VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI-590018, KARNATAKA

An Internship Report
On

USED CAR PRICE PREDICTION

Submitted in partial fulfilment of the requirements for the VIII Semester of degree of ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belagavi*

Submitted by

K S VINAY

1RN20IS067

Under the Guidance of

Ms. Chaitra S

Assistant Professor

Department of ISE

ESTD : 2001
An Institute with a Difference

Department of Information Science and Engineering

RNS Institute of Technology

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post,
Channasandra, Bengaluru-560098

2023 – 2024

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post,
Channasandra,Bengaluru - 560098

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

ESTD : 2001
An Institute with a Difference

CERTIFICATE

Certified that the internship work entitled ***Used Car Price Prediction*** has been successfully completed by **K S VINAY (1RN20IS067)** a bonafide student of **RNS Institute of Technology, Bengaluru** in partial fulfilment of the requirements for the award of degree in **Bachelor of Engineering in Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year **2023-2024**. The internship report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

Ms. Chaitra S Internship Guide Assistant Professor Department of ISE	**Dr. Nirmalkumar S Benni /** **Ms. Aruna U** Internship Coordinators Associate / Assistant Professor Department of ISE	**Dr. Suresh L** Professor & Head Department of ISE RNSIT

External Viva

Name of the Examiners **Signature with Date**

1. ____________________________ **1.** ______________________

2. ____________________________ **2.** ______________________

NASTECH

Empowering People through New Age Solutions & Technologies

Microsoft Partner Network

CERTIPORT
A PEARSON VUE BUSINESS
AUTHORIZED TESTING CENTER

CERTIFICATE

This is to certify that **K S VINAY** bearing USN **1RN20IS067** from RNSIT has taken part in internship training on the Artificial Intelligence - Machine Learning & Data Science using Python & Azure and successfully completed project work "Used Cars Price Prediction" in New Age Solutions Technologies (NASTECH) from August 2023 to September 2023.

K S VINAY has shown great enthusiasm and interest in the project. The incumbents' conduct and performance were found satisfactory during all phases of the project.

Thank you very much.
Sincerely Yours,

Deepak Garg
Founder

DECLARATION

I, **K S VINAY [1RN20IS067]**, student of VIII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Internship project work entitled ***Used Car Price Prediction*** has been carried out and submitted in partial fulfilment of the requirements for the *VIII Semester degree of* ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belagavi* during academic year 2023-2024.

Place: Bengaluru
Date:

K S VINAY
[USN:1RN20IS067]

ABSTRACT

This project explores the application of Linear Regression and Random Forest algorithms in predicting used car prices. Leveraging a diverse dataset encompassing car features like make, model, year, mileage, and fuel type, the study employs data pre-processing techniques to enhance model robustness. Linear Regression serves as a baseline model, offering interpretability, while Random Forest captures intricate relationships in the data. Using scikit-learn in Python, models are rigorously trained, validated, and tested, with cross-validation to ensure reliability. The project evaluates model performance, highlighting accuracy, precision, recall, and feature importance. The insights gained from this comparative analysis provide valuable guidance for stakeholders in the automotive industry, facilitating data-driven decision-making for buyers, sellers, and dealers in the dynamic used car market.

ACKNOWLEDGMENT

At the very onset I would like to place our gratefulness to all those people who helped me in making the Internship a successful one.

Coming up, this internship to be a success was not easy. Apart from the sheer effort, the enlightenment of the very experienced teachers also plays a paramount role because it is they who guided me in the right direction.

First of all, I would like to thank the **Management of RNS Institute of Technology** for providing such a healthy environment for the successful completion of internship work.

In this regard, I would like to express my sincere gratitude to our beloved Director and Principal **Dr. M K Venkatesha** and **Dr. Ramesh Babu H S** for providing us all the facilities

I'm extremely grateful to our own and beloved Professor and Head of Department of Information science and Engineering, **Dr. Suresh L**, for having accepted to patronize me in the right direction with all her wisdom.

I place my heartfelt thanks to **Ms. Chaitra S** Assistant Professor, Department of Information Science and Engineering for having guided internship and all the staff members of the department of Information Science and Engineering for helping at all times.

I thank **Mr. Deepak Garg,** Data Scientist, NASTECH, for providing the opportunity to be a part of the Internship program and having guided me to complete the same successfully.

I also thank our internship coordinators **Dr. Nirmalkumar S benni**, Associate Professor and **Ms. Aruna U**, Assistant Professor of Department of Information Science and Engineering.

I would thank my parents for having supported me with all their strength and might. Last but not the least, I thank my parents for supporting and encouraging me throughout. I have made an honest effort in this internship project.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATION

AI	:	Artificial Intelligence
ML	:	Machine Learning
MSE	:	Mean Square Error
RMSE	:	Root Mean Square Error
RAM	:	Random Access memory
DF	:	Data Frame

Chapter 1

INTRODUCTION

1.1 About the Project

Machine learning offers a powerful tool for predicting used car prices. By analysing historical sales data, models can learn to estimate fair market values. This project compares two algorithms: Random Forest and Linear Regression. Random Forest excels at non-linear relationships and boasts high accuracy, while Linear Regression prioritizes interpretability and simplicity. Both models will be trained on a used car sales dataset and evaluated on their predictive power. The best performing model will be chosen for deployment in a user-friendly interface, enabling informed decision-making within the used car market.

1.1.1 Domain/Technology

The domain chosen for our project is AI/ML. Machine learning, the fundamental driver of AI, is possible through algorithms that can learn themselves from data and identify patterns to make predictions and achieve your predefined goals, rather than blindly following detailed programmed instructions, like in traditional computer programming. This technology allows the machine to perceive, learn, reason, and communicate through observation of data, like a child that grows up and acquires knowledge from examples. With machine learning, manufacturing companies have increased production capacity up to 20%.

AI technology has undergone a transformative journey from rule-based expert systems to more sophisticated approaches like machine learning, deep learning, artificial neural networks, and reinforcement learning. Unlike traditional rule-based systems, modern AI focuses on learning representations instead of predefined tasks, drawing inspiration from the complexity of animal brains. This evolution allows AI to comprehend intricate industrial processes, optimizing entire production flows instead of isolated tasks. With significant cognitive capacity, AI considers spatial plant organization and temporal constraints, contributing to the seamless enhancement of production processes and overcoming complexities in the manufacturing of living products.

1.1.2 Company Profile

NASTECH is formed with the purpose of bridging the gap between Academic and Industry. NASTECH is one of the leading Global Certification and Training service providers for technical and management programs for educational institutions. They collaborate with educational institutes to understand their requirements and form a strategy in consultation with all stakeholders to fulfil those by skilling, reskilling and upskilling the students and faculties on new age skills and technologies.

1.2 Problem Statement

1.2.1 Existing System and Their Limitations

With the rise in emerging technologies like artificial intelligence, IoT, and quantum computing, computer scientists have started research on how to clone our thought processes into computers. This AI technology has eliminated human dependency to a large extent across industries.

1.2.2 Proposed Solution

The project proposes predicting used car prices using Linear Regression and Random Forest algorithms. Leveraging a diverse dataset and meticulous pre-processing, it ensures data quality. Linear Regression offers simplicity, while Random Forest captures complexities. Implementation in Python's scikit-learn includes rigorous training, validation, and metric analysis, providing insights for informed automotive decisions.

1.2.3 Program Formulation

This project seeks to predict used car prices using Linear Regression and Random Forest algorithms. Objectives involve developing and training models, comparing performance metrics, and delivering actionable insights. Activities encompass dataset collection, feature identification, model implementation, and thorough evaluation. Outputs include trained models and a comparative analysis with documented findings. This systematic approach ensures the project's goal of leveraging machine learning for accurate used car price prediction, aiding informed decision-making in the automotive industry.

Chapter 2

LITERATURE SURVEY

Used Car Price Prediction Using Random Forest Algorithm (Prof. Dipti A. Gaikwad, Pratik S. Suwarnakar, Yash R. Mahajan, Amita U. Petkar)

The document discusses the use of machine learning algorithms, specifically Lasso Regression, Support Vector Machine (SVM), Linear Regression, and Random Forest, to predict used car prices. The aim is to develop an efficient model that can accurately determine the price of a used car based on various factors. The document also includes a literature survey summarizing previous research in the field and provides information on the software requirements for implementing the proposed system. The Random Forest algorithm is chosen as the best fit algorithm based on its higher R2 score. The proposed system aims to assist consumers in making informed decisions when buying a second-hand car.

Used car price prediction using linear regression model (Ashutosh Datt Sharma*1, Vibhor Sharma*2)

The document discusses the use of a linear regression model to predict the price of used cars. It highlights the increasing production of cars and the subsequent growth of the used car market. The availability of online portals and websites has made it necessary for customers, clients, dealers, and sellers to be updated with the current market trends and pricing information for used cars. The document emphasizes the application of machine learning, specifically linear regression, in solving prediction problems. The methodology section explains the collection of data from an online portal and the selection of relevant attributes for prediction. The results and analysis section presents visualizations that show the correlation between the selling price of used cars and various factors such as seller type, transmission, fuel type, kilometres driven, years used, and owner history. The document concludes by discussing the accuracy of the linear regression model, with an r2 score of 0.86. The study is deemed valuable for used car sellers, online pricing services, and individuals interested in buying or selling used cars.

