A PROJECT REPORT ON

CAR PRICE PREDICTION

A project report submitted in fulfillment for the Diploma in AI & ML

Under

Applied Roots with University Of Hyderabad

Project Submitted By

Rohit Kumar Pali

Under The Guidance Of

Mentors : Harsh Sharma

Approved By : Mentor : Harsh Sharma

University Of Hyderabad

Declaration of Authorship

We hereby declare that this thesis titled" Car Price Prediction" and the work presented by the undersigned candidate, as part of Diploma Degree in AI & ML. All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: Rohit Kumar Pali
Thesis Title: Car Price Prediction

CERTIFICATE OF RECOMMENDATION

We hereby recommend that the thesis entitled “Car Price Prediction” prepared under my supervision and guidance by Rohit Kumar Pali be accepted in fulfilment of the requirement for awarding the degree of Diploma in AI & ML Under applied roots with University of Hyderabad. The project, in our opinion, is worthy for its acceptance.

Mentor : Harsh Sharma

Under Applied roots with

University of Hyderabad

ACKNOWLEDGEMENT

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

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

Name: Rohit Kumar Pali

Contents

ABSTRACT :

The newer concept of predicting the prices of certain items. A car price prediction has been a high-interest research area, as it requires noticeable effort and knowledge of the field expert. To build a model for predicting the price of used cars in, we applied one of the machine learning techniques i.e., Linear Regression. Our main proposes a system where the price is a dependent variable that is predicted, and this price is derived from factors like distance is driven by the car, car purchase year, etc.

1. Introduction

Car price prediction is somehow interesting and popular problem. Accurate car price prediction involves expert knowledge, because price usually depends on many distinctive features and factors. Typically, most significant ones are brand and model, age, horsepower and mileage. The fuel type used in the car as well as fuel consumption per mile highly affect price of a car due to a frequent changes in the price of a fuel. Different features like exterior color, door number, type of transmission, dimensions, safety, air condition, interior, whether it has navigation or not will also influence the car price. In this we applied different methods and techniques in order to achieve higher precision of the used car price prediction.

Problem Statement

1. What is the problem all about?

In our day to day lives everyone buys and sells a car every day. Now there are limited facilities and applications to get an appropriate price for one's car. Now we need an application to get an estimate value of the car. We use a linear regression algorithm for this purpose. This object comprises of Car as an object and the price for every car entered in the dataset. The dataset we work on in this venture is imaginary i.e., not real. Linear Regression allows us to compare our entered data to the existing data in the dataset and gives us an estimate value. Car works as a main object. The attributes such as Company Name, Kilometres Driven, Fuel Type and Year of Purchase etc

2. Why is this an important problem to solve?

It is important to solve because with the help of this function we can simply put the data and know the price of the car that we want to purchase. It will help the user to get their desired car.

3. Business/Real-world impact of solving this problem

It will be used by the management to understand how exactly the prices vary with the independent variables. They can accordingly manipulate the design of the cars, the business strategy etc. to meet certain price levels.

2. Data Description:

1. *Source of the dataset*

   So, the source of the dataset is quickr.com

2. *Explanation of each feature and datapoint available*

   In the given dataset it has cars-name, cars-model, year of selling, price of the car,
   Kms is driven by the car & fuel type of the car.

3. *Data size and any challenges you foresee in processing it? Tools that you will use to process this data.*

   Yes there are some errors in the data in some of the columns like
   Years, Price, Kms Driven & Fuel type.

   The solution is divided into the following sections :

   - *Data understanding and exploration*
   - *Data cleaning*
   - *Data Preparation*

Tools that are used

1. *Pandas*
2. *Sci-kit learn*
3. *Numpy*
4. *Flask*

5. *Render template*
6. *Redirect*
7. *Pickle*

Key metrics (KPI) to optimize

a. *Business Metric definition.*

Data science techniques for the price of cars with the available independent variables. That should help the management to understand how exactly the prices vary with the independent variables. They can accordingly manipulate the design of the cars, the business strategy etc.

b. *Why is this metric used?*

This metric represents the part of the variance of the dependent variable explained by the independent variables of the model. It measures the strength of the relationship between your model and the dependent variable.

c. *Pros and cons of the metric used.*

Pros :

-Linear regression performs exceptionally well for linearly separable data
-Easier to implement, interpret and efficient to train

Cons:

- Linear regression is quite sensitive to outliers
d. Where does this metric fail? Where should it not be used?

