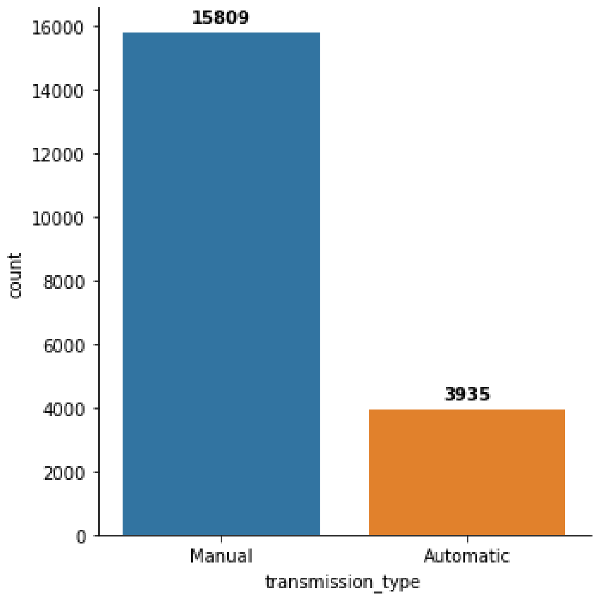


FIGURE 4.11: Transmission Type vs Frequency Plot

- **Feature 12: fuel_type**
  The fuel type column has 5 entry types i.e Petrol, Diesel, CNG, Electric, and LPG type cars. Here the majority of cars are in Petrol and Diesel full category. With a marginal number of entries for CNG, LPG, and Electric cars.

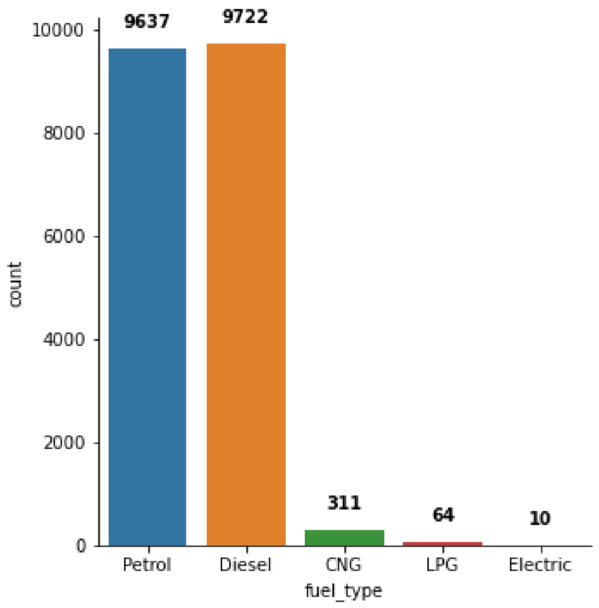


FIGURE 4.12: Fuel Type vs Frequency Plot

4.2 Multivariate Analysis

4.2.1 Correlation Matrix

The correlation heat map is a graphical representation of the correlation matrix. It can also be defined as the measure of dependence between two different variables. The value of the correlation coefficient can take any value from -1 to 1.

- If the value is 1, it is said to be a positive correlation between two variables. This means that when one variable increases, the other variable also increases.
- If the value is -1, it is said to be a negative correlation between two variables. This means that when one variable increases, the other variable decreases.
- If the value is 0, there is no correlation between the two variables. This means that the variables change in a random manner with respect to each other.

Fig. 4.13 describes the correlation matric heat map of the dataset.

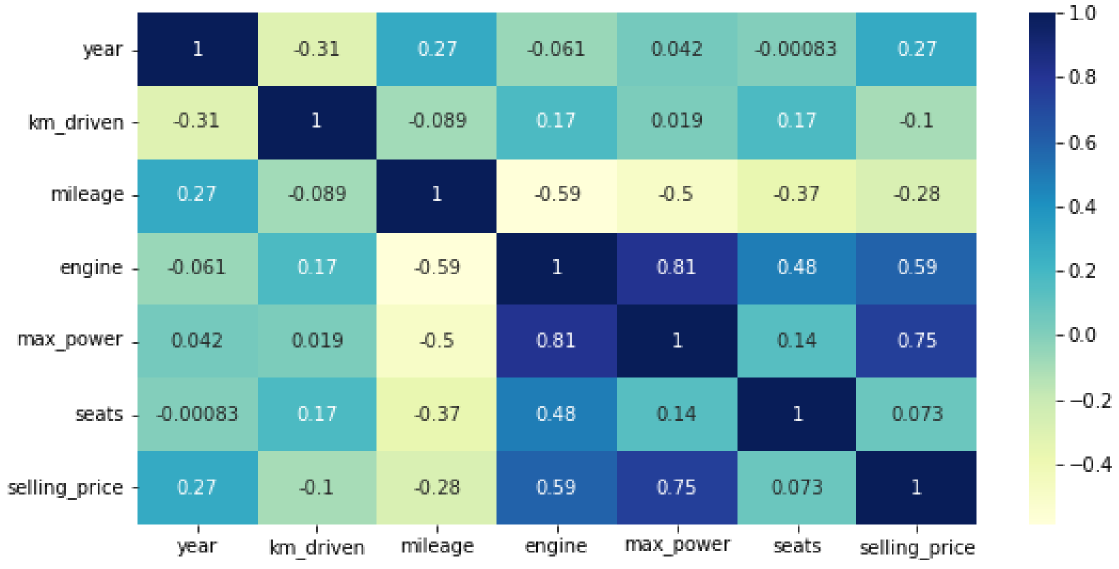


FIGURE 4.13: Correlation Matrix

The following observations can be made from the above matrix:

- The variable mileage has negative coefficient of correlation with every other variable except year column.
- Maximum power and engine columns are highly positively correlated i.e. when one variable increases the other variable also increases.
- Maximum power and selling price variable are also highly correlated i.e. higher the maximum power higher will be the selling price of the car.

