CAR PRICE PREDICTION

A Project Report

Submitted in partial fulfilment of the Requirements for the award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

ABDUL RAAFAY KHAN
4018200

&

MOHAMMED RIYAAN ANSARI

4017983

Under the esteemed guidance of

MS. HINA MAHMOOD

DEPARTMENT OF INFORMATION TECHNOLOGY

Rizvi College of Arts Science and Commerce

(Affiliated to University of Mumbai)

Mumbai, PIN CODE-400050

MAHARASHTRA

2022-23

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PNR No:……………………. Roll no**:**

1. Name of the Student

ABDUL RAAFAY KHAN &

MOHAMMED RIYAAN ANSARI

2. Title of the Project

CAR PRICE PREDICTION

3. Name of the Guide

HINA MAHMOOD

Signature of the Student

Date: …………………

Signature of the Guide

Date: …………………

Signature of the **Coordinator**

Date: …………………

RIZVI COLLEGE OF ARTS SCIENCE AND COMMERCE

(Affiliated to University of Mumbai)

MUMBAI MAHARASHTRA-400050

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled, "**CAR PRICE PREDICTION** ", is a bonafide work of **ABDUL RAAFAY KHAN & MOHAMMED RIYAAN ANSARI** bearing Seat nos. **4018200 & 4017983** submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide **Coordinator**

External Examiner

Date:………… **College Seal**

Abstract

The car price prediction model is a machine learning-based project that aims to predict car prices using various features such as engine size, curb weight, horsepower, car height, car width, wheelbase, car length, and bore ratio. The implementation involves data collection, data preprocessing, data splitting, model training, model evaluation, deployment, and testing. The model has practical implications in the automotive industry, assisting in car valuation, pricing, and negotiations. The project has significant potential for future development, research, and integration into various applications. Despite some limitations, the car price prediction model is a valuable tool for making informed decisions about car prices, benefiting car dealerships, buyers, and sellers.

ACKNOWLEDGEMENT

I owe special thanks to the Department of Information Technology of **Rizvi College of Arts Science and Commerce** for giving me a chance to prepare this project dissertation. I thank the Principal, **Professor Ashfaq Ahmad khan** for his leadership and management. I thank the Coordinator and Head of the Department **Professor Rafat Khan** for providing us the required facilities and guidance throughout the course which culminated into this thesis. Last and not the least to the project guide this semester**- Professor Hina Mahmood**. Deep gratitude to the staff and faculty of Rizvi College for their help and support. And also my beloved **Parents** for their infinite support and love.

Abdul Raafay Khan &

Mohammed Riyaan

Ansari

DECLARATION

Content I hereby declare that the project entitled, “Human Following Robot” done at **Rizvi College of Arts Science and Commerce**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the awardof degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

Contents

CHAPTER 1

1.1 Introduction

Determining whether the listed price of a used car is a challenging task, due to the many factors that drivea used vehicle's price on the market. The focus of this project is developing machine learning models thatcan accurately predict the price of a used car based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the sale prices of differentmakes and models . We will compare the performance of various machine learning algorithms like LinearRegression, Ridge Regression, Lasso Regression, Elastic Net, Decision Tree Regressor and choose the best out of it. Depending on various parameters we will determine the price of the car. Regression Algorithms are used because they provide us with continuous value as an output and not a categorized value because of which it will be possible to predict the actual price a car rather than the price range of a car. User Interface has also been developed which acquires input from any user and displays the Price of acar according to user's inputs.

1.2 Background

In this project we propose a prediction system for predicting car prices by using various factors such asengine, car body, horse power, car width and height, etc.

We implemented a machine learning model that predicts the price of a vehicle based on the preferenceof the costumer like, car model, fuel type, etc. That data will be visualized in an interactive Power BI dashboard showcasing average price of the total cars available showcasing the relation between actualmarket price of the car and the price predicted by the machine.

1.3 Objectives

The main objectives of the car price prediction model project are:

1. To develop a machine learning-based model that can accurately predict car prices.
2. To collect and preprocess car data from various sources.
3. To train and evaluate the model using suitable machine learning algorithms.
4. To deploy the model into various applications.
5. To provide a useful tool for car dealerships, buyers, and sellers to make informed decisions about car prices.

6. To explore the potential for future development and research of the car price prediction model.

1.4 Purpose, Scope and Applicability

Purpose

The purpose of this project aims to predict the Price of a used Car by taking inputs such as Companyname, it's Model name, Year of Purchase, and other parameters.

Scope

In future this machine learning model may bind with various website which can provide real time data forprice prediction. Also we may add large historical data of car price which can help to improve accuracy ofthe machine learning model. We can build an android app as user interface for interacting with user. For better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates and train on clusters of data rather than the whole dataset.

Applicability

The increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction systemwhich effectively determines the worthiness of the car using a variety of features. The proposed systemwill help to determine the accurate price of used car price prediction.

Organization of Reports

Chapter 1: is about the introduction of our project where we have given clear insights about ourproject domain and other related concepts.

Chapter 2: specifies about survey of technologies where all different existing methods and modelsare examined.

Chapter 3: Includes the need to make the project along with the planning phase

Chapter 4: Describes the basic connectivity of the module.

Chapter 5: Contains the code of the system along with different types of tests carried out.

Chapter 6: Shows the result of the test carried out.

Chapter 7: Contains a summary of the project, its limitation.

CHAPTER 2

2.1 SURVEY OF TECHNOLOGIES

2.1.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Libraries:

Pandas

Pandas is a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas has grown into one of the most popular Pythonlibraries. It has an extremely active community of contributors.

Numpy

NumPy is a Python library used for working with arrays. It also has functions for working in domain oflinear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

Matpodlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations inPython. Matplotlib makes easy things easy and hard things possible.
Create publication quality plots.

Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides aselection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

2.1.2 Microsoft Excel(Dataset)

Excel definition: a software program created by Microsoft that uses spreadsheets to organize numbers and data with formulas and functions. Excel analysis is ubiquitous around the world and used by businesses of all sizes to perform financial analysis.

2.1.3 Visualization Tool:

Microsoft Power BI

Power BI is a technology-driven business intelligence tool provided by Microsoft for analyzing and visualizing raw data to present actionable information. It combines business analytics, data visualization, and best practices that help an organization to make data-driven decisions. In February
2019, Gartner confirmed Microsoft as Leader in the "2019 Gartner Magic Quadrant for Analytics and Business Intelligence Platform" as a result of the capabilities of the Power BI platform.

Jupyter Notebook

Jupyter Notebook is an open-source, web-based interactive environment, which allows you to create and share documents that contain live code, mathematical equations, graphics, maps, plots, visualizations, and narrative text. It integrates with many programming languages like Python, PHP, R, C#, etc.

CHAPTER 3

3.1 Requirement and Analysis

3.1.1 Requirement Specification

Requirement analysis is a critical step in any software development project, including the car price prediction model project. It involves understanding and documenting the project's objectives, user needs, and system requirements. The requirements analysis phase for the car price prediction model project involves the following steps:

1. Defining the scope of the project: The first step is to define the project's scope, including the features and functionalities of the car price prediction model. This step helps to establish the project's objectives, goals, and limitations.

2. Identifying the users and stakeholders: The second step is to identify the users and stakeholders of the car price prediction model. This includes car dealerships, buyers, and sellers who require an accurate tool for car valuation, pricing, and negotiations.

3. Gathering and documenting the requirements: The third step is to gather and document the system requirements, including the data sources, data features, machine learning algorithms, model performance metrics, and deployment options. This step involves consulting with the users and stakeholders to understand their needs and expectations.

4. Analyzing and prioritizing the requirements: The fourth step is to analyze and prioritize the requirements based on their importance, feasibility, and urgency. This step helps to ensure that the project meets the users' needs and delivers value within the given time and budget constraints.

Analysis is another critical step in the car price prediction model project, which involves understanding the problem domain, the data sources, and the machine learning algorithms. The analysis phase for the car price prediction model project involves the following steps:

1. Understanding the problem domain: The first step is to understand the problem domain of car pricing, including the factors that influence car prices, such as the car's age, mileage, condition, and features.

2. Analyzing the data sources: The second step is to analyze the data sources, including the data quality, data completeness, and data consistency. This step helps to ensure that the data is suitable for machine learning algorithms and is free from errors and biases.

3. Selecting the machine learning algorithms: The third step is to select suitable machine learning algorithms based on the project's objectives and the data features. This step involves evaluating various machine learning algorithms, such as linear regression, decision trees, and neural networks.

4. Designing the model architecture: The fourth step is to design the model architecture, including the input features, the output variable, and the model complexity. This step helps to ensure that the model can learn from the data and make accurate predictions.

In summary, the requirement analysis and analysis phases are crucial steps in the car price prediction model project. These steps help to establish the project's objectives, user needs, and system requirements, and to understand the problem domain, the data sources, and the machine learning algorithms. By following a rigorous and systematic approach to requirement analysis and analysis, the project team can ensure that the car price prediction model meets the users' needs and delivers value.

3.1.2 Software and Hardware Requirements

Software Requirements

Jupyter Notebook (anaconda3)

Microsoft PowerBI

Microsoft Excel

Operating System: Windows 10

Tools: Jupyter Notebook, Kaggle, Web Browser (Google Chrome or Firefox)

Python Libraries: NumPy, Pandas, Sklearn, Matplotlib, Seaborn

Hardware Requirements

RAM: 4 GB or above

Storage: 30 to 50 GB

Processor: Any Processor above 500MHz

3.1.3 Planning and Scheduling

‘Project Planning and Scheduling’, though separate, are two sides of the same coin in project management.

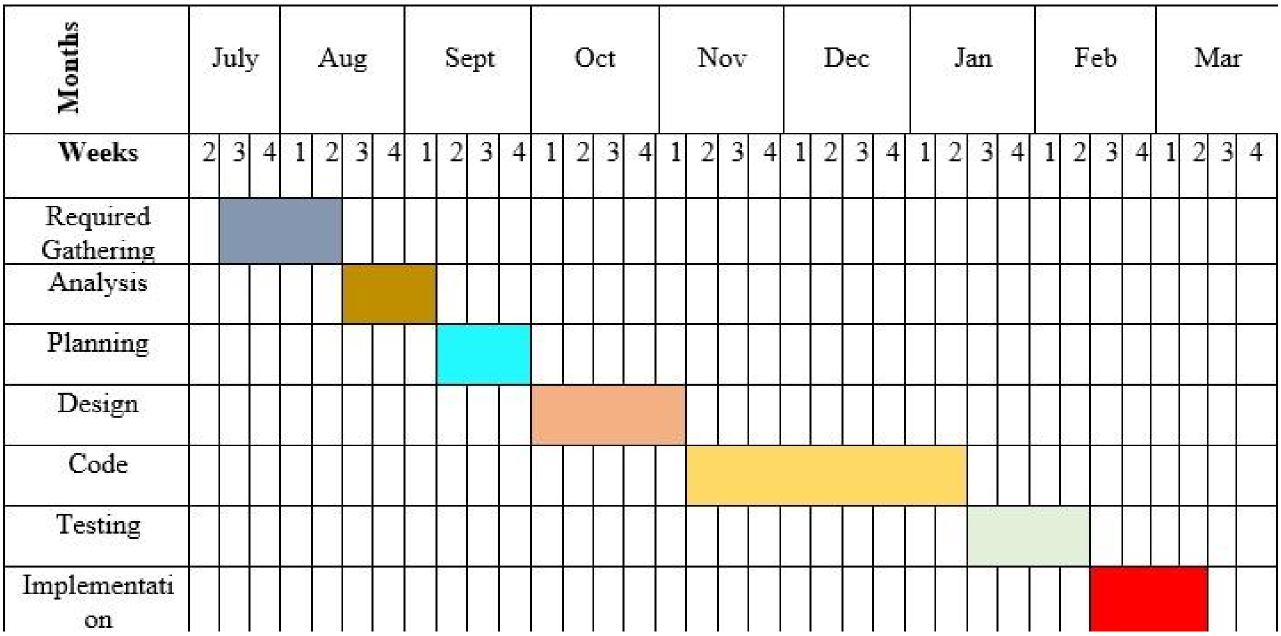
Fundamentally, ‘Project planning’ is all about choosing and designing effective policies and methodologies to attain project objectives. While ‘Project scheduling’ is a procedure of assigning tasks to get them completed by allocating appropriate resources within an estimated budget and time-frame.