Linear regression fails at finding relationships that are non-linear in nature. So if a variable increase at the rate of the log of another variable, linear regression will not describe the relationship well.

Real-world challenges and constraints

a. What real-world constraints do you have while solving this problem?

-We assume linear relationship between target & independent variable

-Feature are not always independent they are corelated

b. What are the requirements that your solution must meet?

Linear relationship: There exists a linear relationship between the independent variable, x, and the dependent variable, y.

*. **List/Cite all references you go through**.*

https://www.kaggle.com/code/goyalshalini93/car-price-prediction-linear-regression-rfe/notebook

https://towardsdatascience.com/predicting-car-price-using-machine-learning-8d2df3898f16

https://www.irjet.net/archives/V8/i4/IRJET-V8I4278.pdf

https://www.researchgate.net/publication/360005567_Car_Price_Prediction_using_Linear_Regression_Technique_of_Machine_Learning

3. EDA and Feature Extraction

EDA :

Exploratory Data Analysis (EDA) is an approach to analyze the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
import io
import pandas as pd

data = files.upload()

<IPython.core.display.HTML object>

Saving quikr_car.csv to quikr_car.csv


import pandas as pd
car=pd.read_csv('quikr_car.csv')
car.head()


 name   company  year         Price  \
0    Hyundai Santro Xing XO eRLX Euro III   Hyundai  2007        80,000
1                 Mahindra Jeep CL550 MDI  Mahindra  2006       4,25,000
2              Maruti Suzuki Alto 800 Vxi    Maruti  2018  Ask For Price
3  Hyundai Grand i10 Magna 1.2 Kappa VTVT   Hyundai  2014       3,25,000
4         Ford EcoSport Titanium 1.5L TDCi      Ford  2014       5,75,000


 kms_driven fuel_type
0  45,000 kms    Petrol
```

```
1      40 kms    Diesel
2  22,000 kms    Petrol
3  28,000 kms    Petrol
4  36,000 kms    Diesel
```

```
car.shape
```

```
(892, 6)
```

```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 892 entries, 0 to 891
Data columns (total 6 columns):

 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   name        892 non-null    object
 1   company     892 non-null    object
 2   year        892 non-null    object
 3   Price       892 non-null    object
 4   kms_driven  840 non-null    object
 5   fuel_type   837 non-null    object

dtypes: object(6)
memory usage: 41.9+ KB
```

```
car['year'].unique()
```

```
array(['2007', '2006', '2018', '2014', '2015', '2012', '2013', '2016',
       '2010', '2017', '2008', '2011', '2019', '2009', '2005', '2000',
       '...', '150k', 'TOUR', '2003', 'r 15', '2004', 'Zest', '/-Rs',
       'sale', '1995', 'ara)', '2002', 'SELL', '2001', 'tion', 'odel',
       '2 bs', 'arry', 'Eon', 'o...', 'ture', 'emi', 'car', 'able', 'no.',
       'd...', 'SALE', 'digo', 'sell', 'd Ex', 'n...', 'e...', 'D...',
       ', Ac', 'go .', 'k...', 'o c4', 'zire', 'cent', 'Sumo', 'cab',
       't xe', 'EV2', 'r...', 'zest'], dtype=object)
```

In the given data we find some false data present not only in year coloumn but in other coloumn also so first we have to clean the false data and make a new data.

So the issue is found in this data are :

Now we have to start cleaning our data but before that we have to make a backup of the data

```
backup=car.copy()
```

```
car=car[car['year'].str.isnumeric()]
```

```
car['year']=car['year'].astype(int)
```

```
car=car[car['Price']!="Ask For Price"]
```

```
car['Price']=car['Price'].str.replace(',','').astype(int)
```

```
car['kms_driven']=car['kms_driven'].str.split(' ').str.get(0).str.replace(',','')
```