Using linear regression for used car price prediction (sümeyra muti, kazım yildiz)

The document discusses the application of linear regression in predicting used car prices, particularly focusing on the context of Turkey in 2020. It highlights the widespread use of machine learning algorithms, including linear regression, for price prediction across various domains. The study employs a dataset containing vehicle features and prices, split into training and test sets. The results indicate a 73% R2 score, measuring model prediction success. Factors influencing used car prices, such as model, year, kilometres, gear type, and comfort features, are emphasized. The document references other studies utilizing machine learning for price estimation in diverse fields and underscores the importance of data pre-processing for enhanced accuracy. The authors suggest further research with improved datasets to refine predictions.

Predicting the value of used car using Linear regression (Raj Upreti, Rajiv Dahal)

The document is a report on a minor project titled "Used Car Price Prediction Using Linear Regression." The project aims to build a model that can predict the price of used cars in Nepal. The authors collected data from a web portal and used machine learning techniques, specifically linear regression, to develop the prediction model. The report provides an overview of artificial intelligence (AI) and machine learning concepts, including supervised learning and regression. It discusses the problem of predicting used car prices and presents the objectives of the project. The authors also conducted a literature review of previous studies on car price prediction. The report includes sections on feasibility, system design, methodology, and achievements. The achieved accuracy of the prediction model is reported to be 80.45%. The document references other studies utilizing machine learning for price estimation in diverse fields and underscores the importance of data pre-processing for enhanced accuracy. Overall, the report provides a comprehensive overview of the project on used car price prediction using linear regression.

Chapter 3

REQUIREMENT ANALYSIS, TOOLS & TECHNOLOGY

3.1 Hardware and Software Requirements

3.1.1 Hardware Requirements

- Processor: 8 Octa Core or above
- RAM: 6 GB or more
- Hard Disk: 100 GB or more

3.1.2 Software Requirements

- Operating System: Windows 7 or above
- IDE: Jupyter Notebook
- Engine: Python3

3.2 Tools/Languages/Technologies

- Pandas
- NumPy
- Matplotlib
- seaborn
- scikit-learn
- CUDA-compatible GPU (optional for enhanced performance)

Chapter 4

SYSTEM DESIGN

4.1 Problem Statement

Earlier to predict the price or value of used vehicles was a tedious task which should be done manually and the accuracy was less, hence by using the emerging trend in machine learning we could easily evaluate the price of such used vehicles by considering the required parameters like kilometres driven, transmission type, fuel type, model, vehicle year etc. The dataset undergoes a thorough data cleaning process, involving the removal of irrelevant columns and handling missing values. Subsequently, feature engineering is applied to extract meaningful information from certain columns, such as converting the manufacturing year to the age of the car. Categorical variables are encoded using one-hot encoding, and the dataset is split into training and testing sets. Two regression models, Linear Regression and Random Forest Regression, are implemented and trained on the pre-processed data. The models are then evaluated using the R2 score, a metric indicating how well the predicted values align with the actual prices. The code provides a visual comparison of predicted versus actual prices for both models. Ultimately, the goal is to create an effective predictive model for used car prices, offering insights into the factors influencing the resale value of automobiles.

4.2 Methods

The main methods that are followed in the project are:

- Data Import, Exploration and cleaning
- Data Pre-processing
- Train-Test Split
- Training and predicting

4.3 Libraries

- Numpy, pandas (for data processing)
- Matplotlib, seaborn (for data visualization)
- Scikit learn (for model building)

4.4 Flow Chart

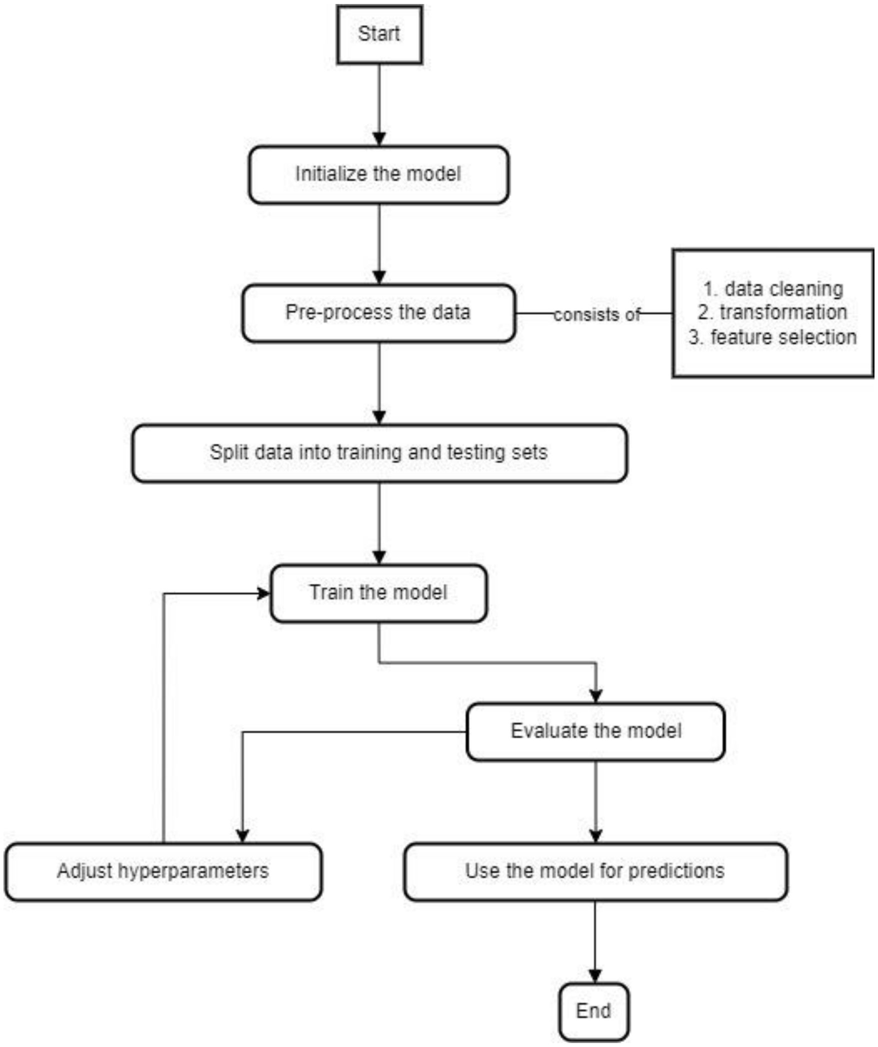


Figure 4.1: *Flow Chart*

The flowchart shown in Figure 4.1 explains about the whole project of Used car price prediction by using an algorithmic approach. The steps are: importing a dataset of used cars, performing data cleaning to handle missing values and irrelevant columns. Feature engineering includes converting the manufacturing year to car age and extracting numerical values from categorical columns. The data is split into training and testing sets, and categorical variables are encoded. Standardization is applied, and the required models are trained based on our preference. Then model performance is evaluated using the result score, with visual comparisons of predicted versus actual prices.

Chapter 5

DETAILED DESIGN

5.1 High level design

High-level design explains the architecture that would be used for developing software products. The architecture diagram provides an overview of an entire system, identifying the main components that would be developed for the product and their interfaces.

5.1.1 Architecture of Python Notebook

The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results. The Jupyter notebook combines two components:

- A web application: a browser-based tool for interactive authoring of documents that combine explanatory text, mathematics, computations, and their rich media output.
- Notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

5.1.2 The Python Kernel

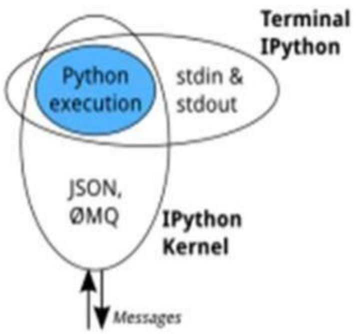


Figure 5.1: Kernel Execution

All the interfaces, like The Notebook, the Qt console, IPython console in the terminal, and third-party interfaces use the IPython Kernel. The IPython Kernel is a separate process which is responsible for running user code, and things like computing possible completions. Frontends, like the notebook or the Qt console, communicate with the IPython Kernel using JSON-messages sent over ZeroMQsockets. The co-execution machinery for the kernel is shared with terminal IPython as shown in figure 5.1. A kernel process can be connected to more than one frontend simultaneously.

5.1.3 The Notebook

The Notebook frontend does something extra. In addition to running your code, it stores code and output, together with markdown notes, in an editable document called a notebook as shown in figure 5.2. When you save it, this is sent from your browser to the notebook server, which saves it on disk as a JSON file with a “.ipynb” extension.