Planning and scheduling are processes that turn project action plans for scope, time, cost, and quality into an operating timetable. Planning is largely concerned with choosing the necessary rules and procedures in order to fulfil the project's objectives.

Gantt-chart

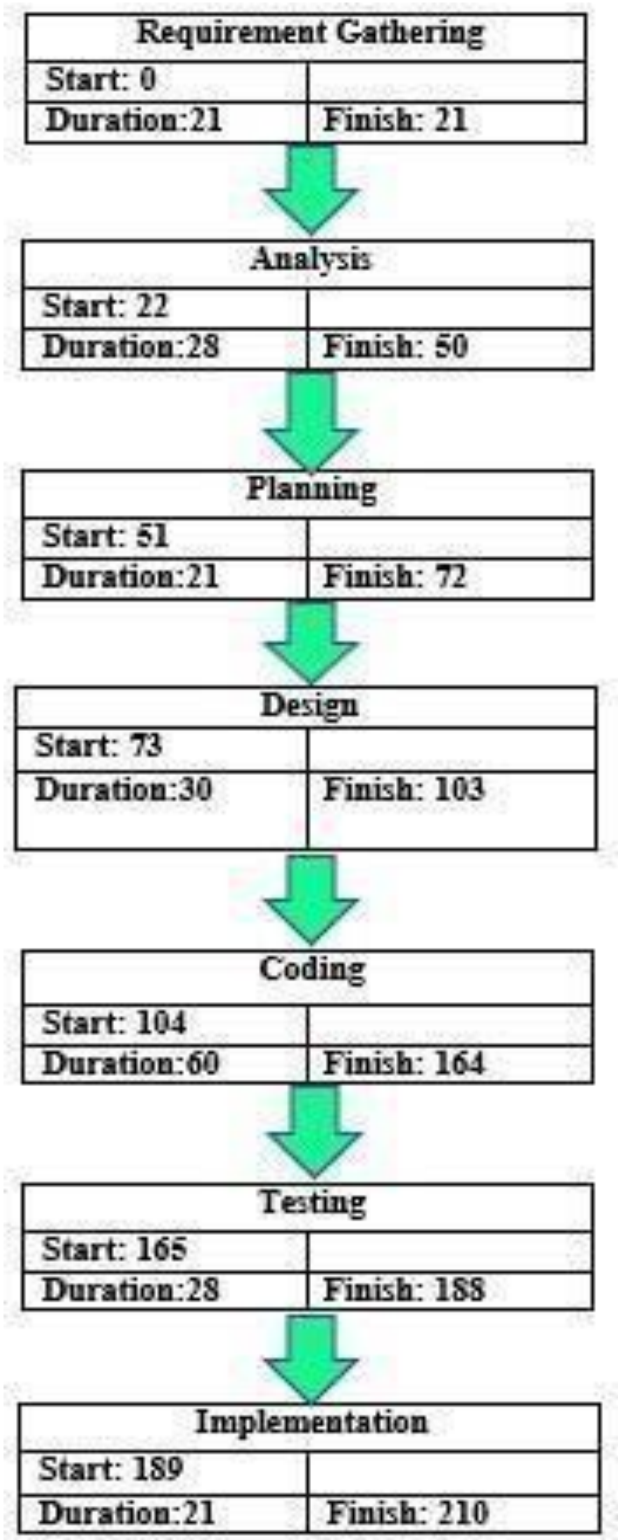
A Gantt chart is a sort of bar chart used to show project schedules. The tasks completed aredisplayed on the vertical axis of this graph, while the time intervals for each completed task are displayed on the horizontal axis. The length of each activity is displayed in the graph by the widthof the coloured horizontal bars.

Gantt charts show the beginning and ending dates of a project's terminal and summary elements. The project's work breakdown structure is made up of both the terminal and summarysections.



Pert chart

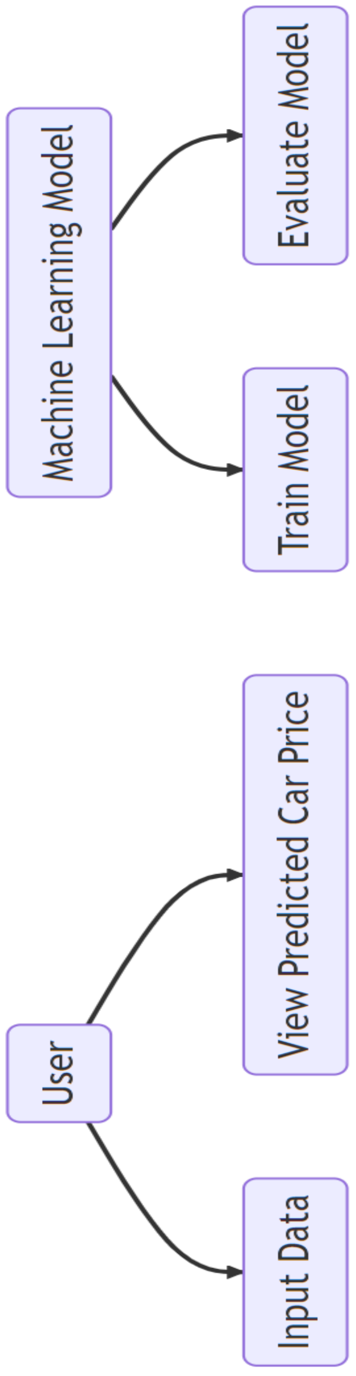
A graphical representation of a project's tasks, schedule, and timeframes is provided by PERT (Project Evaluation Review Technique) charts, which are comparable to Gantt charts in this regard.It is a tool for project management that shows the timeline of a project graphically. Project (or Program) Evaluation and Review Technique is referred to as PERT.



3.1.4 Conceptual models

Use Case Diagram:

The dynamic behaviour of the system is represented in this diagram. It simulates the duties, services, and operations needed by an application's system. It shows a system's high-level functionality and also describes how a user interacts with a system.

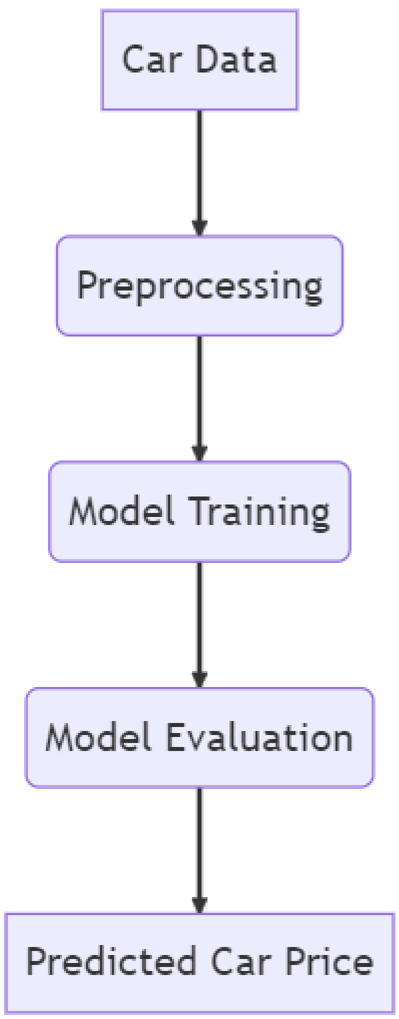


Data Flow Diagram

A common visual depiction of the information flow in the system is a (DFD).

For each system or process, the information flow is mapped out via a data flow diagram. Data flow diagrams can be as basic as hand-drawn process overviews or morecomplex, multi-level DFDs that gradually delve deeper into the handling of the data.

They can be used to analyse an existing system or model a new system.



CHAPTER 4

4.1 SYSTEM DESIGN

This Interactive Dashboard shows the relation between Number of cars per Carbody, ActualPrice v/s Predicted Price, Price v/s Engine Size, etc.

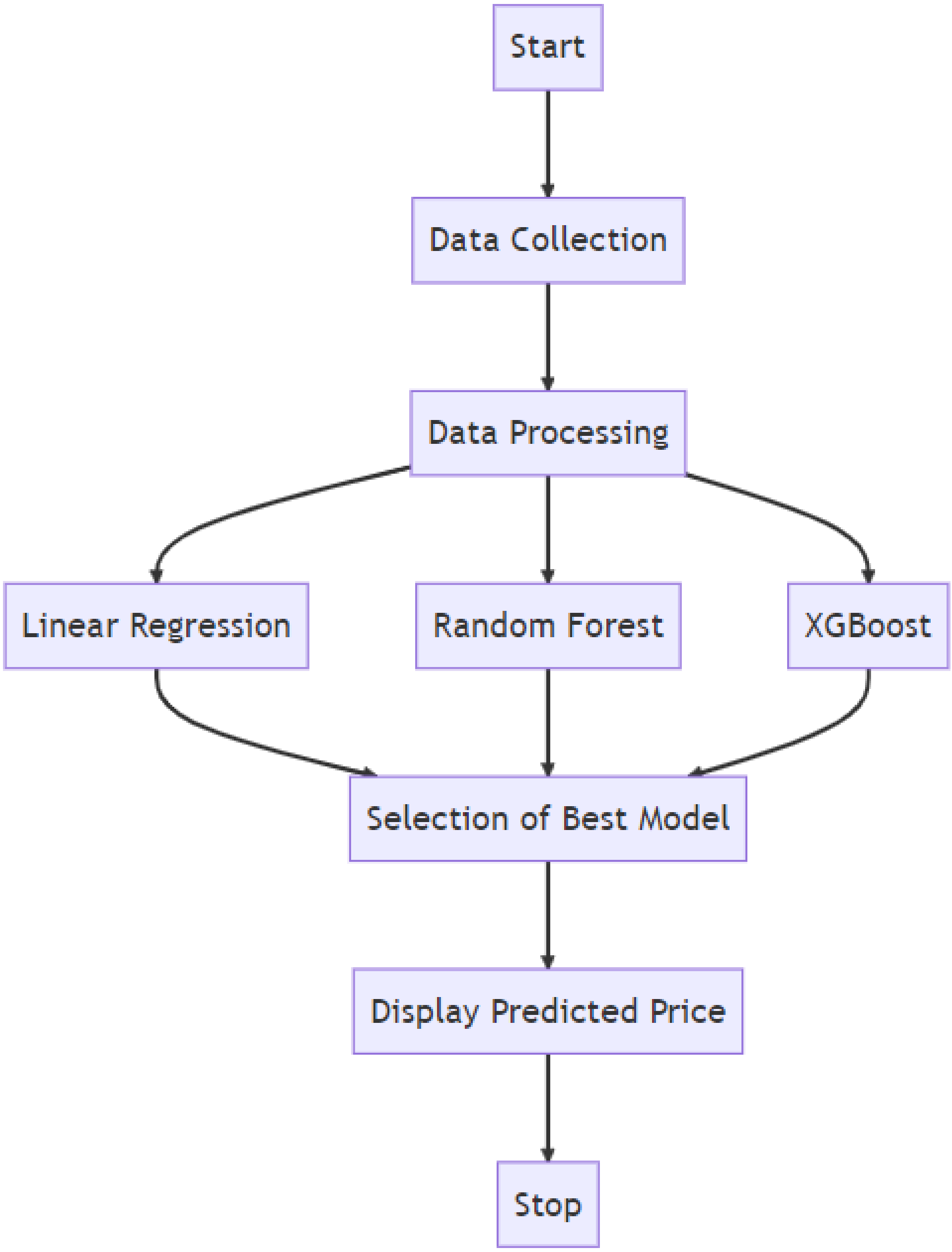
We will find these results by filtering data using factors like:

- Engine Type(DOHC, DOHCV, I, OHC, OHCF, OHCV)
  - In this Filter we can select the given type of engine from a dropdown box and listthe cars with that type of engine in the relation.

- Cylinder Number(Three, Four, Five, Six, Eight, Twelve)
  - In this filter we can select the number of cylinders from a dropdown and it willlist the cars accordingly.

- Fuel Type(Gas i.e. Petrol and Diesel)
  - This Filter has only two options Gas i.e. Petrol and Diesel.

These filtering options will make changes in the dashboard and show filtered results according toour preferences.


Start
Data Collection
Data Processing
Linear Regression
Random Forest
XGBoost
Selection of Best Model
Display Predicted Price
Stop

Reading the Data

```
(205, 26)
```

Out[31]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight	curbweight	enginetype
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	2823	ohcv
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	2337	ohc
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	2824	ohc

Converting Categorical Data into Numerical Data

In [51]:

```python
# Making categorical variables into numeric representation

new_raw_data = pd.get_dummies(raw_data, columns = ['carbody', 'aspiration','doornumber','drivewheel','enginelocation', 'fuelsystem', 'enginetype', 'fu
new_raw_data.head()
```

Out[51]:

	car_ID	symboling	CarName	wheelbase	carlength	carwidth	carheight	curbweight	cylindernumber	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm
0	1	3	alfa-romero giulia	88.6	168.8	64.1	48.8	2548	four	130	3.47	2.68	9.0	111	500
1	2	3	alfa-romero stelvio	88.6	168.8	64.1	48.8	2548	four	130	3.47	2.68	9.0	111	500
2	3	1	alfa-romero Quadrifoglio	94.5	171.2	65.5	52.4	2823	six	152	2.68	3.47	9.0	154	500
3	4	2	audi 100 ls	99.8	176.6	66.2	54.3	2337	four	109	3.19	3.40	10.0	102	550
4	5	2	audi 100ls	99.4	176.6	66.4	54.3	2824	five	136	3.19	3.40	8.0	115	550

In [133…

```python
# Making categorical variables into numeric representation

new_raw_data = pd.get_dummies(raw_data, columns = ['carbody', 'aspiration','doornumber','drivewheel','enginelocation', 'fuelsystem', 'enginetype', 'fu

new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'two'] = 2
new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'three'] = 3
new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'four'] = 4
new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'five'] = 5
new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'six'] = 6
new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'eight'] = 8
new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'twelve'] = 12
new_raw_data['cylindernumber'] = new_raw_data['cylindernumber'].astype(int)

warnings.simplefilter(action='ignore', category=FutureWarning)

new_raw_data.dtypes
```

```
C:\Users\pitsi\AppData\Roaming\Python\Python37\site-packages\pandas\core\indexing.py:670: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  iloc._setitem_with_indexer(indexer, value)
```

Activate Window
Go to Settings to activ

```
Out[133…  car_ID                       int32
          symboling                    int64
          CarName                     object
          wheelbase                  float64
          carlength                  float64
          carwidth                   float64
          carheight                  float64
          curbweight                   int64
          cylindernumber               int32
          enginesize                   int64
          boreratio                  float64
          stroke                     float64
          compressionratio           float64
          horsepower                   int64
          peakrpm                      int64
          citympg                      int64
          highwaympg                   int64
          price                      float64
          carbody_convertible          uint8
          carbody_hardtop              uint8
          carbody_hatchback            uint8
          carbody_sedan                uint8
          carbody_wagon                uint8
          aspiration_std               uint8
          aspiration_turbo             uint8
          doornumber_four              uint8
          doornumber_two               uint8
          drivewheel_4wd               uint8
          drivewheel_fwd               uint8
          drivewheel_rwd               uint8
          enginelocation_front         uint8
          enginelocation_rear          uint8
          fuelsystem_1bbl              uint8
          fuelsystem_2bbl              uint8
          fuelsystem_4bbl              uint8
          fuelsystem_idi               uint8
          fuelsystem_mfi               uint8
          fuelsystem_mpfi              uint8
          fuelsystem_spdi              uint8
          fuelsystem_spfi              uint8
          enginetype_dohc              uint8
          enginetype_dohcv             uint8
```

Reading New Converted Data

In [53]:
```
new_raw_data.head()
```

Out[53]:

	car_ID	symboling	CarName	wheelbase	carlength	carwidth	carheight	curbweight	cylindernumber	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm
0	1	3	alfa-romero giulia	88.6	168.8	64.1	48.8	2548	4	130	3.47	2.68	9.0	111	5000
1	2	3	alfa-romero stelvio	88.6	168.8	64.1	48.8	2548	4	130	3.47	2.68	9.0	111	5000
2	3	1	alfa-romero Quadrifoglio	94.5	171.2	65.5	52.4	2823	6	152	2.68	3.47	9.0	154	5000
3	4	2	audi 100 ls	99.8	176.6	66.2	54.3	2337	4	109	3.19	3.40	10.0	102	5500
4	5	2	audi 100ls	99.4	176.6	66.4	54.3	2824	5	136	3.19	3.40	8.0	115	5500

Trying Regression with less variables

In [96]:
```
final_fi['Variable'][0:10].values
```

Out[96]:
```
array(['curbweight', 'carheight', 'wheelbase', 'horsepower', 'carlength',
       'boreratio', 'enginesize', 'compressionratio', 'highwaympg',
       'peakrpm'], dtype=object)
```

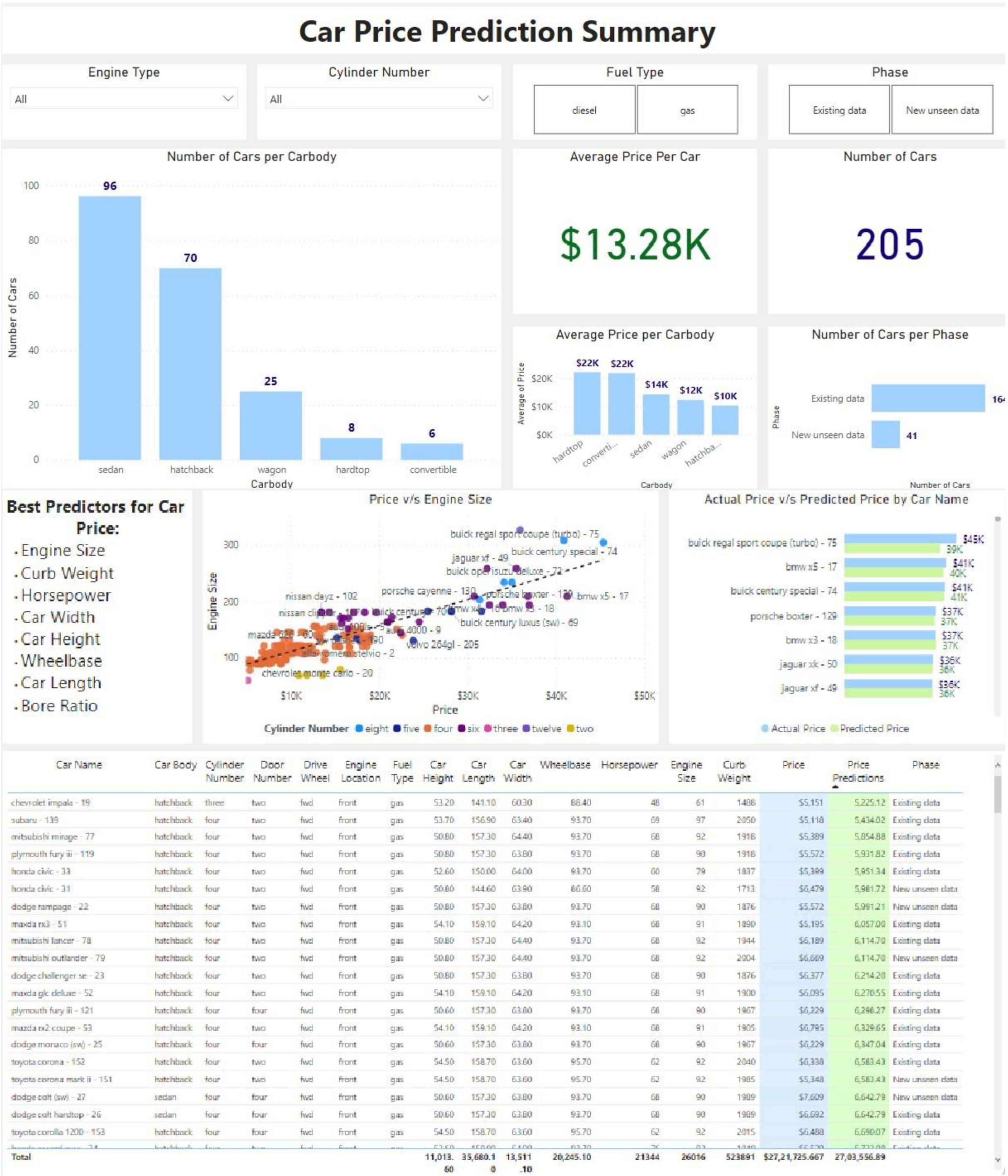
In [100…
```
y2
```

Out[100…
```
0      13495
1      16500
2      16500
3      13950
4      17450
       ...
200    16845
201    19045
202    21485
203    22470
204    22625
Name: price, Length: 205, dtype: int32
```