4.2.2 Dependency of All Variables with Target Variable

- **Selling Price vs Year**

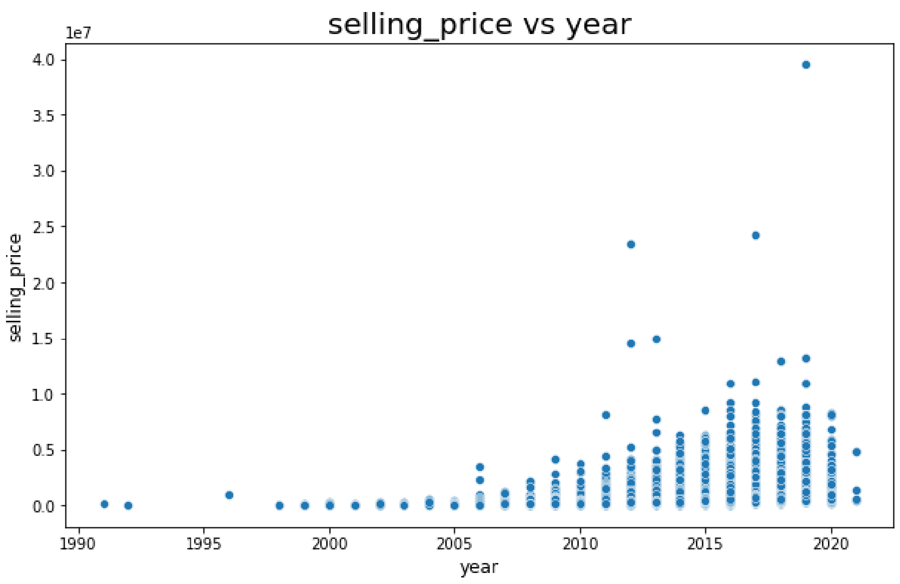


FIGURE 4.14: Scatter Plot- Selling Price vs Year

Observation: It is observed that the cars manufactured in recent years have a higher selling price than the older cars. Here selling price is of continuous type and the year is of the discrete type, that is why we are using a scatter plot for the visualization.

- **Selling Price vs Mileage**

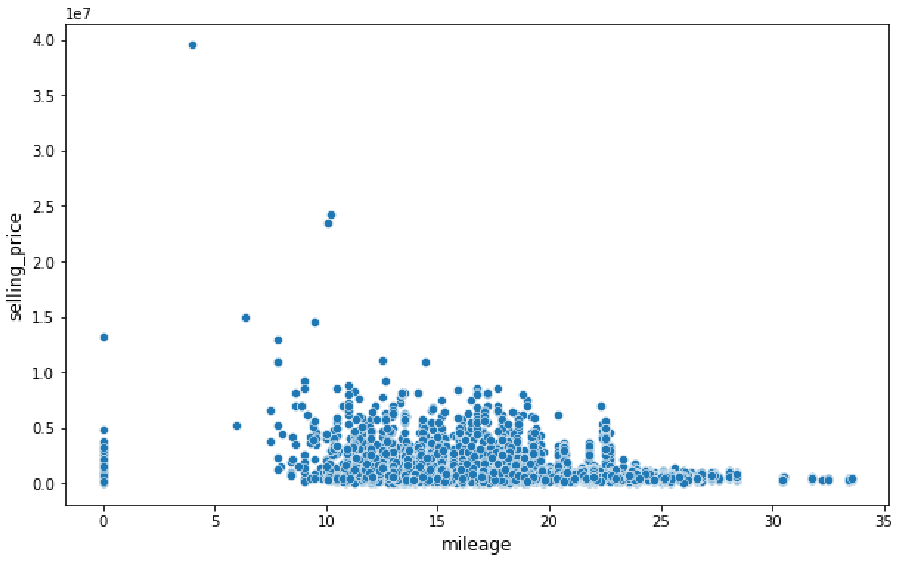


FIGURE 4.15: Scatter Plot- Selling Price vs Mileage

Observation: Higher the mileage lower is the selling price. It can be seen from the plot that mileage with 20 kmpl and greater has a lower selling price as compared to mileage with less than 20 kmpl. This can also be seen from the negative coefficient of correlation in the heat map matrix above.