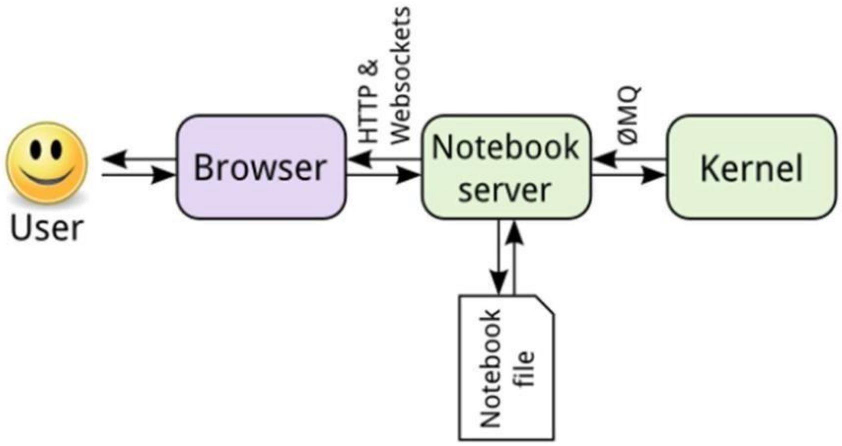


Figure 5.2: User to Notebook Interface

The kernel, is responsible for saving and loading notebooks, so you can edit notebooks even if you don't have the kernel for that language, you just won be able to run code. The kernel doesn't know anything about the notebook document: just gets sent cells of code to execute and interpret the results of that.

5.2 Linear regression Algorithm

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data. The simplest form of linear regression with one independent variable is known as simple linear regression, while multiple linear regression involves more than one independent variable.

In the context of simple linear regression, the relationship between the dependent variable (often denoted as (y)) and the independent variable (often denoted as (x)) is represented by the equation: [y = mx + b]

where: (y) is the dependent variable, (x) is the independent variable, (m) is the slope of the line (the change in (y) with respect to a one-unit change in (x)), (b) is the y-intercept (the value of (y) when (x) is 0). The goal of linear regression is to find the values of (m) and (b) that minimize the sum of the squared differences between the observed and predicted values of the dependent variable.

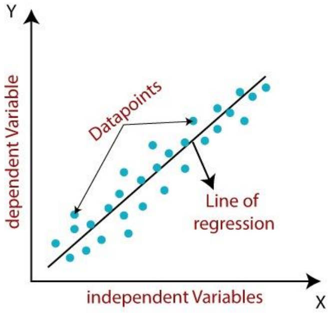


Figure 5.3: *linear regression graph*

5.3 Building Linear regression model

Building a linear regression model involves defining the problem, collecting and exploring relevant data, cleaning and preparing the dataset, choosing the linear regression model, defining a cost function (typically mean squared error), training the model on the training set, evaluating its performance on the testing set using metrics like MSE, interpreting coefficients, making predictions, and iterating to improve if necessary. Linear regression assumes a linear relationship between variables and has specific assumptions to consider, such as independence of errors. The model provides insights into the relationship between the dependent and independent variables for predictive analytics.

5.4 Working of linear regression model

The working of linear regression involves finding the best-fitting line that represents the linear relationship between the dependent variable (target) and one or more independent variables (features) in a dataset. Here's a step-by-step overview:

- **Initialize Parameters**: Start with initial values for the coefficients (slope and intercept).
- **Define the Model**: Formulate the linear equation representing the relationship between the independent and dependent variables: (y = mx + b).
- **Define the Cost Function**: Choose a cost function, typically the mean squared error (MSE), which quantifies the difference between predicted and actual values.
- **Optimization:** For gradient Descent Adjust the coefficients iteratively to minimize the chosen cost function. Gradient descent is a common optimization algorithm used for this purpose.
- **Update Coefficients**: Update the coefficients using the gradient of the cost function with respect to each coefficient. This process continues until convergence, where further adjustments do not significantly improve the model.
- **Model Evaluation**: Evaluate the model's performance using a separate test dataset. Common metrics include MSE, RMSE, and R-squared.
- **Interpret Coefficients**: Interpret the coefficients to understand the strength and direction of the relationship between variables. The slope (m) represents the change in the dependent variable for a unit change in the independent variable.
- **Make Predictions**: Use the trained model to make predictions on new or unseen data.

Linear regression aims to find the line that minimizes the overall difference between predicted and actual values. The process involves mathematical optimization to adjust the model parameters and iteratively improve its fit to the data. The result is a simple and interpretable model that can be used for prediction and understanding the relationships between variables.

5.5 Random Forest Algorithm

Random Forest is an ensemble learning algorithm that combines the strength of multiple decision trees to enhance predictive performance. It operates by constructing an ensemble of trees during training, each trained on a random subset of the data and considering a random subset of features at each split. This randomness and diversity among the trees help mitigate overfitting and improve generalization. For classification tasks, the final prediction is determined by a majority vote of the individual tree predictions, while for regression tasks, it is the average of the predictions. Random Forests are known for their robustness, accuracy, and ability to handle complex datasets, making them a popular choice in machine learning for tasks ranging from classification to regression and feature importance analysis.

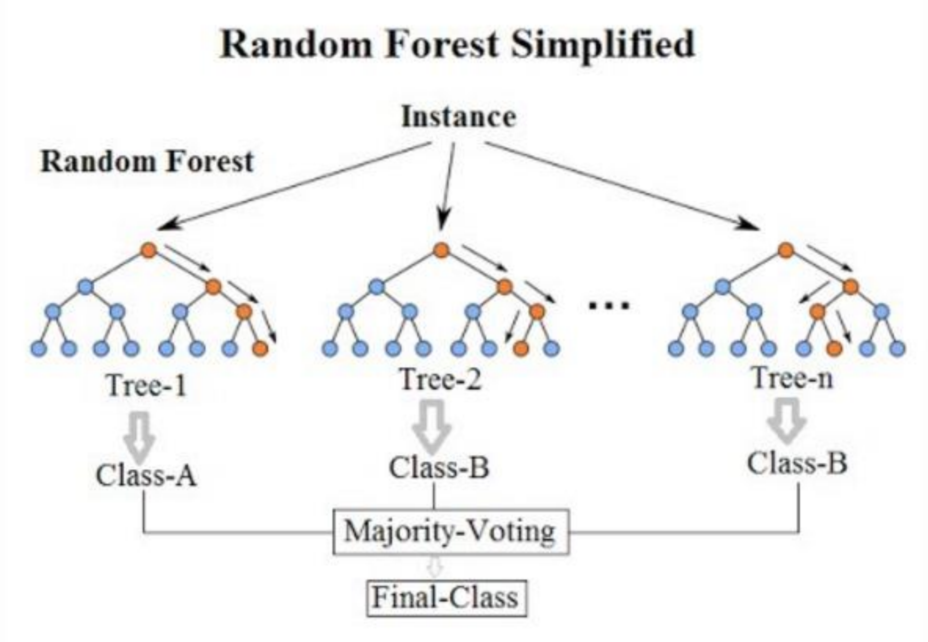


Figure 5.4: *Random Forest Model*

5.6 Building the Random Forest Model

To construct a Random Forest model, begin by gathering and pre-processing your dataset, ensuring it is well-suited for the task at hand by addressing missing values and encoding categorical variables. Depending on whether the objective is classification or regression, choose the appropriate model type—either Random Forest Classifier or Random Forest Regressor from libraries like scikit-learn. Instantiate the model, customizing hyperparameters such as the number of trees to align with specific requirements. Train the model using the fit method on the training data and subsequently make predictions on the test dataset. Evaluate the model's performance using relevant metrics, such as accuracy for classification tasks or mean squared error for regression. Optionally, fine-tune the model by adjusting hyperparameters to optimize performance. Explore feature importance to gain insights into the contribution of each feature to the model's predictions. Iterate and improve the model as needed, considering adjustments to parameters, pre-processing techniques, or the inclusion of additional features. The Random Forest model, with its ensemble of decision trees, proves to be a robust and versatile tool in machine learning, offering accurate predictions and accommodating complex relationships within diverse datasets.