In [97]:
```
# Split the data into X & y test1 = final_fi['Variable'][0:10].values X2 = new_raw_data[test1].values y2 = new_raw_data['price'].astype(int) # Hold-out validation X_train, X_test, y_train, y_test = train_test_split(X2, y2, train_size=0.80, test_size = 0.2, random_state=15) print(X.shape) print(y.shape) lm2 = LinearRegression(fit_intercept = True) lm2.fit(X_train, y_train) y_pred = lm2.predict(X_train) # Model Accuracy on testing dataset print('The Accuracy on the testing dataset is: ', lm2.score(X_test, y_test) ) print('The RMSE on the testing dataset is: ',sqrt(mean_squared_error(y_test,lm2.predict(X_test)))) print('The MAE on the testing dataset is: ',mean_absolute_error(y_test,lm2.predict(X_test)))
```

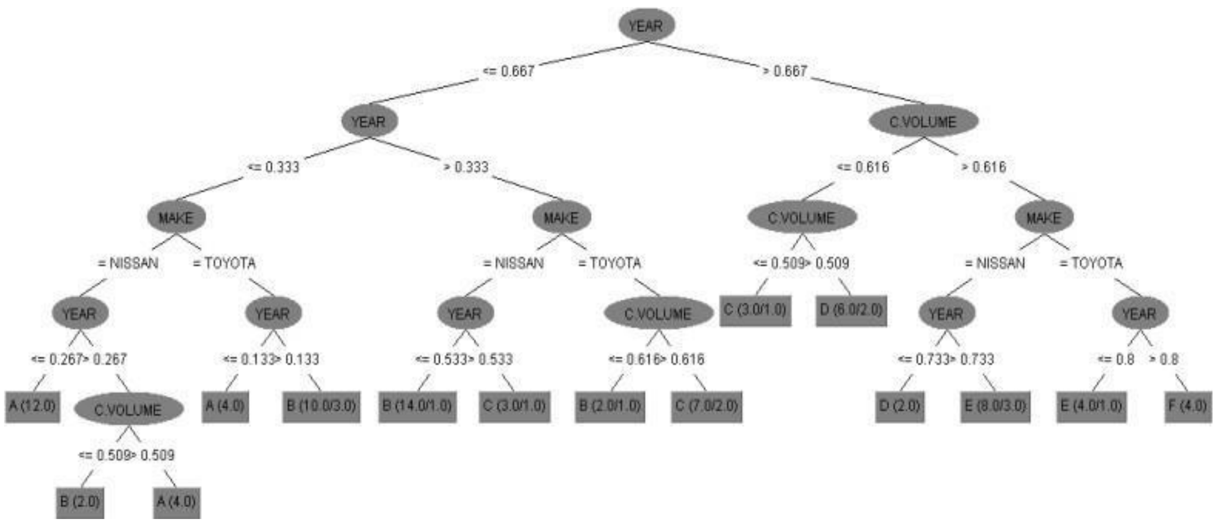
```
(205, 46)
(205,)
The Accuracy  on the testing dataset is:  0.8886586460418306
The RMSE  on the testing dataset is:  2786.112247787891
The MAE  on the testing dataset is:  2045.0313297332943
```

PowerBI Dashboard



ER Diagram

An Entity Relationship (ER) Diagram is a form of flowchart that shows the relationships between "entities" like people, things, or concepts inside a system. ER Diagrams are most frequently used in the disciplines of software engineering, businessinformation systems, education, and research to build or troubleshoot relational databases. They are also known as ERDs or ER Models, and they employ a predetermined collection of symbols to represent the interconnectivity of entities, connections, and their qualities. These symbols include rectangles,diamonds, ovals, andconnecting lines. They have verbs for connections and nouns for entities,mirroring the grammatical framework.



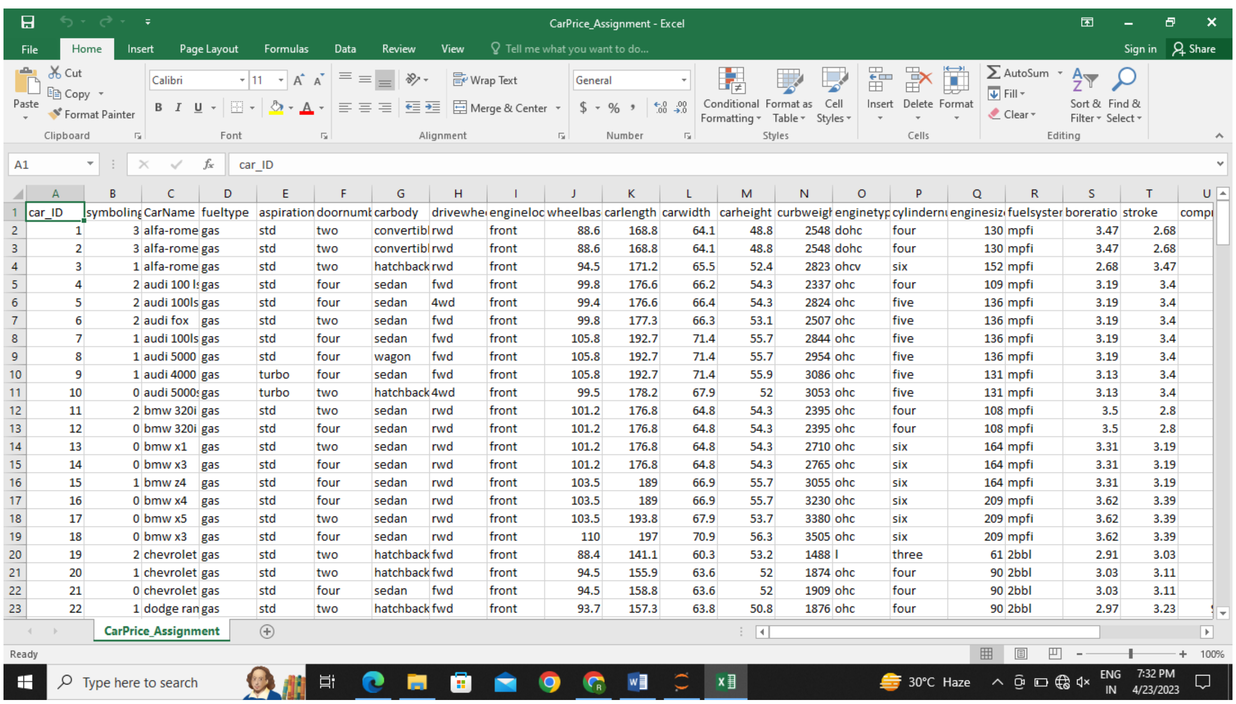
CHAPTER 5

5.1 IMPLEMENTATION AND TESTING

5.1.1 Data Collection

The first step in implementing the car price prediction model is to collect data. A dataset containing relevant car data, including the features mentioned above and corresponding car prices, is necessary fortraining and evaluating the machine learning model. The data can be collected from various sources, such as online car marketplaces, car dealerships, or publicly available datasets. The dataset should be large enough to provide sufficient data points for training and testing the model.

Raw Data



car_ID	symboling	CarName	fueltype	aspiration	doornumb	carbody	drivewhe	engineloc	wheelbas	carlength	carwidth	carheight	curbweigh	enginetyp	cylindern	enginesiz	fuelsyster	boreratio	stroke	compi
1	3	alfa-rome	gas	std	two	convertibl	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	
2	3	alfa-rome	gas	std	two	convertibl	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	
3	1	alfa-rome	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	2823	ohcv	six	152	mpfi	2.68	3.47	
4	2	audi 100 l:	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	2337	ohc	four	109	mpfi	3.19	3.4	
5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	2824	ohc	five	136	mpfi	3.19	3.4	
6	2	audi fox	gas	std	two	sedan	fwd	front	99.8	177.3	66.3	53.1	2507	ohc	five	136	mpfi	3.19	3.4	
7	1	audi 100ls	gas	std	four	sedan	fwd	front	105.8	192.7	71.4	55.7	2844	ohc	five	136	mpfi	3.19	3.4	
8	1	audi 5000	gas	std	four	wagon	fwd	front	105.8	192.7	71.4	55.7	2954	ohc	five	136	mpfi	3.19	3.4	
9	1	audi 4000	gas	turbo	four	sedan	fwd	front	105.8	192.7	71.4	55.9	3086	ohc	five	131	mpfi	3.13	3.4	
10	0	audi 5000s	gas	turbo	two	hatchback	4wd	front	99.5	178.2	67.9	52	3053	ohc	five	131	mpfi	3.13	3.4	
11	2	bmw 320i	gas	std	two	sedan	rwd	front	101.2	176.8	64.8	54.3	2395	ohc	four	108	mpfi	3.5	2.8	
12	0	bmw 320i	gas	std	four	sedan	rwd	front	101.2	176.8	64.8	54.3	2395	ohc	four	108	mpfi	3.5	2.8	
13	0	bmw x1	gas	std	two	sedan	rwd	front	101.2	176.8	64.8	54.3	2710	ohc	six	164	mpfi	3.31	3.19	
14	0	bmw x3	gas	std	four	sedan	rwd	front	101.2	176.8	64.8	54.3	2765	ohc	six	164	mpfi	3.31	3.19	
15	1	bmw z4	gas	std	four	sedan	rwd	front	103.5	189	66.9	55.7	3055	ohc	six	164	mpfi	3.31	3.19	
16	0	bmw x4	gas	std	four	sedan	rwd	front	103.5	189	66.9	55.7	3230	ohc	six	209	mpfi	3.62	3.39	
17	0	bmw x5	gas	std	two	sedan	rwd	front	103.5	193.8	67.9	53.7	3380	ohc	six	209	mpfi	3.62	3.39	
18	0	bmw x3	gas	std	four	sedan	rwd	front	110	197	70.9	56.3	3505	ohc	six	209	mpfi	3.62	3.39	
19	2	chevrolet	gas	std	two	hatchback	fwd	front	88.4	141.1	60.3	53.2	1488	l	three	61	2bbl	2.91	3.03	
20	1	chevrolet	gas	std	two	hatchback	fwd	front	94.5	155.9	63.6	52	1874	ohc	four	90	2bbl	3.03	3.11	
21	0	chevrolet	gas	std	four	sedan	fwd	front	94.5	158.8	63.6	52	1909	ohc	four	90	2bbl	3.03	3.11	
22	1	dodge ran	gas	std	two	hatchback	fwd	front	93.7	157.3	63.8	50.8	1876	ohc	four	90	2bbl	2.97	3.23	

Loading Raw Data

In [2]:
```
# Loading the data
raw_data = pd.read_csv('D:\Final Project\CarPrice_Assignment.csv')

# print the shape
print(raw_data.shape)

#runs the first 5 rows
raw_data.head()
```