- **Selling Price vs Engine**

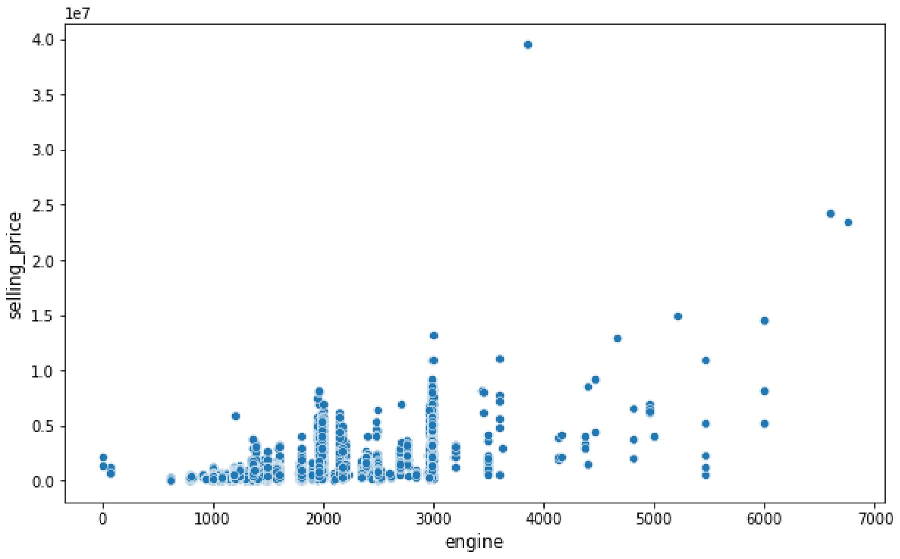


FIGURE 4.16: Scatter Plot- Selling Price vs Engine

Observation: It can be observed that the higher the engine capacity higher would be the selling price. It can also be validated with a correlation coefficient of 0.59.

- **Selling Price vs Maximum Power**

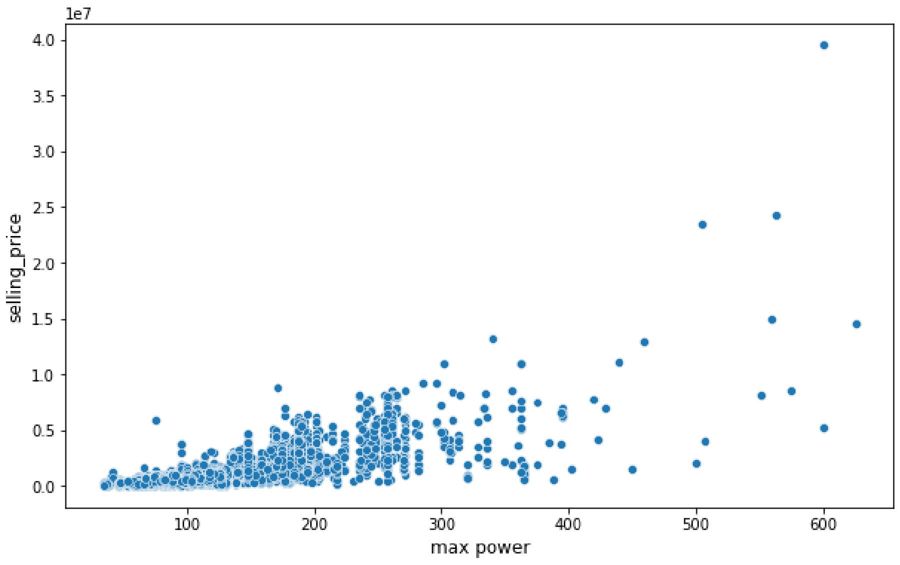


FIGURE 4.17: Scatter Plot- Selling Price vs Max. Power

Observation: With a high correlation of 0.75 there is also a positive relation between the above two variables.

- **Selling Price vs Seller Type**

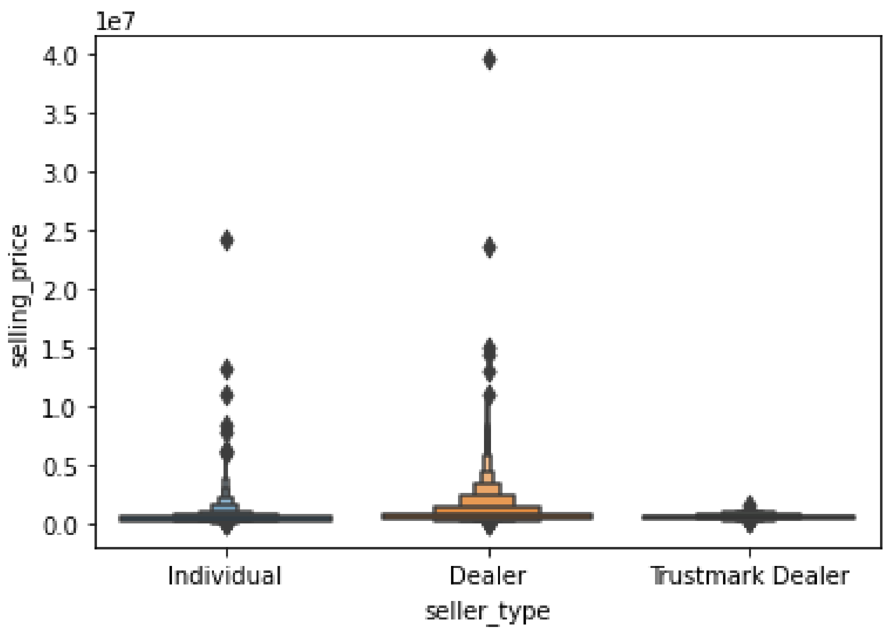


FIGURE 4.18: Box Plot- Selling Price vs Seller Type

Here we are using the box plot method since one variable is categorical and one is continuous and it can be clearly observed that the Dealer seller type has a higher aggregate selling price than the Individual and Trustmark Dealers seller types.

- **Selling Price vs Transmission Type**

Observation: Automatic type cars have a higher selling price than Manual type cars.

4.3 Encoding Techniques Used

The Table 4.2 describes the categorical features in the dataset with the number of categories in each feature.

TABLE 4.2: Categorical features.