```
car=car[car['kms_driven'].str.isnumeric()]
```

```
car=car[~car['fuel_type'].isna()]
```

```
car['name']=car['name'].str.split(' ').str.slice(0,3).str.join(' ')
```

```
car=car.reset_index(drop=True)
```

```
car.describe()
```

```
              year         Price
count   816.000000  8.160000e+02
mean   2012.444853  4.117176e+05
std       4.002992  4.751844e+05
min    1995.000000  3.000000e+04
25%    2010.000000  1.750000e+05
50%    2013.000000  2.999990e+05
```

75% 2015.000000 4.912500e+05
max 2019.000000 8.500003e+06

car=car[car['Price']<6e6]

car

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	80000	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	425000	40	Diesel
2	Hyundai Grand i10	Hyundai	2014	325000	28000	Petrol
3	Ford EcoSport Titanium	Ford	2014	575000	36000	Diesel
4	Ford Figo	Ford	2012	175000	41000	Diesel
..	...	...	...	...	...	...
811	Maruti Suzuki Ritz	Maruti	2011	270000	50000	Petrol
812	Tata Indica V2	Tata	2009	110000	30000	Diesel
813	Toyota Corolla Altis	Toyota	2009	300000	132000	Petrol
814	Tata Zest XM	Tata	2018	260000	27000	Diesel
815	Mahindra Quanto C8	Mahindra	2013	390000	40000	Diesel

[815 rows x 6 columns]

Now i have to store the clean data

car.to_csv('Cleaned Car.csv')