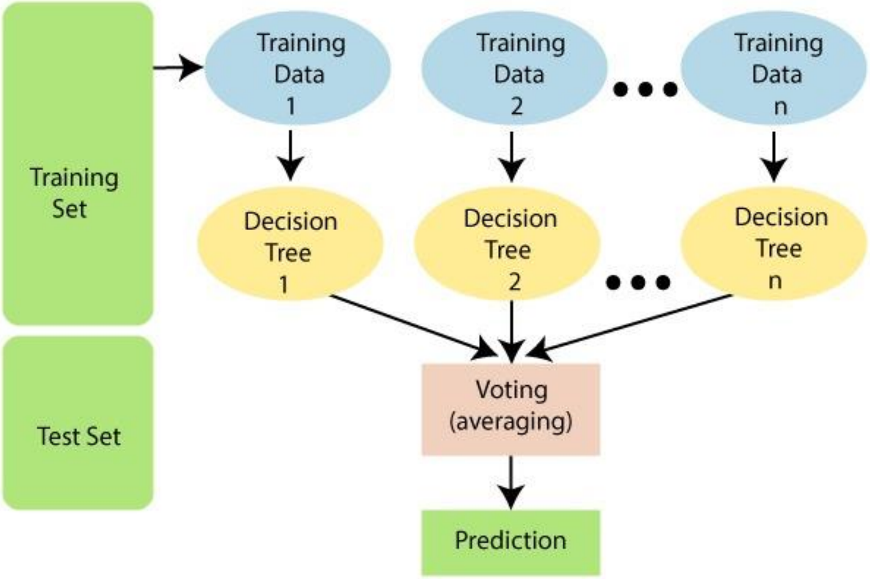


Figure 5.5: *Building the Random Forest Model*

5.7 Working of Random Forest Model

The working of a Random Forest involves creating an ensemble of decision trees and combining their outputs for more robust and accurate predictions. Here's a step-by-step explanation of how Random Forest works:

- **Sampling**: subsets of the training data are sampled with replacement (bootstrapped sampling) to create diverse datasets for each tree in the ensemble.
- **Feature Randomness**: At each node of every decision tree, a random subset of features is considered for splitting. This introduces variability among the trees, preventing them from becoming overly correlated.
- **Decision Trees Construction**: Multiple decision trees are constructed using the bootstrapped datasets and feature subsets. Each tree is grown deep, employing recursive binary splits based on information gain or reduction in impurity for classification tasks and mean squared error reduction for regression tasks.
- **Voting or Averaging**: For classification tasks, the final prediction is determined by a majority vote among the individual trees. For regression tasks, it is the average of the predictions made by all trees in the forest.
- **Ensemble Strength**: The ensemble of trees collectively makes predictions that are more robust and less susceptible to overfitting compared to individual trees.
- **Predictions**: Once trained, the Random Forest can make predictions on new or unseen data by passing the data through each tree in the forest and aggregating the results.
- **Feature Importance**: Random Forests can provide insights into feature importance, indicating the contribution of each feature to the overall predictive power of the model. This is calculated based on how often a feature is used for splitting across all trees.

Random Forests are known for their versatility, high accuracy, and resistance to overfitting. They are widely used in various machine learning tasks, including classification, regression, and feature importance analysis. The combination of multiple trees, each trained on a different subset of data, results in a powerful ensemble model suitable for complex datasets.

5.8 Training and predicting

Now let’s discuss the training and prediction steps in simpler terms:

5.8.1 Training the Models:

5.8.1.1 Linear Regression Model:

1. Standardizing Data: The features of the training data are standardized, which means they are transformed to have a mean of 0 and a standard deviation of 1. This helps in making sure all features are on a similar scale.

2. Creating the Model: A Linear Regression model is created. This model aims to find a linear relationship between the input features (like car age, mileage, etc.) and the target variable (used car price).

3. Training the Model: The model is trained using the standardized training data. During training, the model learns the patterns and relationships in the data to make predictions.

5.8.1.2 Random Forest Model:

1. Creating the Model: A Random Forest Regressor model is created. This type of model uses an ensemble of decision trees to make predictions.

2. Training the Model: Similar to Linear Regression, the Random Forest model is trained on the training data. It learns complex patterns and relationships in the data.

5.8.2 Making Predictions:

5.8.2.1 Linear Regression Model:

Using the Model: The trained Linear Regression model is used to make predictions on the test data. It predicts the prices of used cars based on the features present in the test dataset.

5.8.2.2 Random Forest Model:

Using the Model: The trained Random Forest model is also used to make predictions on the same test data. It combines predictions from multiple decision trees to arrive at a final prediction.

5.8.3 Model Evaluation:

1. **R2 Score**: After making predictions, the script calculates the R2 score for each model. R2 score is a measure of how well the model predicts the actual prices. A higher R2 score indicates a better fit to the data.
2. **Comparison**: The script then prints and compares the R2 scores of the Linear Regression and Random Forest models. This comparison helps in understanding which model performs better in predicting used car prices. In our project, the Random Forest model is found to have a higher R2 score, suggesting better performance compared to Linear Regression.

Chapter 6

IMPLEMENTATION

Implementation is the process of defining how the system should be built, ensuring that it is operational and meets quality standards. It is a systematic and structured approach for effectively integrating a software-based service or component into the requirements of end-users.

6.1 Overview of System Implementation

The plan contains an overview of the system, a brief description of the major tasks involved in the implementation, the overall resources needed to support the implementation effort and any site-specific implementation requirements.

6.1.1 Selection of Programming Language-Python

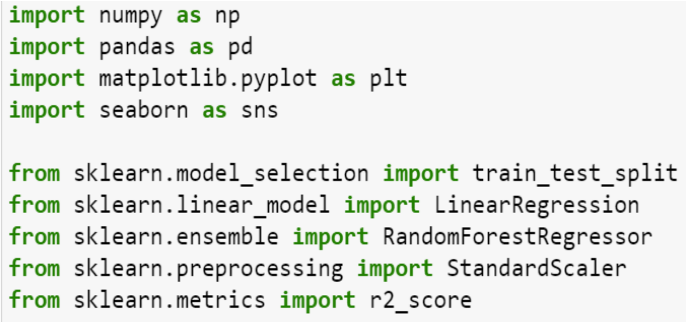
Programmers prefer python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy. A bug or bad input will never cause a segmentation fault.

Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source-level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. On the other hand, often the quickest way to debug a program is to add a few print statements to the Source. The fast edit-test-debug cycle makes this simple approach very effective. The various IDE for python is spider, pycharm, atom, jupyter notebook to name a few. The ide used in this program is jupyter notebook by Anaconda.

6.2 Detailed Implementation

6.2.1 Import the required libraries

The provided Python code imports essential libraries for data science and machine learning. It includes NumPy and Pandas for data manipulation, Matplotlib and Seaborn for visualization, and Scikit-Learn for machine learning tasks. The %matplotlib inline command ensures inline plotting in Jupyter notebooks. Specifically, the code imports modules for data splitting, linear regression, random forest regression, feature scaling, and model evaluation. This setup suggests the code is likely part of a regression analysis or prediction task using machine learning, with the subsequent code expected to include data processing, model training, and evaluation steps.

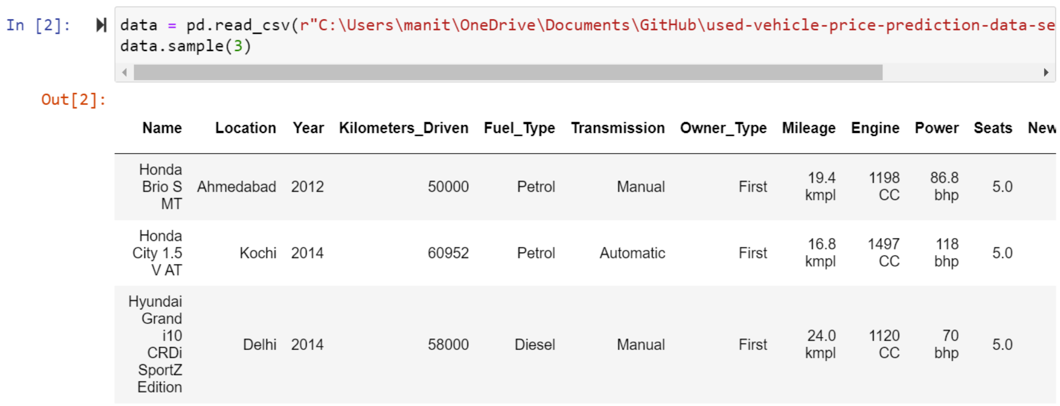


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
```

Figure 6.1: *Import the required libraries*

6.2.2 Load the Dataset

In the provided Python code, the pandas library is utilized to import a dataset from a CSV file. The CSV file, located at "C:\Users\manit\OneDrive\Documents\GitHub\used-vehicle-price-prediction-data-science\dataset.csv," is read into a pandas DataFrame named 'data'. This DataFrame likely contains information related to used vehicle prices for a data science project. The subsequent line, 'data.sample(3),' randomly selects and displays 3 rows from the DataFrame, offering a glimpse into the dataset's structure and enabling an initial assessment of the data's characteristics. This approach is common in data exploration and aids in understanding the dataset before further analysis or modelling.



```
In [2]:  data = pd.read_csv(r"C:\Users\manit\OneDrive\Documents\GitHub\used-vehicle-price-prediction-data-se
         data.sample(3)
```

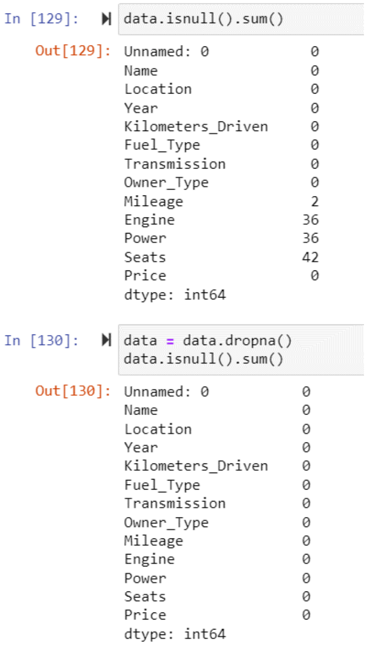
Out[2]:

Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New
Honda Brio S MT	Ahmedabad	2012	50000	Petrol	Manual	First	19.4 kmpl	1198 CC	86.8 bhp	5.0	
Honda City 1.5 V AT	Kochi	2014	60952	Petrol	Automatic	First	16.8 kmpl	1497 CC	118 bhp	5.0	
Hyundai Grand i10 CRDi SportZ Edition	Delhi	2014	58000	Diesel	Manual	First	24.0 kmpl	1120 CC	70 bhp	5.0	

Figure 6.2: *Load the dataset*

6.2.3 Data Cleaning

The code initially evaluates the presence of null values in each column of the dataset (data) using the isnull().sum() method. Subsequently, it employs the dropna() method to eliminate rows containing any null values, thereby updating the dataset. The final step involves reassessing the dataset to verify the successful removal of null values. This meticulous process ensures the dataset is thoroughly cleaned, free from any missing values, and ready for further analytical exploration or machine learning model development.