Feature	**No. of categories**
Seller_type	3
Fuel_type	5
Transmission_type	2

Since the number of categories is less in seller_type, fuel_type and transmission_type features, one hot encoding can be used for this. One Hot Encoding:

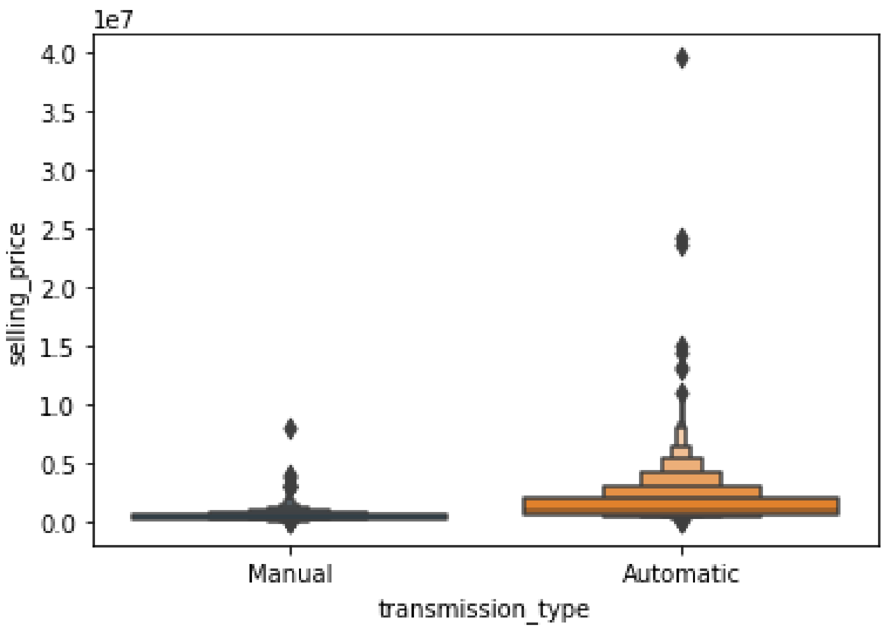
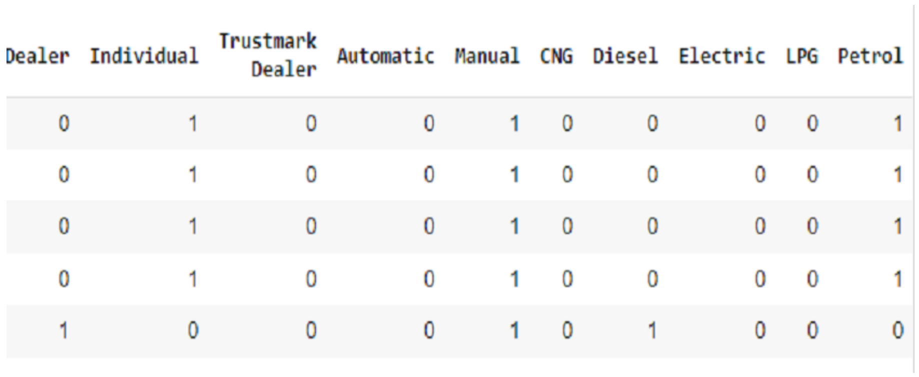


FIGURE 4.19: Box Plot- Selling Price vs Transmission Type

- Can be directly fed to the machine learning mathematical model.
- Can be used when the categories are not ordinal.
- Is used when the number of categories are less.

Since seller_type, fuel_type, and transmission_type variables are not ordinal and the number of categories is also less therefore, we are using one-hot encoding for the process. The below Fig. 4.20 describes the features obtained after one hot encoding.



Dealer	Individual	Trustmark Dealer	Automatic	Manual	CNG	Diesel	Electric	LPG	Petrol
0	1	0	0	1	0	0	0	0	1
0	1	0	0	1	0	0	0	0	1
0	1	0	0	1	0	0	0	0	1
0	1	0	0	1	0	0	0	0	1
1	0	0	0	1	0	1	0	0	0

FIGURE 4.20: Features after One Hot Encoding

Chapter 5

Predictive Modelling

Before the models are trained, data preprocessing is carried out. The full name column which was divided into car name and company name is taken. Car name has 274 unique names which are very high and therefore it is removed. For the company name column, since there are 42 unique categorical values, therefore it is also removed. Also, for the discrete and continuous features, standardization is carried out using StandardScaler function. Standardization is done to make the data internally consistent. Since the range is different for all continuous columns example. Km_driven has a range of 0 to more than 10000 kms whereas the mileage column has a range of 0 to 35 kmpl, hence this would create a problem when applied to the models. Models like K-NN, Lasso, Ridge, and Linear Regression are sensitive to standardization.

5.1 Base-line Model and Metrics

The baseline model used for the problem is Linear Regression and the metric used in the R-squared value. Here linear regression is used since it is the most basic regression model and has less complexity as compared to other models. R-squared value or also known as the coefficient of determination is the proportion of the variation in the dependent variable that is predictable from the independent variable. Below is the mathematical notation for the same.

$$R^2 = 1 - \frac{RSS}{TSS} \tag{5.1}$$

Here, RSS: sum of squares of residuals
TSS: the total sum of squares
The code for the metric R-squared value from scratch.

```
def r2_value(y,y_pred):
    y_mean=np.mean(y)
    ss_reg=np.sum(np.square(np.subtract(y,y_pred)))
    ss_tot=np.sum(np.square(np.subtract(y,y_mean)))
    R_square=1-(ss_reg/ss_tot)
    return R_square
```

5.2 Cross Validation

Cross-validation is a statistical method used to estimate the performance (or accuracy) of machine learning models. It is used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited. In cross-validation, you make a fixed number of folds (or partitions) of the data, run the analysis on each fold, and then average the overall error estimate.