Extract the features and lables.So in the data price is not a features of the car so we have to the extract the price from the given data

Now In X we have car features & In Y we have car price

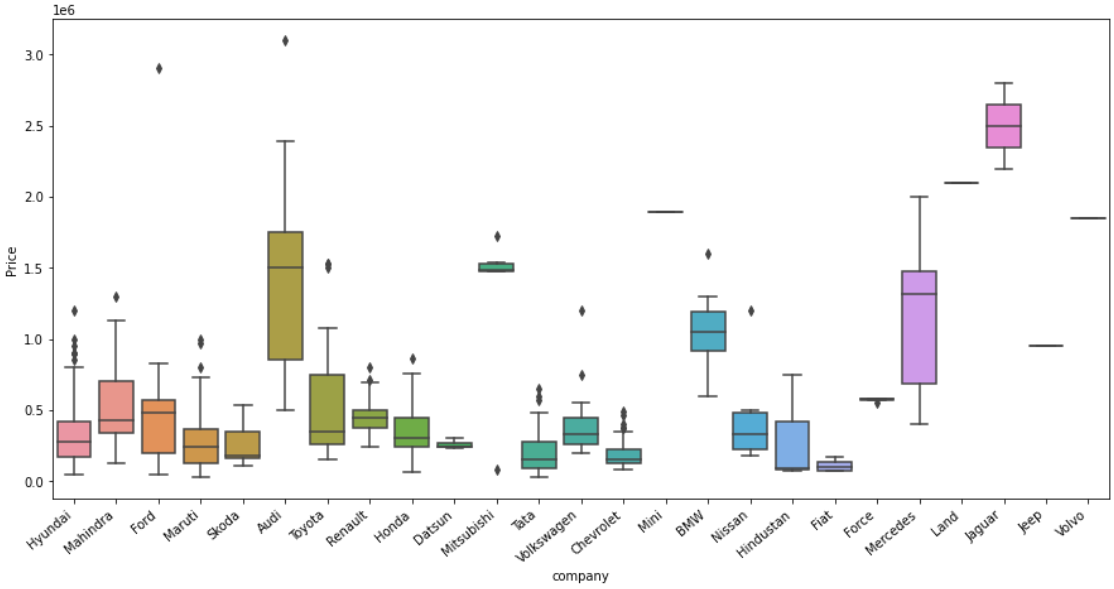
x=car.drop(columns='Price')

y=car['Price']

Now we use seaborn to represent the given data in a graphical manner

import seaborn as sns

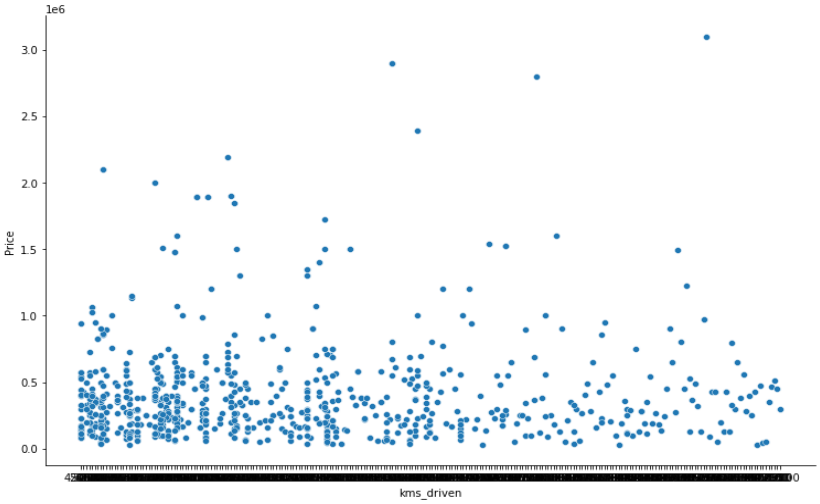
plt.subplots(figsize=(15,7))
ax=sns.boxplot(x='company',y='Price',data=car)
ax.set_xticklabels(ax.get_xticklabels(),rotation=40,ha='right')
plt.show()



Now checking relationship between Kms & Price

sns.relplot(x='kms_driven',y='Price',data=car,height=7,aspect=1.5)

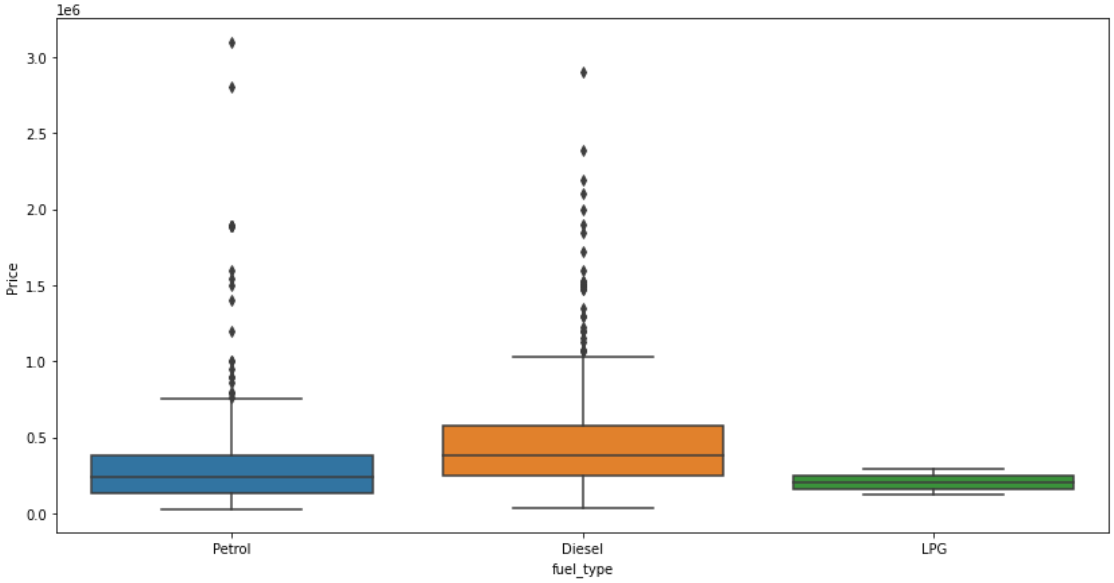
<seaborn.axisgrid.FacetGrid at 0x7f53113a9490



Relationship Of Fueltype & Price

plt.subplots(figsize=(14,7))
sns.boxplot(x='fuel_type',y='Price',data=car)

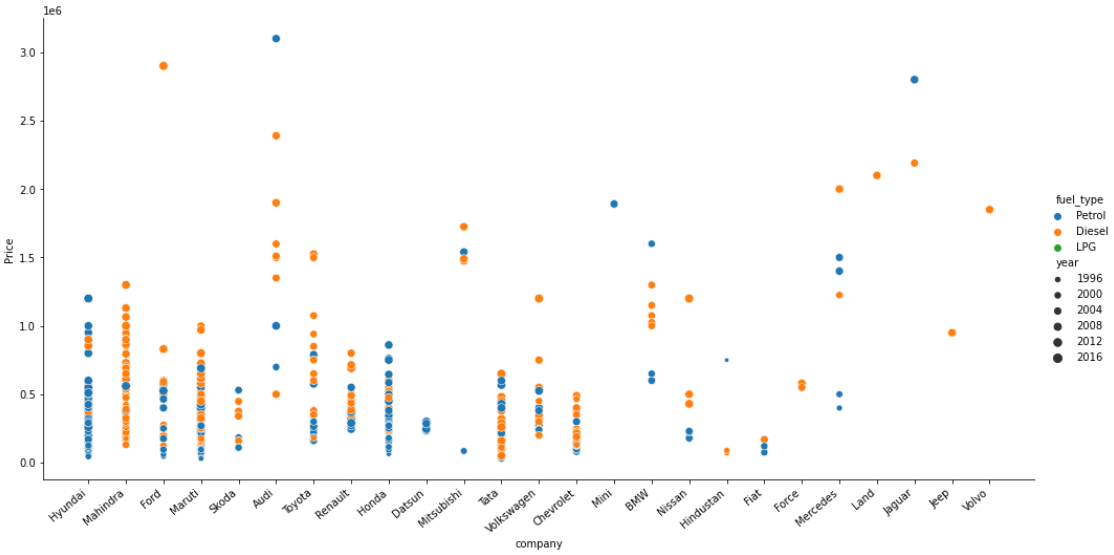
<matplotlib.axes._subplots.AxesSubplot at 0x7f5310ed32d0>



Now Price With Fueltype, Year & Company

ax=sns.relplot(x='company',y='Price',data=car,hue='fuel_type',size='year',height=7,aspect=2)
ax.set_xticklabels(rotation=40,ha='right')

<seaborn.axisgrid.FacetGrid at 0x7f5310dbfa10>



x

	name	company	year	kms_driven	fuel_type
0	*Hyundai Santro Xing*	*Hyundai*	*2007*	*45000*	*Petrol*