```
In [129]:  data.isnull().sum()

Out[129]:  Unnamed: 0            0
           Name                  0
           Location              0
           Year                  0
           Kilometers_Driven     0
           Fuel_Type             0
           Transmission          0
           Owner_Type            0
           Mileage               2
           Engine               36
           Power                36
           Seats                42
           Price                 0
           dtype: int64

In [130]:  data = data.dropna()
           data.isnull().sum()

Out[130]:  Unnamed: 0           0
           Name                 0
           Location             0
           Year                 0
           Kilometers_Driven    0
           Fuel_Type            0
           Transmission         0
           Owner_Type           0
           Mileage              0
           Engine               0
           Power                0
           Seats                0
           Price                0
           dtype: int64
```

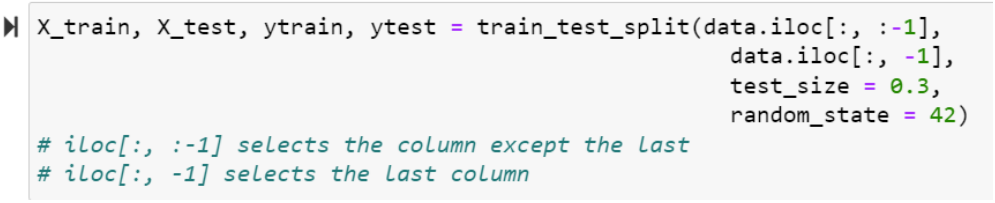
Figure 6.3: *data cleaning step*

6.2.4 Test-Train Split

Here the code is for splitting a dataset into training and testing sets using the train_test_split function from the Scikit-Learn library. Here's a breakdown of the code:

- train_test_split: This function is part of Scikit-Learn and is commonly used to split a dataset into training and testing sets for machine learning tasks.
- data.iloc[:, :-1]: Selects all columns except the last one. This is typically done to separate the features (independent variables) from the target variable.
- data.iloc[:, -1]: Selects the last column, representing the target variable (dependent variable).
- test_size=0.3: Specifies that 30% of the data will be used as the testing set, while the remaining 70% will be used for training.
- random_state=42: Sets a seed for randomization, ensuring reproducibility. The same seed will produce the same split each time the code is run.
- X_train, X_test, y_train, y_test: These variables store the resulting training and testing sets. X_train and X_test contain the feature sets for training and testing, while y_train and y_test contain the corresponding target variable values.

In summary, the code splits the dataset into training and testing sets, where 70% of the data is used for training machine learning models (X_train and y_train), and 30% is reserved for testing the model's accuracy (X_test and y_test). The iloc indexing is employed to separate features and the target variable appropriately.



```
X_train, X_test, ytrain, ytest = train_test_split(data.iloc[:, :-1],
                                                   data.iloc[:, -1],
                                                   test_size = 0.3,
                                                   random_state = 42)
# iloc[:, :-1] selects the column except the last
# iloc[:, -1] selects the last column
```

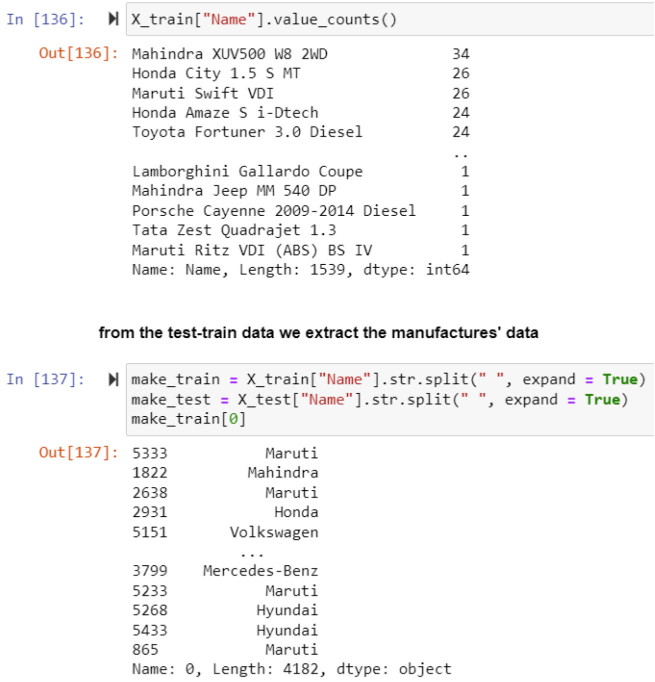
Figure 6.4: *Test-Train Split*

6.2.5 Pre-processing of the data set

In machine learning, dataset pre-processing enhances data quality for model training. Steps include handling missing values, scaling numerical features, encoding categorical variables, addressing imbalanced data, and creating meaningful features. These processes optimize the dataset for a chosen algorithm, improving model performance and predictive accuracy.

6.2.5.1 Manufacture's column pre-processing

The "Name" column in both the training and testing datasets is intelligently processed. By splitting the entries based on spaces, new columns are created to extract and store the manufacturer names. This enriches the datasets, providing specific information about car manufacturers. The resulting "Manufacturer" columns offer a valuable addition to the data, potentially influencing subsequent analysis or machine learning model development. The sample display of the updated training dataset demonstrates the successful integration of the manufacturer information, fostering a more comprehensive understanding of the dataset's characteristics.



```
In [136]:   X_train["Name"].value_counts()

Out[136]:   Mahindra XUV500 W8 2WD             34
            Honda City 1.5 S MT                26
            Maruti Swift VDI                   26
            Honda Amaze S i-Dtech              24
            Toyota Fortuner 3.0 Diesel         24
                                               ..
            Lamborghini Gallardo Coupe          1
            Mahindra Jeep MM 540 DP             1
            Porsche Cayenne 2009-2014 Diesel    1
            Tata Zest Quadrajet 1.3             1
            Maruti Ritz VDI (ABS) BS IV         1
            Name: Name, Length: 1539, dtype: int64
```

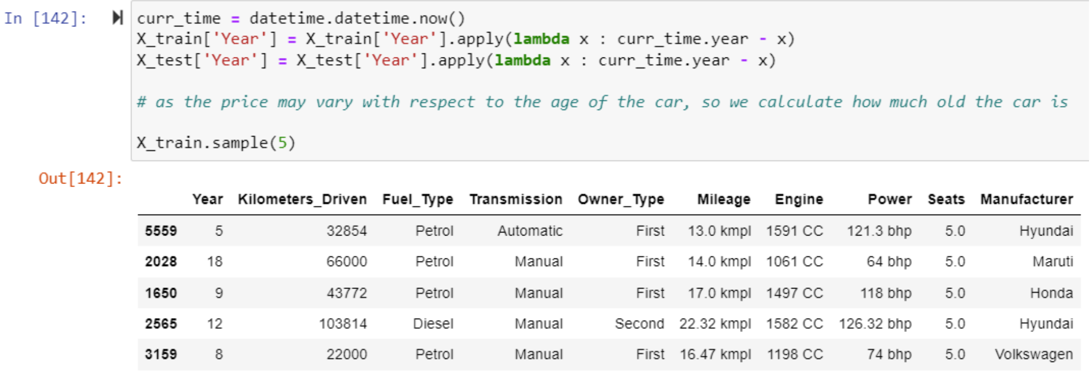
from the test-train data we extract the manufactures' data

```
In [137]:   make_train = X_train["Name"].str.split(" ", expand = True)
            make_test = X_test["Name"].str.split(" ", expand = True)
            make_train[0]