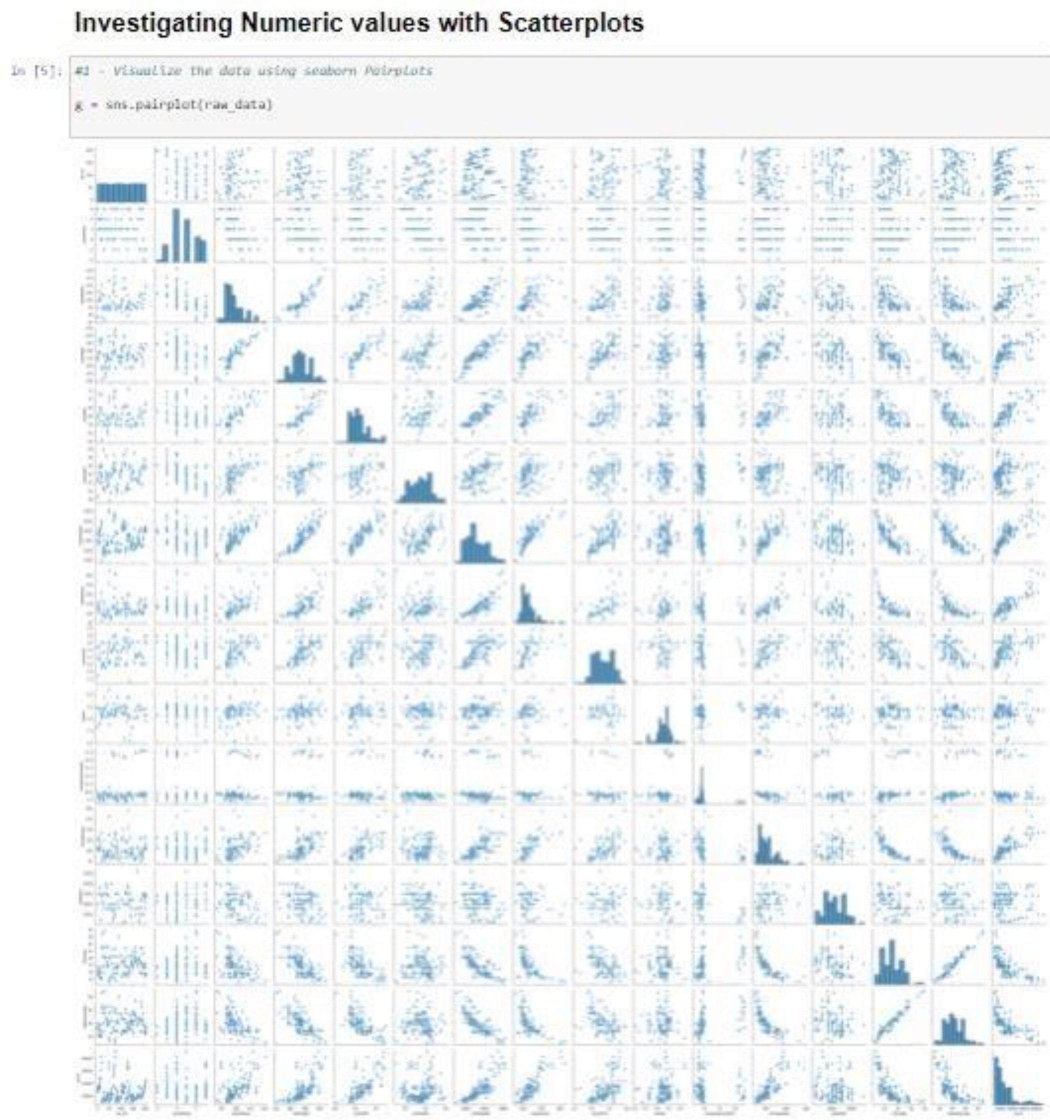
```
(205, 26)
```

Out[2]:

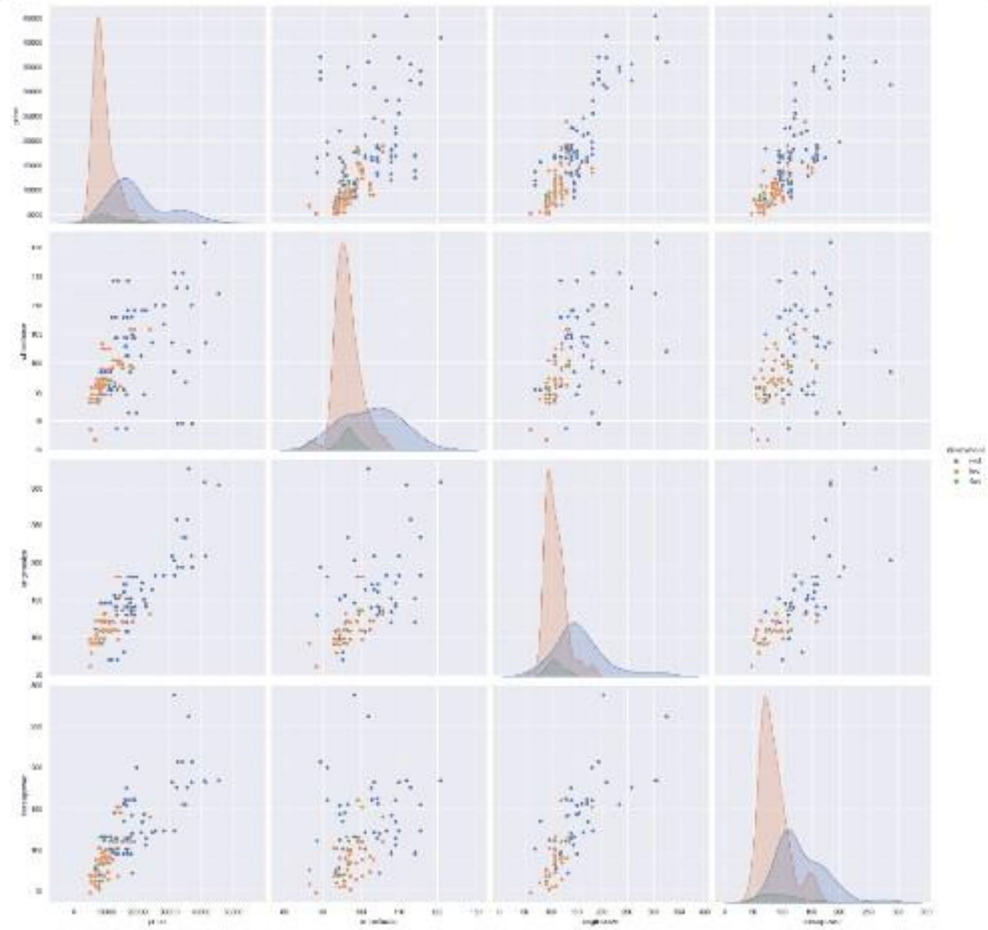
	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight	curbweig
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	25
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	25
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	28
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	23
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	28

5.1.2 Data Preprocessing

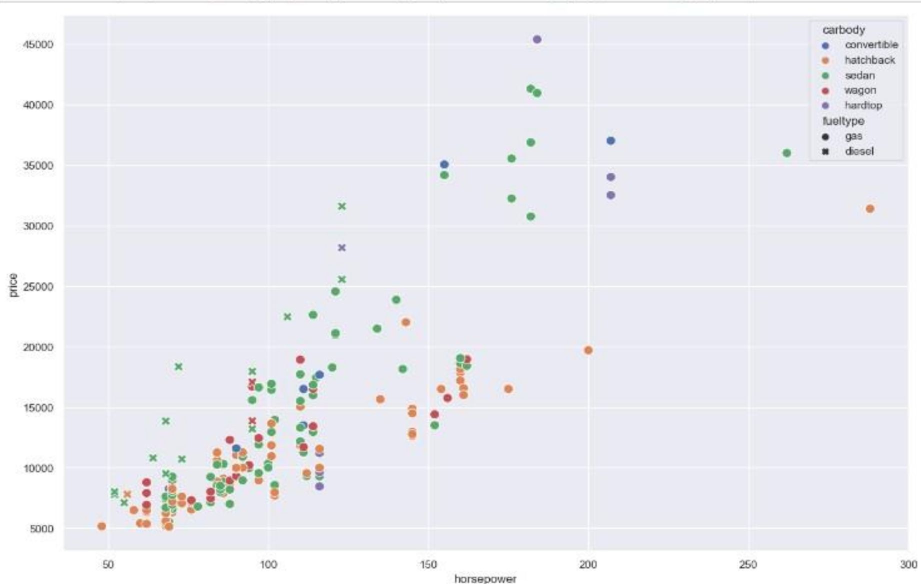
Once the dataset is collected, the next step is to preprocess the data. Data preprocessing involves cleaning, transforming, and encoding the data to make it suitable for training a machine learning model. This may include handling missing values, converting categorical variables into numerical representations, normalizing or scaling numerical features, and removing any irrelevant or redundant data.



In [43]:
```
#2 - Visualising a Subset of our data - Important features

g = sns.pairplot(raw_data[['price', 'wheelbase', 'enginesize','horsepower','drivewheel']], hue = 'drivewheel', height = 5)
```



In [45]:
```
#4 - Relationship between price and horsepower by carbody

ax = sns.scatterplot(x="horsepower", y="price", data=raw_data, hue = 'carbody', style = 'fueltype', s=90)
```



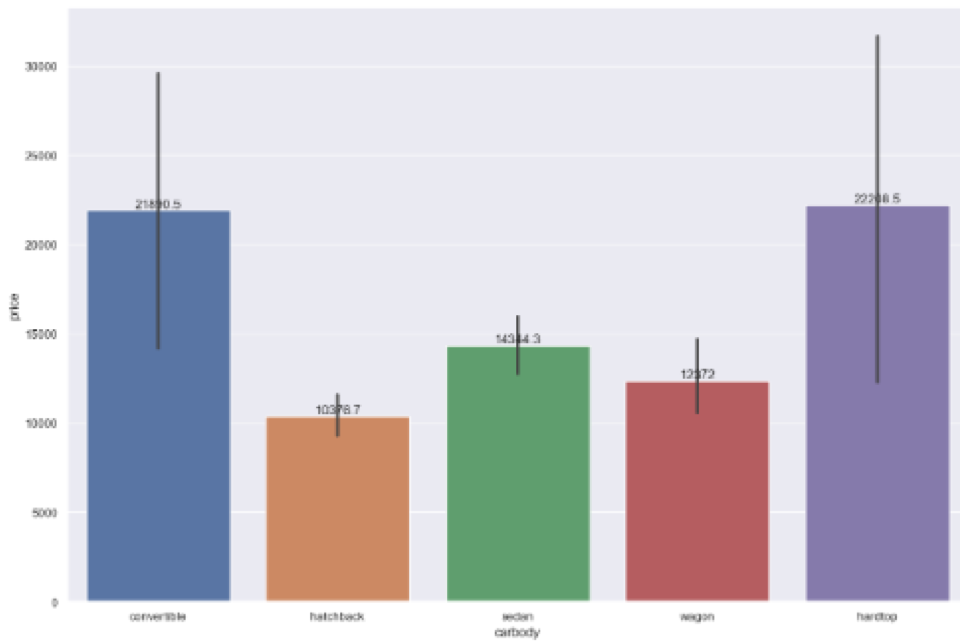
Investigating the Categorical Data

In [46]:

```python
#5 - Average price by carbody

ax = sns.barplot(x="carbody", y="price", data=raw_data)
ax.bar_label(ax.containers[0])

# Notes:
# 1 - the Lines signify the confidence interval
# 2 - Takes mean by default

raw_data[['carbody', 'price']].groupby('carbody', as_index = False).agg({'price':'mean'})
```