```
1       Mahindra Jeep CL550  Mahindra  2006         40    Diesel
2         Hyundai Grand i10   Hyundai  2014      28000    Petrol
3    Ford EcoSport Titanium      Ford  2014      36000    Diesel
4                 Ford Figo      Ford  2012      41000    Diesel
..                      ...       ...   ...        ...       ...
811      Maruti Suzuki Ritz    Maruti  2011      50000    Petrol
812           Tata Indica V2      Tata  2009      30000    Diesel
813    Toyota Corolla Altis    Toyota  2009     132000    Petrol
814           Tata Zest XM      Tata  2018      27000    Diesel
815      Mahindra Quanto C8  Mahindra  2013      40000    Diesel

[815 rows x 5 columns]
```

y

```
0       80000
1      425000
2      325000
3      575000
4      175000
       ...
811    270000
812    110000
813    300000
814    260000
815    390000

Name: Price, Length: 815, dtype: int64
```

Now we have to use the train_test_split & linear regression

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.2)
```

Here r2_score will measures the linear regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
```

Now we have to change X_train & X_test using OHE

```
ohe = OneHotEncoder()
ohe.fit(x[['name','company','fuel_type']])
```

```
OneHotEncoder()
```

Transform X_train & X_test using coloumn transfer & pipeline

Now we have to transform column data using column transfer Using this we only transform the data of name , column & fuel type and rest of the column is passthrough.Also we the pass the categories in OHE