Out[137]:   5333           Maruti
            1822         Mahindra
            2638           Maruti
            2931            Honda
            5151       Volkswagen
                          ...
            3799    Mercedes-Benz
            5233           Maruti
            5268          Hyundai
            5433          Hyundai
            865            Maruti
            Name: 0, Length: 4182, dtype: object
```

Figure 6.5: *manufacture's column pre-processing*

6.2.5.2 Year column pre-processing

The code enhances the dataset by deriving the age of each car from the "Year" column. It calculates the age by subtracting the manufacturing year from the current year. The resulting age feature can be insightful for analysing how the age of a car correlates with its resale price. The sample display of the updated training dataset showcases the integration of the calculated car age, offering a preview of the dataset's augmented information. This transformation contributes to a more comprehensive understanding of the dataset's characteristics, particularly in relation to the impact of a car's age on its resale value.



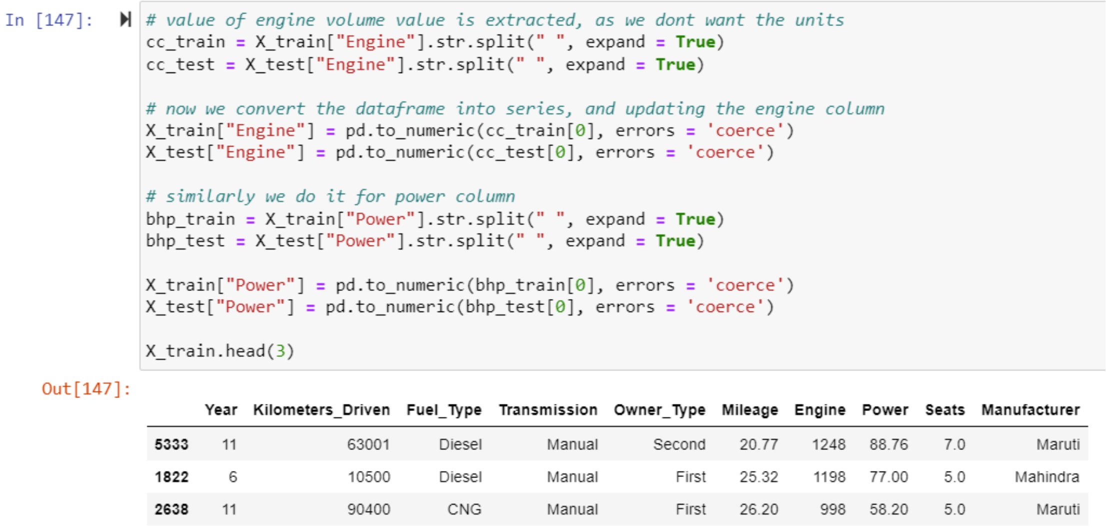
```
In [142]:  curr_time = datetime.datetime.now()
           X_train['Year'] = X_train['Year'].apply(lambda x : curr_time.year - x)
           X_test['Year'] = X_test['Year'].apply(lambda x : curr_time.year - x)

           # as the price may vary with respect to the age of the car, so we calculate how much old the car is

           X_train.sample(5)
Out[142]:
```

	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Manufacturer
5559	5	32854	Petrol	Automatic	First	13.0 kmpl	1591 CC	121.3 bhp	5.0	Hyundai
2028	18	66000	Petrol	Manual	First	14.0 kmpl	1061 CC	64 bhp	5.0	Maruti
1650	9	43772	Petrol	Manual	First	17.0 kmpl	1497 CC	118 bhp	5.0	Honda
2565	12	103814	Diesel	Manual	Second	22.32 kmpl	1582 CC	126.32 bhp	5.0	Hyundai
3159	8	22000	Petrol	Manual	First	16.47 kmpl	1198 CC	74 bhp	5.0	Volkswagen

Figure 6.6: *year column pre-processing*

6.2.5.3 Engine and power column pre-processing

The code focuses on refining the "Engine" and "Power" columns in the dataset. It systematically extracts the numerical values from these columns, excluding the accompanying units. The process involves splitting the entries, creating DataFrames for processing, and subsequently converting the relevant parts into numeric format. Non-numeric entries are handled gracefully by coercing them to NaN (Not a Number). This systematic conversion ensures that the columns now exclusively contain numeric values, a crucial step for accurate analysis or modeling. The displayed sample of the updated training dataset provides a glimpse of the refined structure, showcasing the numeric representation of "Engine" and "Power."



```
In [147]:  # value of engine volume value is extracted, as we dont want the units
           cc_train = X_train["Engine"].str.split(" ", expand = True)
           cc_test = X_test["Engine"].str.split(" ", expand = True)

           # now we convert the dataframe into series, and updating the engine column
           X_train["Engine"] = pd.to_numeric(cc_train[0], errors = 'coerce')
           X_test["Engine"] = pd.to_numeric(cc_test[0], errors = 'coerce')

           # similarly we do it for power column
           bhp_train = X_train["Power"].str.split(" ", expand = True)
           bhp_test = X_test["Power"].str.split(" ", expand = True)

           X_train["Power"] = pd.to_numeric(bhp_train[0], errors = 'coerce')
           X_test["Power"] = pd.to_numeric(bhp_test[0], errors = 'coerce')

           X_train.head(3)
```

Out[147]:

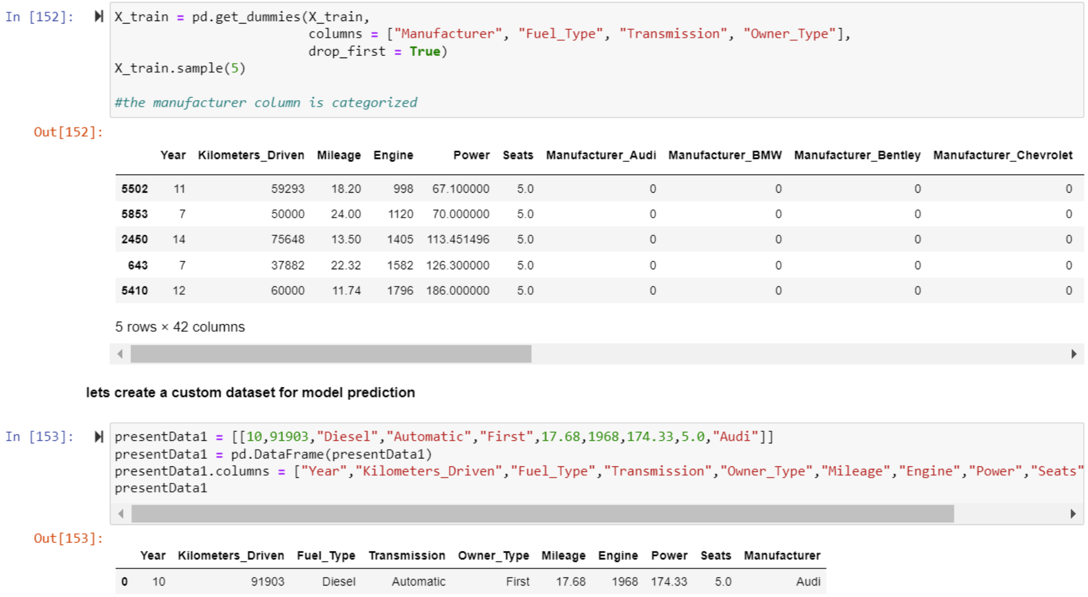
	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Manufacturer
5333	11	63001	Diesel	Manual	Second	20.77	1248	88.76	7.0	Maruti
1822	6	10500	Diesel	Manual	First	25.32	1198	77.00	5.0	Mahindra
2638	11	90400	CNG	Manual	First	26.20	998	58.20	5.0	Maruti

Figure 6.7: *engine and power column pre-processing*

6.2.6 Processing the training dataset

The provided code employs one-hot encoding to transform categorical columns in the training dataset (X_train). Categorical variables such as "Manufacturer," "Fuel_Type," "Transmission," and "Owner_Type" are converted into binary columns, representing the presence or absence of each category. This encoding facilitates the inclusion of categorical information in machine learning models. The sample display of the updated training dataset illustrates how the one-hot encoding process has augmented the original dataset with additional binary columns.

Additionally, a sample DataFrame (presentData1) is created to showcase the expected format for new data points during model prediction. This ensures consistency in the encoding process when dealing with unseen data. One-hot encoding is a common pre-processing step in machine learning, enabling models to work effectively with categorical data.



In [152]:

```
X_train = pd.get_dummies(X_train,
                         columns = ["Manufacturer", "Fuel_Type", "Transmission", "Owner_Type"],
                         drop_first = True)
X_train.sample(5)

#the manufacturer column is categorized
```

Out[152]:

	Year	Kilometers_Driven	Mileage	Engine	Power	Seats	Manufacturer_Audi	Manufacturer_BMW	Manufacturer_Bentley	Manufacturer_Chevrolet
5502	11	59293	18.20	998	67.100000	5.0	0	0	0	0
5853	7	50000	24.00	1120	70.000000	5.0	0	0	0	0
2450	14	75648	13.50	1405	113.451496	5.0	0	0	0	0
643	7	37882	22.32	1582	126.300000	5.0	0	0	0	0
5410	12	60000	11.74	1796	186.000000	5.0	0	0	0	0

5 rows × 42 columns

lets create a custom dataset for model prediction

In [153]:

```
presentData1 = [[10,91903,"Diesel","Automatic","First",17.68,1968,174.33,5.0,"Audi"]]
presentData1 = pd.DataFrame(presentData1)
presentData1.columns = ["Year","Kilometers_Driven","Fuel_Type","Transmission","Owner_Type","Mileage","Engine","Power","Seats"
presentData1
```

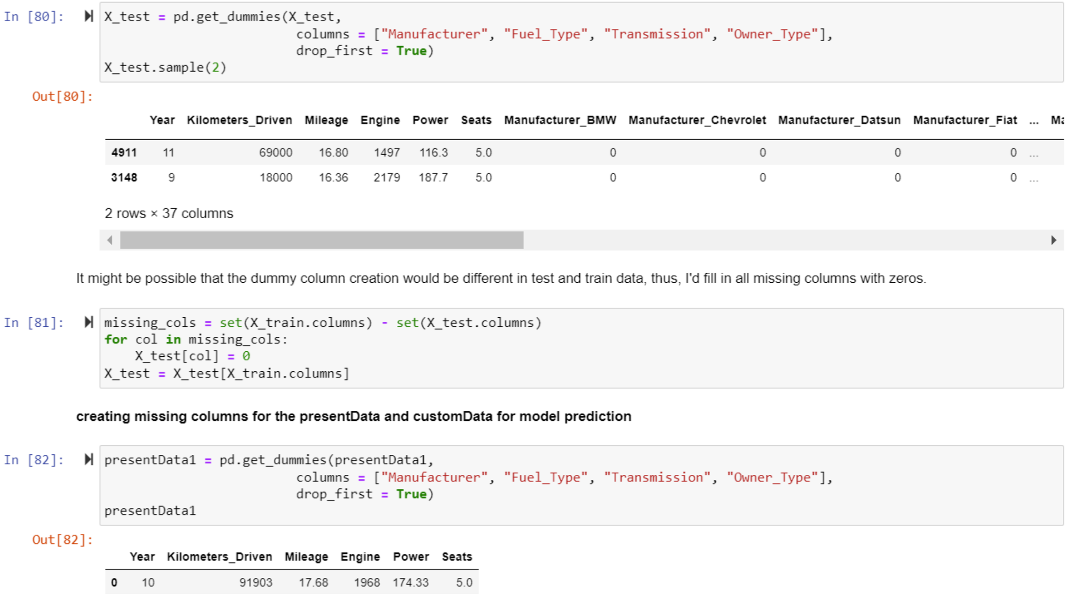
Out[153]:

	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Manufacturer
0	10	91903	Diesel	Automatic	First	17.68	1968	174.33	5.0	Audi

Figure 6.8: *train data processing by one-hot encoding*

6.2.7 Processing the test dataset

The provided code orchestrates a thorough data pre-processing endeavour aimed at ensuring the seamless integration of the test dataset (X_test) with its training counterpart (X_train) for optimal machine learning model deployment. Through one-hot encoding, the categorical columns in the test set undergo a transformation that mirrors the encoding schema applied during the training phase. Beyond this, the code addresses potential structural discrepancies by identifying and rectifying missing columns in the test set. For each absent column, a corresponding new column is introduced, bolstering the dataset's alignment with the training set. This meticulous process extends its reach to a sample DataFrame (presentData1), laying the foundation for uniform treatment of novel data points during model predictions. Such assiduous preparation is pivotal, not only in ensuring model compatibility but also in fortifying its resilience and predictive accuracy across diverse datasets and unforeseen scenarios.



```
In [80]:  X_test = pd.get_dummies(X_test,
                                columns = ["Manufacturer", "Fuel_Type", "Transmission", "Owner_Type"],
                                drop_first = True)
          X_test.sample(2)

Out[80]:
        Year  Kilometers_Driven  Mileage  Engine  Power  Seats  Manufacturer_BMW  Manufacturer_Chevrolet  Manufacturer_Datsun  Manufacturer_Fiat  ...  Ma
 4911     11              69000    16.80    1497  116.3    5.0                 0                       0                    0                  0  ...
 3148      9              18000    16.36    2179  187.7    5.0                 0                       0                    0                  0  ...

2 rows × 37 columns
```

It might be possible that the dummy column creation would be different in test and train data, thus, I'd fill in all missing columns with zeros.

```
In [81]:  missing_cols = set(X_train.columns) - set(X_test.columns)
          for col in missing_cols:
              X_test[col] = 0
          X_test = X_test[X_train.columns]
```

creating missing columns for the presentData and customData for model prediction

```
In [82]:  presentData1 = pd.get_dummies(presentData1,
                                columns = ["Manufacturer", "Fuel_Type", "Transmission", "Owner_Type"],
                                drop_first = True)
          presentData1

Out[82]:
     Year  Kilometers_Driven  Mileage  Engine   Power  Seats
 0     10              91903    17.68    1968  174.33    5.0
```

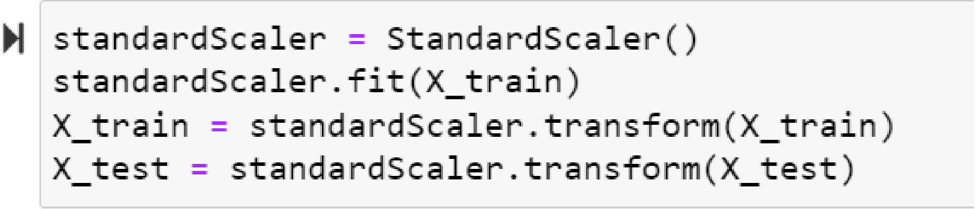
Figure 6.9: *test data processing by one-hot encoding*

6.2.8 Training and predicting

Standardizing data is a pre-processing technique in machine learning used to bring the features of a dataset to a common scale. This is important because it ensures that all variables contribute equally to the model, preventing any particular feature from dominating others. The process involves two key steps:

- **Calculating Mean and Standard Deviation**: The mean and standard deviation of each feature in the dataset are calculated. The mean represents the average value of the feature, and the standard deviation measures how spread out the values are.
- **Transforming Data**: Each feature in the dataset is then transformed by subtracting the mean and dividing by the standard deviation. This ensures that all features have a comparable scale, typically centred around 0 and with a standard deviation of 1.

By standardizing the data, the impact of outliers is reduced, and models can perform more reliably across various features. It's particularly useful when using algorithms that rely on distance measures or optimization techniques, contributing to the stability and accuracy of machine learning models during training and evaluation.



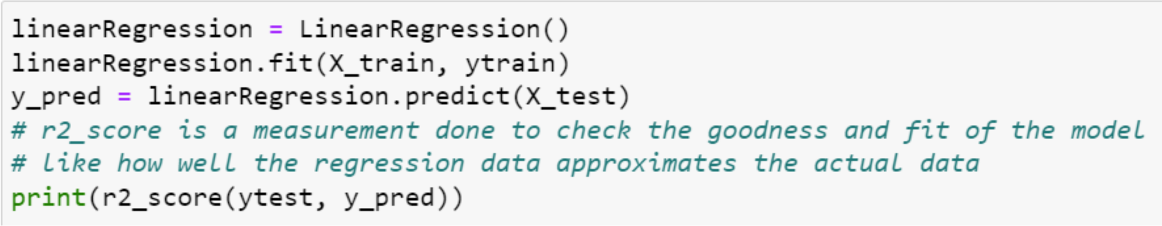
```
standardScaler = StandardScaler()
standardScaler.fit(X_train)
X_train = standardScaler.transform(X_train)
X_test = standardScaler.transform(X_test)
```

Figure 6.10: *standardizing the test-train dataset*

6.2.9 Training the Linear regression model

In the presented process, a Linear Regression model is instantiated, trained, and evaluated. The model is trained using standardized features from the training dataset (X_train) and corresponding target values (y_train). Once trained, the model predicts target values for the standardized test dataset (X_test). The R2 score is then calculated to assess the goodness of fit, providing a measure of how well the model's predictions approximate the actual target values in the test set (y_test). The R2 score ranges from 0 to 1, where a higher score indicates a better fit.

Additionally, a scatter plot is generated to visually compare the predicted values (y_pred) against the true values in the test set. This graphical representation aids in intuitively understanding the alignment between the model's predictions and the actual outcomes, offering insights into the model's predictive performance.



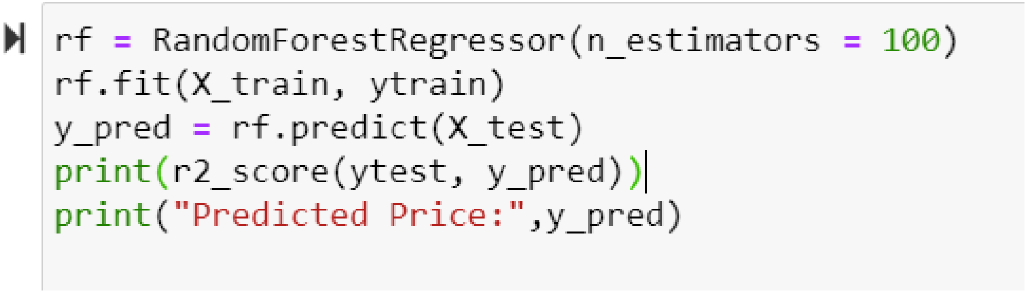
```
linearRegression = LinearRegression()
linearRegression.fit(X_train, ytrain)
y_pred = linearRegression.predict(X_test)
# r2_score is a measurement done to check the goodness and fit of the model
# like how well the regression data approximates the actual data
print(r2_score(ytest, y_pred))
```

Figure 6.11: *Training the Linear regression model*

6.2.10 Training the Random Forest model

In this segment, a Random Forest Regressor model is employed for predictive modelling. The model is initialized with 100 decision trees and trained using the standardized training data (X_train) and corresponding target values (y_train). Subsequently, predictions (y_pred) are generated for the target variable in the standardized test dataset (X_test). The R2 score, a measure of goodness of fit, is then calculated to assess how well the model approximates the actual target values in the test set (y_test). A higher R2 score indicates a better fit.

Additionally, a scatter plot is generated to visually compare the predicted values against the true values in the test set, providing an intuitive representation of the model's predictive accuracy and its alignment with the actual outcomes. This analysis serves to evaluate and understand the performance of the Random Forest model in predicting the target variable based on the given features.