Out[46]:

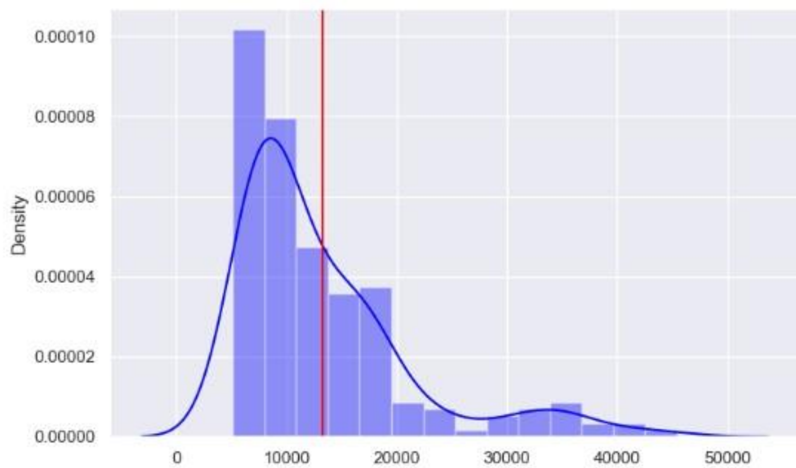
	carbody	price
0	convertible	21890.500000
1	hardtop	22208.500000
2	hatchback	10376.652386
3	sedan	14344.270833
4	wagon	12371.960000



Distribution of Price

Investigating the distribution with disti plots

```
In [48]: #7 - Investigating the distribution of price, adding the mean

         x = raw_data['price'].values

         sns.distplot(x, color = 'blue');

         # Calculating the mean
         mean = raw_data['price'].mean()

         #ploting the mean
         plt.axvline(mean, 0,1, color = 'red')

Out[48]: <matplotlib.lines.Line2D at 0x29d59ed2eb0>
```



Converting Categorical values into Numerical Representation

In [54]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [55]:
```
raw_data.head()
```

Out[55]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight	curbweig
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	25
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	25
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	28
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	23
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	28

In [56]:
```
# Making categorical variables into numeric representation

new_raw_data = pd.get_dummies(raw_data, columns = ['carbody', 'aspiration','doornumber','drivewheel','enginelocation', 'fuelsyst
new_raw_data.head()
```

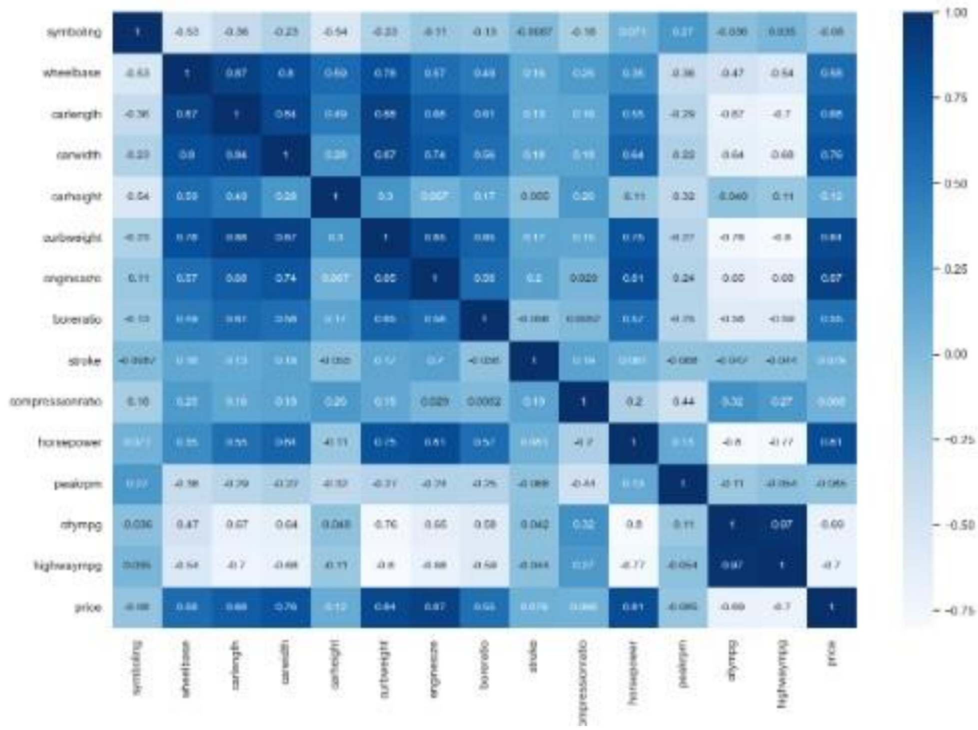
Out[56]:

	car_ID	symboling	CarName	wheelbase	carlength	carwidth	carheight	curbweight	cylindernumber	enginesize	boreratio	stroke	compressionratio	hors
0	1	3	alfa-romero giulia	88.6	168.8	64.1	48.8	2548	four	130	3.47	2.68	9.0	
1	2	3	alfa-romero stelvio	88.6	168.8	64.1	48.8	2548	four	130	3.47	2.68	9.0	
2	3	1	alfa-romero Quadrifoglio	94.5	171.2	65.5	52.4	2823	six	152	2.68	3.47	9.0	
3	4	2	audi 100 ls	99.8	176.6	66.2	54.3	2337	four	109	3.19	3.40	10.0	
4	5	2	audi 100ls	99.4	176.6	66.4	54.3	2824	five	136	3.19	3.40	8.0	

In [57]:
```
# Making categorical variables into numeric representation

new_raw_data = pd.get_dummies(raw_data, columns = ['carbody', 'aspiration','doornumber','drivewheel','enginelocation', 'fuelsyst

new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'two'] = 2
new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'three'] = 3
new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'four'] = 4
new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'five'] = 5
new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'six'] = 6
new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'eight'] = 8
new_raw_data['cylindernumber'].loc[new_raw_data['cylindernumber'] == 'twelve'] = 12
new_raw_data['cylindernumber'] = new_raw_data['cylindernumber'].astype(int)

warnings.simplefilter(action='ignore', category=FutureWarning)

new_raw_data.dtypes
```

Selecting the important features

In [59]:
```python
# Example 12 - Heatmap

# droping columns we don't need
#del new_raw_data['car_ID']
del new_raw_data['CarName']

# Increases the size of sns plots
sns.set(rc={'figure.figsize':(15,10)})

n_variables = ['symboling', 'wheelbase', 'carlength', 'carwidth', 'carheight',
       'curbweight', 'enginesize', 'boreratio', 'stroke', 'compressionratio',
       'horsepower', 'peakrpm', 'citympg', 'highwaympg', 'price']

pc = new_raw_data[n_variables].corr(method ='pearson')

cols = n_variables

ax = sns.heatmap(pc, annot=True,
                 yticklabels=cols,
                 xticklabels=cols,
                 annot_kws={'size':10},
                 cmap="Blues")
```



Feature Importance

Feature Importance

Steps to run Feature Importance

1. Split the data into x & y
2. Run a Tree-Based estimators (i.e. Decision Tree & Random Forest)
3. Run Feature Importance

```
In [61]: # Split the data into X & y

         X = new_raw_data.drop(['price'], axis = 1).values
         X_columns = new_raw_data.drop(['price'], axis = 1)
         y = new_raw_data['price'].astype(int)

         print(X.shape)
         print(y.shape)

         (205, 47)
         (205,)
```

```
In [62]: # Run a Tree-based estimators (i.e. decision trees & random forests)

         dt = DecisionTreeClassifier(random_state=15, criterion  = 'entropy', max_depth = 10)
         dt.fit(X,y)

Out[62]: DecisionTreeClassifier(criterion='entropy', max_depth=10, random_state=15)
```

```
In [63]: dt.feature_importances_

Out[63]: array([0.067097, 0.008362, 0.106282, 0.057973, 0.039068, 0.110696,
                0.241150, 0.001786, 0.053168, 0.004961, 0.016314, 0.008851,
                0.094566, 0.014268, 0.014436, 0.048669, 0.000000, 0.005677,
                0.011418, 0.001297, 0.009659, 0.001297, 0.012283, 0.017902,
                0.006655, 0.000000, 0.009690, 0.003890, 0.003890, 0.000000,
                0.000000, 0.004934, 0.000000, 0.008393, 0.000000, 0.003083,
                0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.009690,
                0.000000, 0.000000, 0.000000, 0.000000, 0.002594])
```

```
In [64]: #del final_fi

         # Calculating FI
         for i, column in enumerate(new_raw_data.drop('price', axis=1)):
             print('Importance of feature {}:, {:.3f}'.format(column, dt.feature_importances_[i]))

             fi = pd.DataFrame({'Variable': [column], 'Feature Importance Score': [dt.feature_importances_[i]]})

             try:
                 final_fi = pd.concat([final_fi,fi], ignore_index = True)
             except:
                 final_fi = fi


         # Ordering the data
         final_fi = final_fi.sort_values('Feature Importance Score', ascending = False).reset_index()
         final_fi
```

```
Importance of feature car_ID:, 0.067
Importance of feature symboling:, 0.008
Importance of feature wheelbase:, 0.106
Importance of feature carlength:, 0.058
Importance of feature carwidth:, 0.039
Importance of feature carheight:, 0.111
Importance of feature curbweight:, 0.241
Importance of feature cylindernumber:, 0.002
Importance of feature enginesize:, 0.053
Importance of feature boreratio:, 0.005
Importance of feature stroke:, 0.016
Importance of feature compressionratio:, 0.009
Importance of feature horsepower:, 0.095
Importance of feature peakrpm:, 0.014
Importance of feature citympg:, 0.014
Importance of feature highwaympg:, 0.049
Importance of feature carbody_convertible:, 0.000
Importance of feature carbody_hardtop:, 0.006
Importance of feature carbody_hatchback:, 0.011
Importance of feature carbody_sedan:, 0.001
Importance of feature carbody_wagon:, 0.010
Importance of feature aspiration_std:, 0.001
Importance of feature aspiration_turbo:, 0.012
Importance of feature doornumber_four:, 0.018
Importance of feature doornumber_two:, 0.007
Importance of feature drivewheel_4wd:, 0.000
Importance of feature drivewheel_fwd:, 0.010
Importance of feature drivewheel_rwd:, 0.004
Importance of feature enginelocation_front:, 0.004
Importance of feature enginelocation_rear:, 0.000
Importance of feature fuelsystem_1bbl:, 0.000
Importance of feature fuelsystem_2bbl:, 0.005
Importance of feature fuelsystem_4bbl:, 0.000
Importance of feature fuelsystem_idi:, 0.008
Importance of feature fuelsystem_mfi:, 0.000
Importance of feature fuelsystem_mpfi:, 0.003
Importance of feature fuelsystem_spdi:, 0.000
Importance of feature fuelsystem_spfi:, 0.000
Importance of feature enginetype_dohc:, 0.000
Importance of feature enginetype_dohcv:, 0.000
Importance of feature enginetype_l:, 0.000
Importance of feature enginetype_ohc:, 0.010
Importance of feature enginetype_ohcf:, 0.000
Importance of feature enginetype_ohcv:, 0.000
Importance of feature enginetype_rotor:, 0.000
Importance of feature fueltype_diesel:, 0.000
Importance of feature fueltype_gas:, 0.003
```

Out[64]:

	level_0	index	Variable	Feature Importance Score
0	0	6.0	curbweight	0.241150
1	53	NaN	curbweight	0.241150
2	1	5.0	carheight	0.110898
3	52	NaN	carheight	0.110898
4	2	2.0	wheelbase	0.106282
...	...	...	...	...
89	43	32.0	fuelsystem_4bbl	0.000000
90	34	44.0	enginetype_rotor	0.000000
91	33	45.0	fueltype_diesel	0.000000
92	32	39.0	enginetype_dohcv	0.000000
93	42	16.0	carbody_convertible	0.000000

94 rows × 4 columns

5.1.3 Splitting the Data

After preprocessing the data, it is important to split the dataset into training and testing sets. The training set is used for training the machine learning model, while the testing set is used for evaluating the model's performance. Typically, the dataset is split into a certain percentage, such as 80% for training and 20% for testing, using techniques like random sampling or stratified sampling to ensure representative distribution of data across both sets.