5.2.1 Types of Cross-Validation:

1. Holdout Method
2. K fold cross-validation
3. Stratified K fold cross-validation
4. Leave-one-out cross-validation

Out of the above cross-validation techniques K fold cross-validation is used because of the following advantages:
1. Every data point gets to be tested exactly once and is used in training k-1 times.
2. The variance of the resulting estimate is reduced as k increases.
3. Reduced bias.

The main disadvantage of k-fold cross validation is that the value of k increases the training time of the model also increases as the model has to run k times. Below is the code from scratch for k fold cross validation with random search cv for hyperparameter tuning.

```
%%time
import random
from random import sample
def RandomSearchCV(X_train,Y_train,regressor, param_range, folds):

    train_scores=[]
    cv_scores=[]

    #Creating 10 random integers from the range give in variable 'param_range'.
    para=[]
    r1=param_range[0]
    r2=param_range[1]
    while(r1<r2+1):
        para.append(r1)
        r1+=1
    params=sample(para,10)
    params.sort()

    for k in params:

        #Splitting x-train and y-train into subarrays according to the fold value.
        x_t=[]
        y_t=[]
        for i in range (0,folds):
            j=0
            if i>=1:
                j=j+1
            x_t.append(X_train[(int((len(X_train)/folds)*i)+j):(int(len(X_train)/folds)*(i+1))])
            y_t.append(Y_train[(int((len(Y_train)/folds)*i)+j):(int(len(Y_train)/folds)*(i+1))])
        train_scores_fold=[]
        cv_scores_fold=[]
        for i in range (0,folds):


            x_cv=x_t[i]
            y_cv=y_t[i]
            x_train=np.zeros((1,17))
            y_train=np.zeros((1))
            for m in range (0,folds):
                if i!=m:
                    x_train=np.concatenate((x_train,x_t[m]))
                    y_train=np.concatenate((y_train,y_t[m]))

            #Calculating r2 scores for test and cv datasets.
            x_train=np.delete(x_train,0,0)
            y_train=np.delete(y_train,0,0)

            regressor.n_neighbors=k
            regressor.fit(x_train,y_train)

            Y_predicted=regressor.predict(x_cv)
            cv_scores_fold.append(r2_score(y_cv,Y_predicted))

            Y_predicted=regressor.predict(x_train)
            train_scores_fold.append(r2_score(y_train,Y_predicted))
        train_scores.append(np.mean(np.array(train_scores_fold)))
        cv_scores.append(np.mean(np.array(cv_scores_fold)))
    return train_scores,cv_scores,params
param_range=(1,25)
folds=10
neigh = KNeighborsRegressor()
trainscores,testscores,params=RandomSearchCV(X_train,y_train,neigh,param_range,folds)
plt.plot(params,trainscores, label='train curve')
plt.plot(params,testscores, label='test curve')
plt.title('Hyper-parameter VS R2 score plot')
plt.legend()
plt.show()
```

5.3 Model Performance on Default Settings with 4-fold cross-validation.

5.3.1 Linear Regression

The following parameters were selected on the default setting for Linear Regression and the R-. squared value for all folds and the overall mean is shown below:

Table 5.1: Default Setting - Linear Regression.

Parameter	**Value**
copy_X	TRUE
fit_intercept	TRUE
n_jobs	None
normalize	deprecated
positive	FALSE

The r-squared value on the test dataset is 0.6683 and the time taken for training is 64.8 ms.

Table 5.2: 4-fold CV Score - Linear Regression.

4-Fold CV	
Folds	**CV Score-R2**
CV1	0.5467
CV2	0.6769
CV3	0.6605
CV4	0.6004
Mean R2	0.6211

5.3.2 Lasso Regression

The following parameters were selected on the default setting for Lasso Regression.

Table 5.3: Default Setting - Lasso Regression.

Parameter	**Value**
alpha	1
copy_X	TRUE
fit_intercept	TRUE
max_iter	1000
normalize	deprecated
positive	FALSE
precompute	FALSE
random_state	None
selection	Cyclic
tol	0.0001
warm_start	FALSE

Table 5.4: 4-fold CV Score - Lasso Regression.

4-Fold CV	
Folds	**CV Score-R2**
CV1	0.5467
CV2	0.6769
CV3	0.6605
CV4	0.6004
Mean R2	0.6211

The r-squared value on the test dataset is 0.6683, and the time taken for training is 179 ms. It can be observed that there is not much difference between Linear Regression and Lasso Regression in the default setting.

5.3.3 K-NN Regressor

The following parameters were selected on the default setting for K-NN Regression.

The r-squared value on the test dataset is 0.8432, and the time taken for training is 1.34 s.

Table 5.5: Default Setting - KNN Regression.

Parameter	**Value**
algorithm	auto
leaf_size	30
metric	minkowski
metric_param	none
n_jobs	none
n_neighbours	5
p	2
weights	uniform

Table 5.6: 4-fold CV Score - KNN Regression.

4-Fold CV	
Folds	**CV Score-R2**
CV1	0.6811
CV2	0.8688
CV3	0.8698
CV4	0.7911
Mean R2	0.8027

5.3.4 Decision Tree

Table 5.7: Default Setting - Decision Tree.

Parameter	**Value**
ccp_alpha	0
criterion	squared_error
max_depth	none
max_features	none
max_leaf_nodes	none
min_impurity_decrease	0
min_samples_leaf	1
min_samples_split	2
min_weight_fraction_leaf	0
random_state	none
splitter	best

Table 5.8: 4-fold CV Score - Decision Tree.

4-Fold CV	
Folds	**CV Score-R2**
CV1	0.7132
CV2	0.8584
CV3	0.6239
CV4	0.6790
Mean R2	0.7186

The r-squared value on the test dataset is 0.8831, and the time taken for training is 414 ms. The decision tree algorithm has shown improvement both in R-squared value and training time.