```
rf = RandomForestRegressor(n_estimators = 100)
rf.fit(X_train, ytrain)
y_pred = rf.predict(X_test)
print(r2_score(ytest, y_pred))
print("Predicted Price:",y_pred)
```

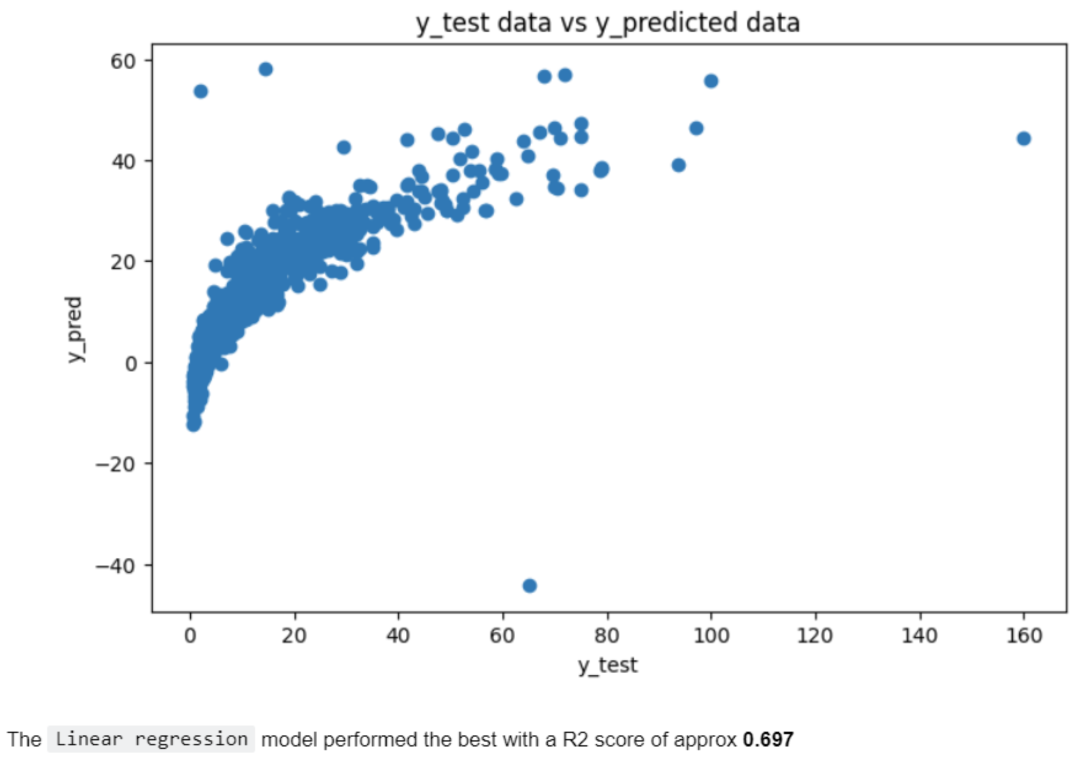
Figure 6.12: *Training the Random Forest model*

Chapter 7

RESULTS AND SNAPSHOTS

7.1 R2 score of linear regression model

The Linear Regression model achieved an R2 score of approximately 0.698, signifying its ability to explain around 69.8% of the variance in the test set of used car prices. This score serves as a measure of the model's goodness of fit, indicating how well it approximates the actual data. The scatter plot visually represents the relationship between the predicted prices (y_pred) and the actual prices (y_test). Overall, this score suggests a reasonable performance of the Linear Regression model in predicting used car prices based on the standardized features.

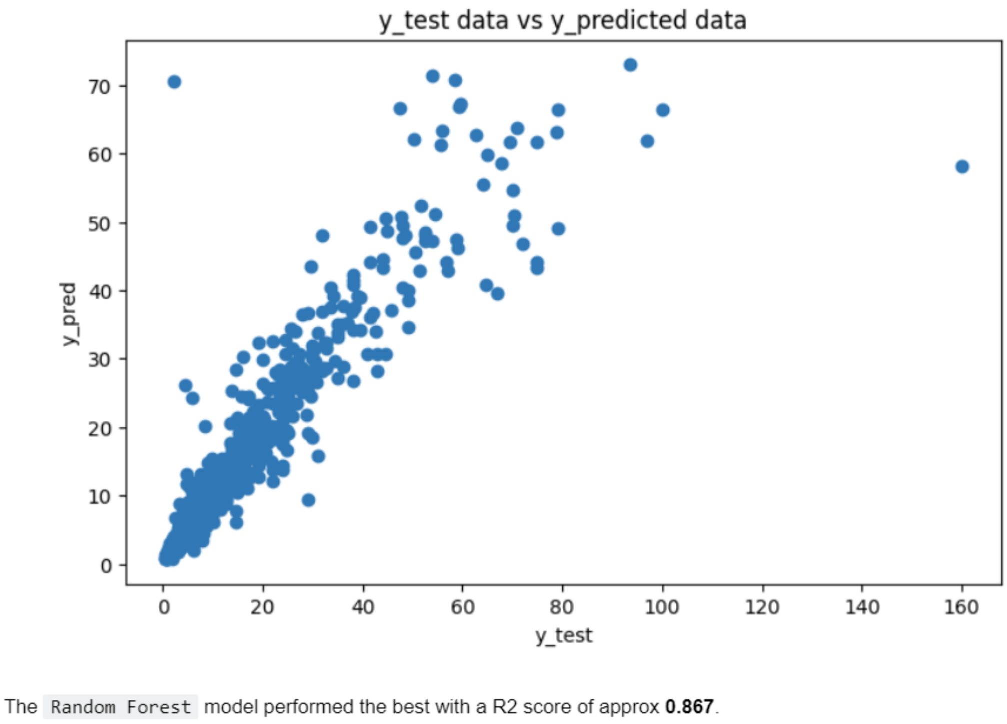


The `Linear regression` model performed the best with a R2 score of approx **0.697**

Figure7.1: *R2 score plot of linear regression model*

7.2 R2 score of random forest model

The Random Forest model demonstrated superior performance with an R2 score of approximately 0.868. This score signifies that the Random Forest model explains around 86.8% of the variance in the test set of used car prices. The scatter plot visually illustrates the relationship between the predicted prices (y_pred) and the actual prices (y_test). In summary, the Random Forest model outperformed the Linear Regression model, suggesting a higher accuracy in predicting used car prices based on the given features.



The `Random Forest` model performed the best with a R2 score of approx **0.867**.

Figure7.1: *R2 score plot of random forest model*

Chapter 8

CONCLUSION AND FUTURE ENHANCEMENTS

8.1 Conclusion

The project focused on predicting used car prices, a comprehensive data exploration, data pre-processing, model building and predicting steps were undertaken. The two phases involved training and evaluating machine learning models, specifically Linear Regression and Random Forest Regression. These models were assessed based on the R2 score, a metric gauging the goodness of fit. Results indicated that the Linear Regression model achieved an R2 score of approximately 69.7%, while the Random Forest Regression model exhibited superior performance, yielding a higher R2 score of around 86.7%. Therefore, the project findings suggest that the Random Forest Regression model is more effective in predicting used car prices compared to Linear Regression. The models, particularly the Random Forest variant, form a robust foundation for predicting prices based on the selected features. While further optimization could potentially enhance predictive accuracy, the consistently strong performance of the Random Forest model makes it the preferred choice for this specific prediction task. Overall, the project successfully navigated data challenges, explored critical features, and implemented machine learning models to provide valuable insights into predicting used car prices.

8.2 Future Enhancements

- **Developing a mobile app**: Creating a user-friendly mobile app that allows users to input car details and receive an instant price prediction on the go.
- **Integration with online platforms**: Integrating the price prediction model with online car buying and selling platforms can provide sellers with realistic price estimates and buyers with insights into market trends.
- **Localized Price Trends**: Incorporate regional or city-wise price trends to provide more accurate predictions based on the local market conditions.
- **Historical Price Analysis:** Provide users with historical price trends for specific car models. This feature could help users make more informed decisions by understanding how prices have changed over time.

REFERENCES

[1] https://ieeexplore.ieee.org/document/9995772

[2] https://www.academia.edu/77140370/Used_Car_Price_Prediction_using_Different_Machine_Learning_Algorithms

[3] https://towardsdatascience.com/predicting-used-car-prices-with-machine-learning-techniques-8a9d8313952

[4] https://scikit-learn.org/stable/modules/classes.html