Splitting the Raw Data - Hold-Out Validation

In [65]:
```
# Hold-out validation

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.80, test_size = 0.2, random_state=15)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(164, 47)
(41, 47)
(164,)
(41,)
```

In [66]:
```
# Training the Regression

lm = LinearRegression(fit_intercept = True)
lm.fit(X_train, y_train)

y_pred = lm.predict(X_train)
```

In [67]:
```
# Model Accuracy on training dataset

print('The Accuracy  on the training dataset is: ', lm.score(X_train, y_train) )
print('The Accuracy r2  on the training dataset prediction is: ',r2_score(y_train,y_pred) )

print("")
# Model Accuracy on testing dataset
print('The Accuracy  on the testing dataset is: ', lm.score(X_test, y_test) )

print("")
# The Root Mean Squared Error (RMSE)
print('The RMSE  on the training dataset is: ',sqrt(mean_squared_error(y_train,y_pred)))
print('The RMSE  on the testing dataset is: ',sqrt(mean_squared_error(y_test,lm.predict(X_test))))

print("")
# The Mean Absolute Error (MAE)
print('The MAE  on the training dataset is: ',mean_absolute_error(y_train,y_pred))
print('The MAE  on the testing dataset is: ',mean_absolute_error(y_test,lm.predict(X_test)))


print("")
# Coefficients
print('Coefficients: ', lm.coef_ )

print("")
# The Intercept
print('Intercept: ', lm.intercept_)
```

```
The Accuracy  on the training dataset is:  0.939812854622914
The Accuracy r2  on the training dataset prediction is:  0.939812854622914

The Accuracy  on the testing dataset is:  0.8508618050986757

The RMSE  on the training dataset is:  1943.8483958356274
The RMSE  on the testing dataset is:  3224.51796938918

The MAE  on the training dataset is:  1422.0261986002433
The MAE  on the testing dataset is:  2273.4706752812954

Coefficients:  [-15.338950 310.151322 59.886408 -29.377917 546.770787 323.313594 5.310580
 -1430.395287 176.711113 -8126.107835 -5483.145262 -786.406633 13.581598
 2.554469 -10.337758 212.293090 3085.668953 -1977.929992 157.227140
 136.622407 -1401.588508 -633.439236 633.439236 170.495715 -170.495715
 -721.843059 -891.009633 1612.852691 -4227.250691 4227.250691 -1039.568450
 752.677402 650.137888 3709.099959 -2488.091529 1313.157971 -1376.357399
 -1521.055842 -2566.560595 3601.273250 -3836.289677 972.667940 2115.285993
 -5921.093144 5634.796234 3709.099959 -3709.099959]

Intercept:  -31190.58931211301
```

5.1.4 Model Training

With the training and testing sets prepared, the next step is to train the car price prediction model using machine learning algorithms. Various algorithms can be used, such as linear regression, decision trees, or support vector machines, depending on the characteristics of the data and the project requirements. The training set is used to fit the model to the data, and the model learns the underlying patterns and relationships between the features and car prices.

RANDOM FOREST MODEL

Random Forest Model

In [77]:
```
from sklearn.ensemble import RandomForestRegressor

# Hold-out validation
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.80, test_size = 0.2, random_state=15)

regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(X_train, y_train)

y_pred = regr.predict(X_train)


# Model Accuracy on testing dataset
print('The Accuracy  on the testing dataset is: ', regr.score(X_test, y_test) )
print('The RMSE  on the testing dataset is: ',sqrt(mean_squared_error(y_test,regr.predict(X_test))))
print('The MAE  on the testing dataset is: ',mean_absolute_error(y_test,regr.predict(X_test)))
```

```
The Accuracy  on the testing dataset is:  0.87857909726018
The RMSE  on the testing dataset is:  2909.4914752165437
The MAE  on the testing dataset is:  2121.1311847103157
```

In [78]:
```
# Optimizing Random Forest R

from sklearn.model_selection import GridSearchCV

param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}

grid_search = GridSearchCV(estimator = regr, param_grid = param_grid,
                          cv = 3, n_jobs = -1, verbose = 2)

grid_search.fit(X_train, y_train)

grid_search.best_params_
```

```
Fitting 3 folds for each of 288 candidates, totalling 864 fits
```

Out[78]:
```
{'bootstrap': True,
 'max_depth': 80,
 'max_features': 3,
 'min_samples_leaf': 3,
 'min_samples_split': 8,
 'n_estimators': 1000}
```

In [79]:
```
best_grid = grid_search.best_estimator_
best_grid
```

Out[79]:
```
RandomForestRegressor(max_depth=80, max_features=3, min_samples_leaf=3,
                      min_samples_split=8, n_estimators=1000, random_state=0)
```

XG BOOST REGRESSOR

XG Boost Regressor

```
In [81]: from sklearn.ensemble import GradientBoostingRegressor

         reg = GradientBoostingRegressor(random_state=0)
         reg.fit(X_train, y_train)

         y_pred = reg.predict(X_train)

         # Model Accuracy on testing dataset
         print('The Accuracy  on the testing dataset is: ', reg.score(X_test, y_test) )
         print('The RMSE  on the testing dataset is: ',sqrt(mean_squared_error(y_test,reg.predict(X_test))))
         print('The MAE  on the testing dataset is: ',mean_absolute_error(y_test,reg.predict(X_test)))

         The Accuracy  on the testing dataset is:  0.944538667933968
         The RMSE  on the testing dataset is:  1966.3712165085378
         The MAE  on the testing dataset is:  1374.4676099853177

In [84]: pip install xgboost

         Collecting xgboost
           Downloading xgboost-1.7.5-py3-none-win_amd64.whl (70.9 MB)
              ---------------------------------------- 70.9/70.9 MB 6.1 MB/s eta 0:00:00
         Requirement already satisfied: scipy in c:\users\latitude\anaconda3\lib\site-packages (from xgboost) (1.9.1)
         Requirement already satisfied: numpy in c:\users\latitude\anaconda3\lib\site-packages (from xgboost) (1.21.5)
         Installing collected packages: xgboost
         Successfully installed xgboost-1.7.5
         Note: you may need to restart the kernel to use updated packages.

In [85]: # optimizing XGBoost Regressor

         from sklearn.model_selection import GridSearchCV
         import xgboost as xgb

         params = { 'max_depth': [3,6,9,12],
                    'learning_rate': [0.01, 0.05, 0.1],
                    'n_estimators': [100, 500, 1000],
                    'colsample_bytree': [0.3, 0.7]}

         xgbr = xgb.XGBRegressor(seed = 20)

         clf = GridSearchCV(estimator=xgbr,
                            param_grid=params,
                            scoring='neg_mean_squared_error',
                            verbose=1)

         clf.fit(X_train, y_train)

         print("Best parameters:", clf.best_params_)

         Fitting 5 folds for each of 72 candidates, totalling 360 fits
         Best parameters: {'colsample_bytree': 0.3, 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 500}
```

5.1.5 Model Evaluation

After training the model, it is essential to evaluate its performance using the testing set. The testing set is used to assess how well the model generalizes to new, unseen data. Performance metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or R-squared(R2) can be used to evaluate the model's accuracy, precision, and generalization ability. The model may need to be fine-tuned or retrained based on the evaluation results to improve its performance.

RANDOM FOREST MODEL

```
In [80]: best_grid = grid_search.best_estimator_

         regr = best_grid
         regr.fit(X_train, y_train)

         y_pred = regr.predict(X_train)


         # Model Accuracy on testing dataset
         print('The Accuracy  on the testing dataset is: ', regr.score(X_test, y_test) )
         print('The RMSE  on the testing dataset is: ',sqrt(mean_squared_error(y_test,regr.predict(X_test))))
         print('The MAE  on the testing dataset is: ',mean_absolute_error(y_test,regr.predict(X_test)))

         The Accuracy  on the testing dataset is:  0.8791273338526945
         The RMSE  on the testing dataset is:  2902.9156128245904
         The MAE  on the testing dataset is:  1662.0524670474829
```