5.3.5 Random Forest

Table 5.9: Default Setting - Random Forest.

Parameter	**Value**
ccp_alpha	0
criterion	squared_error
max_depth	none
max_features	auto
max_leaf_nodes	none
min_impurity_decrease	0
min_samples_leaf	1
min_sample_split	2
min_weight_fraction_leaf	0
random_state	none
splitter	best
max_samples	none
n_estimators	100
n_jobs	none
verbose	0

Table 5.10: 4-fold CV Score - Random Forest.

4-Fold CV	
Folds	**CV Score-R2**
CV1	0.7523
CV2	0.9193
CV3	0.7232
CV4	0.8202
Mean R2	0.8037

The r-squared value on the test dataset is 0.8825, and the time taken for training is 23.1 s. The training time for random forest is more than compared to other algorithms.

5.3.6 XG Boost

The r-squared value on the test dataset is 0.9110, and the time taken for training is 2.15 s.

TABLE 5.11: Default Setting - XGBoost.

Parameter	**Value**
base_score	0.5
booster	gbtree
colsample_bylevel	1
gamma	0
importance_type	gain
learning_rate	0.1
max_depth	3
n_estimators	100

TABLE 5.12: 4-fold CV Score - XGBoost.

4-Fold CV	
Folds	**CV Score-R2**
CV1	0.7751
CV2	0.6582
CV3	0.7835
CV4	0.9270
Mean R2	0.7859

5.4 Hyperparameter Tuning

5.4.1 Random Search CV with k-fold CV from scratch for K-NN Regressor

Here the k value is taken as a hyperparameter and with random search cv train and test scores are calculated on 4-fold cv. The plot for the train test score with respect to k value is plotted and the correct value of k is chosen from the plot. The code for the above model is already shown in the above section.

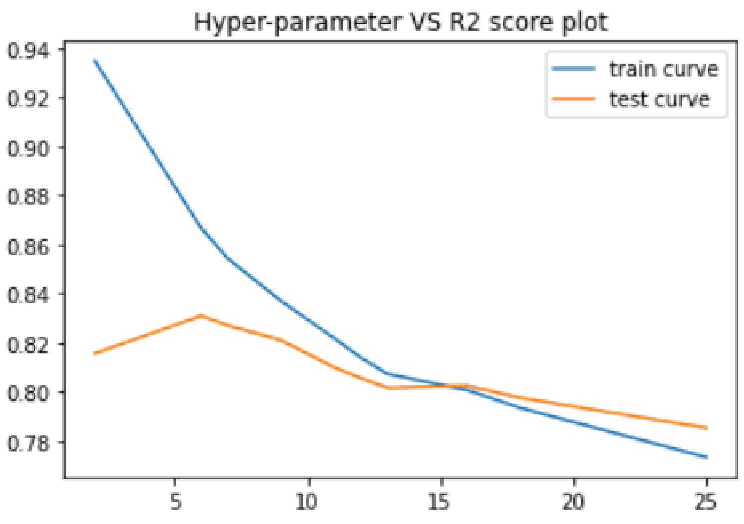


FIGURE 5.1: Hyper-parameter VS R2 score plot

Here in Fig. 5.1, k=15 is best value with an r-squared score of 0.81. The total training time of the model is 7 m 39 s.

5.4.2 Grid Search with 4-fold CV for Lasso Regression

The following values for alpha is taken as hyperparameter: [1.00000000e-03, 2.89426612e-03, 8.37677640e-03, 2.42446202e-02, 7.01703829e-02, 2.03091762e-01, 5.87801607e-01, 1.70125428e+00, 4.92388263e+00, 1.42510267e+01, 4.12462638e+01, 1.19377664e+02, 3.45510729e+02, 1.00000000e+03]
The best-fit value of alpha with the best R-squared value is alpha = 1.19377664e+02 with an R-squared score of 0.6212 and a training time of 1.84 s. The same model on the test dataset gave the following R2 value of 0.6682.

5.4.3 Grid Search with 4-fold CV for Decision Tree

Two parameters are taken as hyperparameters in the tuning process:
Max_depth: range(1,15)
Max_leaf_nodes: range(0,100)
The value of best fit, which gives the best R2 score, are : Max_depth: 8
Max_leaf_nodes: 96
The R-squared score from the above setting on the test dataset is 0.8626, which has a significant improvement from the default setting. Also, the training is on the higher side with 2 m 12 s.

5.4.4 Grid Search with 4-fold CV for XGBoost

Three parameters are selected for the XG Boost algorithm for hyperparameter tuning:
Max_depth: range(1,10)
Learning_rate: 0,0.1,0.2,. . . . . . till 1
n_estimators: range(1,10)
Grid search with 4-fold cv was applied and the best parameters were: Max_depth = 9
Learning_rate = 0.55
n_estimators = 4
The R-squared score from the above setting is 0.9039. Also, the training is on the higher side with 1 m 54 s.

5.4.5 Random Search with 4-fold CV for Random Forest

Parameters are selected for the Random Forest algorithm for hyperparameter tuning:
n_estimators: range(1,10)
Random search with 4-fold cv was applied and the best parameters were: n_estimators = 1000
The R-squared score from the above setting is 0.8712. Also, the training is very high with 1 h 48 m.

Chapter 6

Results and Conclusion

The R-2 value has been used to evaluate the models and each model has been trained with default settings and hyper-parameter tuning with grid search or random search. A 4-Fold cross-validation is applied to all models. Table 6.1 describes the R-2 values of all the models with model training time.

Table 6.1: Model evaluation.

Models	Default Setting		After Hyperparameter Tuning with Grid Search or Random Search	
	R-2 Score	Training Time	R-2 Score	Training Time
Linear Regression	0.6211	64.8ms	-	-
Lasso Regression	0.6211	179ms	0.6212	1.84s
K-NN	0.8027	1.34s	0.8100	7min 39s
Decision Tree	0.6939	414ms	0.7274	2min 12s
Random Forest	0.8037	23.1s	0.8544	1hr 48min
XG Boost	0.7859	2.15s	0.7999	1min 54s

Random forest is the best-performing model but the training time after tuning is high whereas XG BOOST model has shown improvement after tuning and also has a low training time. K-nn regressor is also a good option for both in terms of training time and performance.

Chapter 7

Deployment

The process of deployment typically involves taking a trained machine-learning model and integrating it into a larger software application or system. This may involve implementing the model within a web service, mobile application, or other software systems, and providing an interface through which users or other software components can interact with the model.
Linear Regression, Lasso Regression, KNN Regression, Decision tree, Random Forest, and XGBOOST have been evaluated in this project, out of which the Random Forest model performed the best. Hence we have used Random Forest for Deployment.

7.1 Web application using Flask

Flask is a popular Python web framework that can be used for building web applications. It's lightweight and easy to use, which makes it a great choice for deploying machine-learning models.
The following steps are used to create a application:

1. Create a Flask application: First, you need to create a Flask application by importing the Flask class from the flask module and creating an instance of it.

2. Define a route: You need to define a route that will be used to handle the incoming requests. This can be done using the @app.route decorator. In the following example, we define a route /prediction that will handle POST requests:

```
@app.route('/predict', methods=['POST'])
def predict():
    # code to handle the incoming request
    return result
```

3. Load the machine learning model: Load the machine learning model that you want to deploy. You can use any Python machine-learning library to train the model and save it to a file.

```
import joblib

model = joblib.load('model.pkl')
```

4. Process the incoming request: In the route function, you need to process the incoming request and prepare the input data for the model.

```
@app.route('/predict', methods=['POST'])
def predict():
    data = request.json
    input_data = data['input']

    # preprocess input_data
    processed_data = preprocess(input_data)

    # make predictions
    predictions = model.predict(processed_data)

    # return predictions as JSON
    return jsonify(predictions.tolist())
```

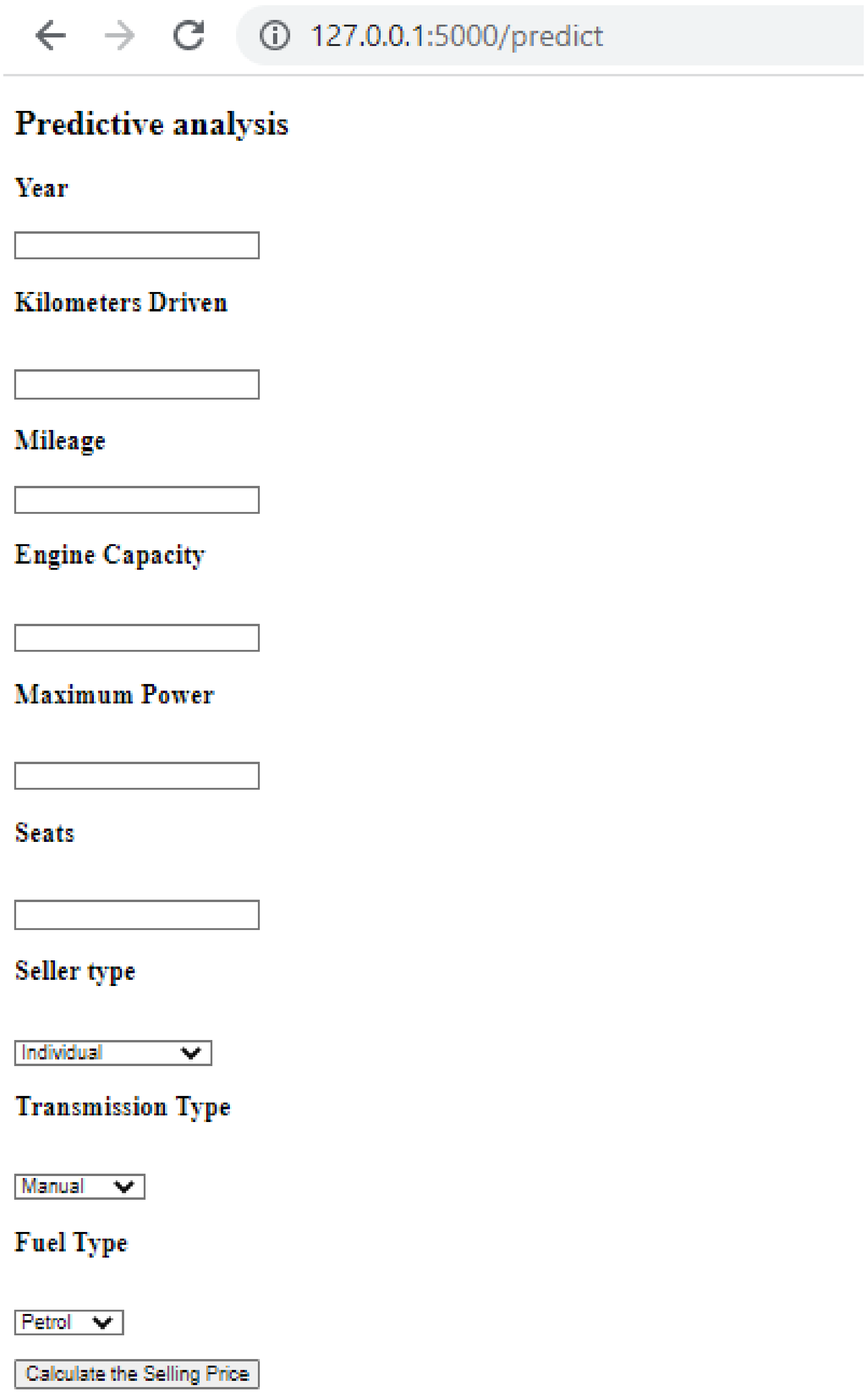
5. Run the application: Finally, you can run the Flask application using the run() method. By default, the application will be available at http://localhost:5000/.

```
    if __name__ == '__main__':
    app.run()
```