The Random Forest Model gave us an accuracy of about **87%**

XG BOOST REGRESSOR

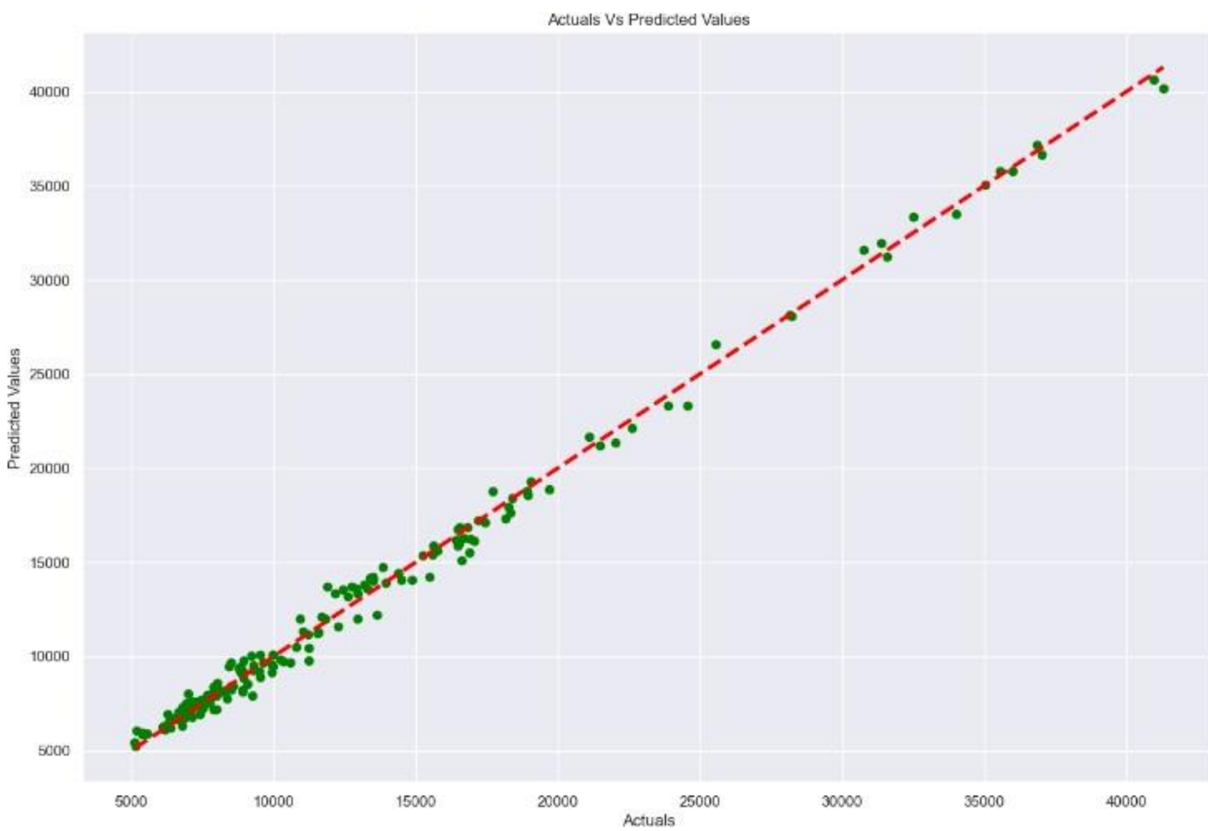
```
In [86]: # Training the model on best parameters

         xgbr = xgb.XGBRegressor(seed = 20, colsample_bytree = 0.7, learning_rate= 0.1, max_depth=12, n_estimators=500)
         xgbr.fit(X_train, y_train)
         xgbr

         y_pred = xgbr.predict(X_train)
         # Model Accuracy on testing dataset
         print('The Accuracy  on the testing dataset is: ', xgbr.score(X_test, y_test) )
         print('The RMSE  on the testing dataset is: ',sqrt(mean_squared_error(y_test,xgbr.predict(X_test))))
         print('The MAE  on the testing dataset is: ',mean_absolute_error(y_test,xgbr.predict(X_test)))

         The Accuracy  on the testing dataset is:  0.9458283000245309
         The RMSE  on the testing dataset is:  1943.3749150373783
         The MAE  on the testing dataset is:  1242.6066239519816
```

The XG Boost Regressor Model gave us an accuracy of about **94%**

5.1.6 Testing

After deployment, it is crucial to thoroughly test the model to ensure its reliability and accuracy. Testinginvolves inputting various test cases, including edge cases and corner cases, to validate the model's predictions. The testing process helps identify any potential issues or errors in the model's predictions and allows for necessary adjustments or refinements to be made.

Plot the graph of the actual price and Predicted price

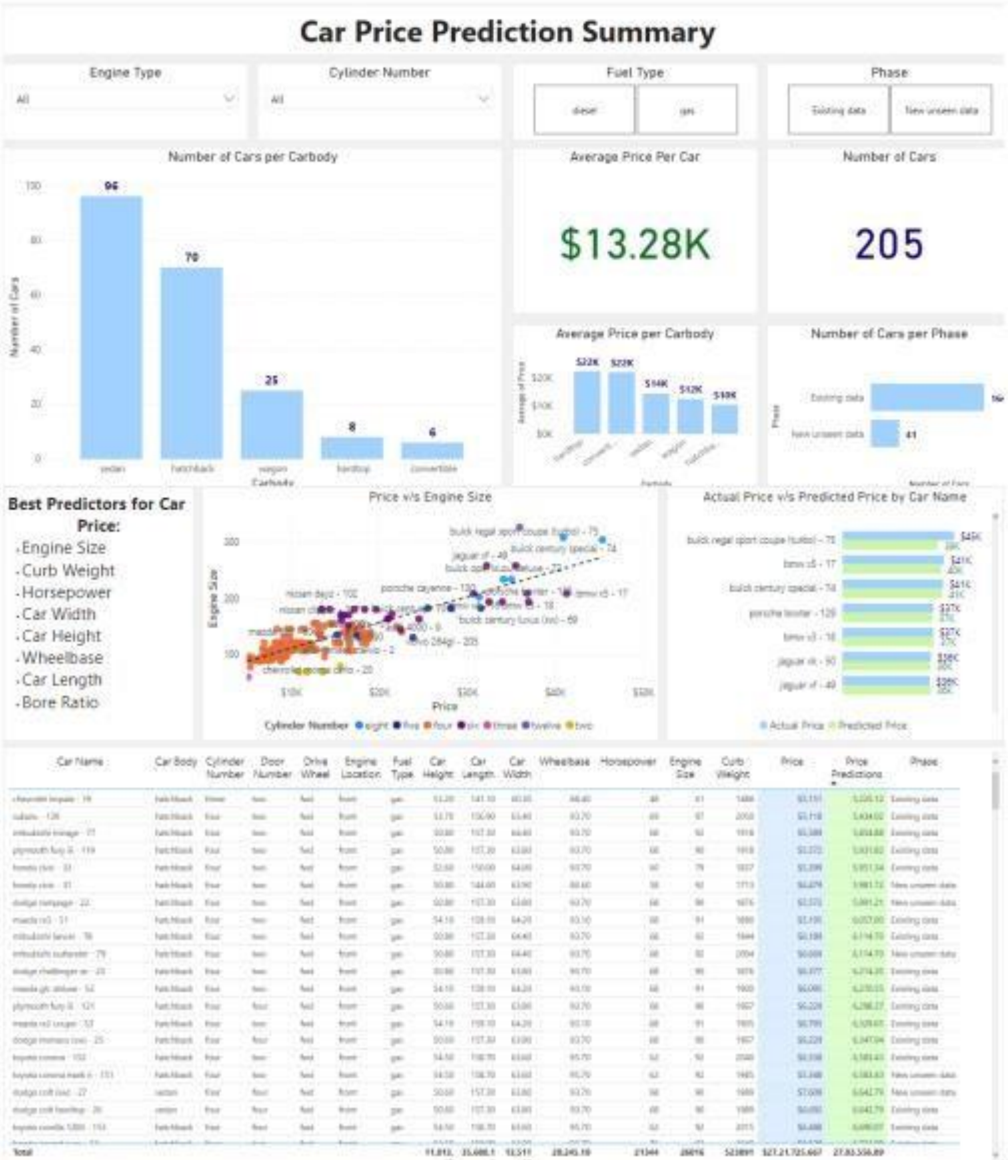


CHAPTER 6

6.1 Results and Discussion

XG BOOST REGRESSOR without optimization gave the best results

```
In [88]: from sklearn.ensemble import GradientBoostingRegressor

         # Split the data into X & y

         #del new_raw_data['Price Predictions']

         X = new_raw_data.drop(['price'], axis = 1).values
         X_columns = new_raw_data.drop(['price'], axis = 1)
         y = new_raw_data['price'].astype(int)

         print(X.shape)
         print(y.shape)

         # Hold-out validation

         X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.80, test_size = 0.2, random_state=15)

         print(X_train.shape)
         print(X_test.shape)
         print(y_train.shape)
         print(y_test.shape)

         reg = GradientBoostingRegressor(random_state=0)
         reg.fit(X_train, y_train)

         y_pred = reg.predict(X_train)

         # Model Accuracy on testing dataset
         print('The Accuracy  on the testing dataset is: ', reg.score(X_test, y_test) )
         print('The RMSE  on the testing dataset is: ',sqrt(mean_squared_error(y_test,reg.predict(X_test))))
         print('The MAE  on the testing dataset is: ',mean_absolute_error(y_test,reg.predict(X_test)))

         (205, 47)
         (205,)
         (164, 47)
         (41, 47)
         (164,)
         (41,)
         The Accuracy  on the testing dataset is:  0.944538667933968
         The RMSE  on the testing dataset is:  1966.3712165085378
         The MAE  on the testing dataset is:  1374.4676099853177
```


Car Price Prediction Summary
Engine Type
Cylinder Number
Fuel Type
Phase
Number of Cars per Carbody
96
70
25
8
6
Average Price Per Car
$13.28K
Number of Cars
205
Average Price per Carbody
Number of Cars per Phase
Best Predictors for Car Price:
-Engine Size
-Curb Weight
-Horsepower
-Car Width
-Car Height
-Wheelbase
-Car Length
-Bore Ratio
Price v/s Engine Size
Actual Price v/s Predicted Price by Car Name

CHAPTER 7

7.1 CONCLUSION

In conclusion, the development and implementation of a car price prediction model using machine learning algorithms have been successful. The project aimed to predict car prices based on various features such as engine size, curb weight, horsepower, car height, car width, wheelbase, car length, and bore ratio. The implementation involved several steps, including data collection, data preprocessing, data splitting, model training, model evaluation, deployment, and testing.

Significance:

The car price prediction model has significant practical implications in the automotive industry. The model can help car dealerships, buyers, and sellers make informed decisions about car prices. It can also assist in the valuation of used cars, aiding buyers and sellers in negotiations. Moreover, the model's accuracy and generalization ability can help reduce the time and cost associated with manual car appraisal processes.

Future Scope:

The car price prediction model has enormous potential for future development and research. The model can be extended to include more features, such as car age, car brand, location, and fuel efficiency, to improve the accuracy and reliability of car price predictions. Additionally, the model can be integrated into various applications, such as car marketplaces, insurance companies, or car rental services, to provide real-time car price predictions and valuation.

Future Work:

There are several modifications and changes that can be made to the car price prediction model to improve its performance and functionality. For instance, the model's architecture and hyperparameters can be fine-tuned to achieve better accuracy and reduce overfitting. Moreover, the model's training data can be augmented, and new data sources can be added to improve the model's generalization ability. Additionally, the model's deployment and testing processes can be optimized to ensure efficient and secure usage.

Limitations:

Despite the car price prediction model's potential, there are several limitations and challenges that need to be addressed. The model's accuracy and reliability are highly dependent on the quality and representativeness of the training data. The model may also face challenges in predicting car prices for rare or unique cars, as there may not be sufficient data points for training. Moreover, the model's performance may be affected by external factors, such as economic conditions, market trends, or regulatory changes.

Conclusion:

In conclusion, the car price prediction model using machine learning algorithms is a promising solution for predicting car prices based on various features. The implementation of the model involves several key steps, including data collection, data preprocessing, data splitting, model training, model evaluation, deployment, and testing. The model has significant practical implications for the automotive industry, aiding in car valuation, pricing, and negotiations. Additionally, the model has enormous potential for future development, research, and integration into various applications. Despite some limitations and challenges, the car price prediction model is a valuable tool for making informed decisions about car prices, benefiting car dealerships, buyers, and sellers.

REFERENCES

Dataset

https://www.kaggle.com/datasets/shaistashaikh/carprice-assignment

YouTube
https://www.youtube.com/watch?v=Q0Q4x58h_BA&t=
2631s