7.2 Front End using HTML

The following HTML form has been prepared to input the values from the users and display the predicted results as shown in Fig. 7.1.

Fig. 7.2 shows the prediction results after taking inputs from users.



127.0.0.1:5000/predict

Predictive analysis

Year

Kilometers Driven

Mileage

Engine Capacity

Maximum Power

Seats

Seller type

Individual

Transmission Type

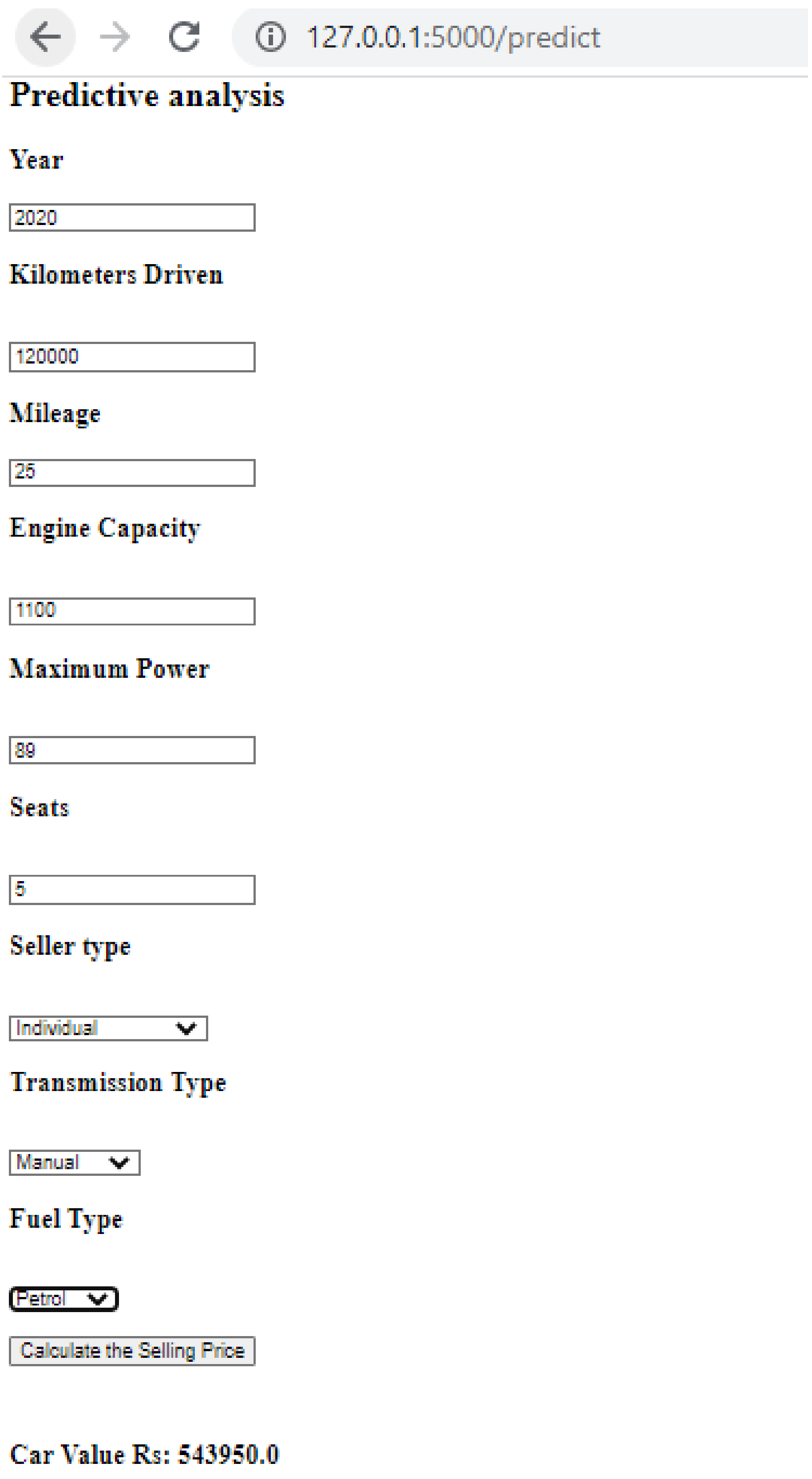
Manual

Fuel Type

Petrol

Calculate the Selling Price

Figure 7.1: HTML Form


127.0.0.1:5000/predict

Predictive analysis

Year

2020

Kilometers Driven

120000

Mileage

25

Engine Capacity

1100

Maximum Power

89

Seats

5

Seller type

Individual

Transmission Type

Manual

Fuel Type

Petrol

Car Value Rs: 543950.0

FIGURE 7.2: Prediction form