```
ohe.categories_
```

```
[array(['Audi A3 Cabriolet', 'Audi A4 1.8', 'Audi A4 2.0', 'Audi A6 2.0',
       'Audi A8', 'Audi Q3 2.0', 'Audi Q5 2.0', 'Audi Q7', 'BMW 3 Series',
       'BMW 5 Series', 'BMW 7 Series', 'BMW X1', 'BMW X1 sDrive20d',
       'BMW X1 xDrive20d', 'Chevrolet Beat', 'Chevrolet Beat Diesel',
       'Chevrolet Beat LS', 'Chevrolet Beat LT', 'Chevrolet Beat PS',
       'Chevrolet Cruze LTZ', 'Chevrolet Enjoy', 'Chevrolet Enjoy 1.4',
       'Chevrolet Sail 1.2', 'Chevrolet Sail UVA', 'Chevrolet Spark',
       'Chevrolet Spark 1.0', 'Chevrolet Spark LS', 'Chevrolet Spark LT',
       'Chevrolet Tavera LS', 'Chevrolet Tavera Neo', 'Datsun GO T',
       'Datsun Go Plus', 'Datsun Redi GO', 'Fiat Linea Emotion',
       'Fiat Petra ELX', 'Fiat Punto Emotion', 'Force Motors Force',
       'Force Motors One', 'Ford EcoSport', 'Ford EcoSport Ambiente',
       'Ford EcoSport Titanium', 'Ford EcoSport Trend',
       'Ford Endeavor 4x4', 'Ford Fiesta', 'Ford Fiesta SXi', 'Ford Figo',
       'Ford Figo Diesel', 'Ford Figo Duratorq', 'Ford Figo Petrol',
       'Ford Fusion 1.4', 'Ford Ikon 1.3', 'Ford Ikon 1.6',
       'Hindustan Motors Ambassador', 'Honda Accord', 'Honda Amaze',
       'Honda Amaze 1.2', 'Honda Amaze 1.5', 'Honda Brio', 'Honda Brio V',
       'Honda Brio VX', 'Honda City', 'Honda City 1.5', 'Honda City SV',
       'Honda City VX', 'Honda City ZX', 'Honda Jazz S', 'Honda Jazz VX',
       'Honda Mobilio', 'Honda Mobilio S', 'Honda WR V', 'Hyundai Accent',
       'Hyundai Accent Executive', 'Hyundai Accent GLE',
       'Hyundai Accent GLX', 'Hyundai Creta', 'Hyundai Creta 1.6',
       'Hyundai Elantra 1.8', 'Hyundai Elantra SX', 'Hyundai Elite i20',
       'Hyundai Eon', 'Hyundai Eon D', 'Hyundai Eon Era',
       'Hyundai Eon Magna', 'Hyundai Eon Sportz', 'Hyundai Fluidic Verna',
       'Hyundai Getz', 'Hyundai Getz GLE', 'Hyundai Getz Prime',
```

'Hyundai Grand i10', 'Hyundai Santro', 'Hyundai Santro AE',
'Hyundai Santro Xing', 'Hyundai Sonata Transform', 'Hyundai Verna',
'Hyundai Verna 1.4', 'Hyundai Verna 1.6', 'Hyundai Verna Fluidic',
'Hyundai Verna Transform', 'Hyundai Verna VGT',
'Hyundai Xcent Base', 'Hyundai Xcent SX', 'Hyundai i10',
'Hyundai i10 Era', 'Hyundai i10 Magna', 'Hyundai i10 Sportz',
'Hyundai i20', 'Hyundai i20 Active', 'Hyundai i20 Asta',
'Hyundai i20 Magna', 'Hyundai i20 Select', 'Hyundai i20 Sportz',
'Jaguar XE XE', 'Jaguar XF 2.2', 'Jeep Wrangler Unlimited',
'Land Rover Freelander', 'Mahindra Bolero DI',
'Mahindra Bolero Power', 'Mahindra Bolero SLE',
'Mahindra Jeep CL550', 'Mahindra Jeep MM', 'Mahindra KUV100',
'Mahindra KUV100 K8', 'Mahindra Logan', 'Mahindra Logan Diesel',
'Mahindra Quanto C4', 'Mahindra Quanto C8', 'Mahindra Scorpio',
'Mahindra Scorpio 2.6', 'Mahindra Scorpio LX',
'Mahindra Scorpio S10', 'Mahindra Scorpio S4',
'Mahindra Scorpio SLE', 'Mahindra Scorpio SLX',
'Mahindra Scorpio VLX', 'Mahindra Scorpio Vlx',
'Mahindra Scorpio W', 'Mahindra TUV300 T4', 'Mahindra TUV300 T8',
'Mahindra Thar CRDe', 'Mahindra XUV500', 'Mahindra XUV500 W10',
'Mahindra XUV500 W6', 'Mahindra XUV500 W8', 'Mahindra Xylo D2',
'Mahindra Xylo E4', 'Mahindra Xylo E8', 'Maruti Suzuki 800',
'Maruti Suzuki A', 'Maruti Suzuki Alto', 'Maruti Suzuki Baleno',
'Maruti Suzuki Celerio', 'Maruti Suzuki Ciaz',
'Maruti Suzuki Dzire', 'Maruti Suzuki Eeco',
'Maruti Suzuki Ertiga', 'Maruti Suzuki Esteem',
'Maruti Suzuki Estilo', 'Maruti Suzuki Maruti',
'Maruti Suzuki Omni', 'Maruti Suzuki Ritz', 'Maruti Suzuki S',
'Maruti Suzuki SX4', 'Maruti Suzuki Stingray',
'Maruti Suzuki Swift', 'Maruti Suzuki Versa',
'Maruti Suzuki Vitara', 'Maruti Suzuki Wagon', 'Maruti Suzuki Zen',
'Mercedes Benz A', 'Mercedes Benz B', 'Mercedes Benz C',
'Mercedes Benz GLA', 'Mini Cooper S', 'Mitsubishi Lancer 1.8',
'Mitsubishi Pajero Sport', 'Nissan Micra XL', 'Nissan Micra XV',
'Nissan Sunny', 'Nissan Sunny XL', 'Nissan Terrano XL',
'Nissan X Trail', 'Renault Duster', 'Renault Duster 110',
'Renault Duster 110PS', 'Renault Duster 85', 'Renault Duster 85PS',
'Renault Duster RxL', 'Renault Kwid', 'Renault Kwid 1.0',
'Renault Kwid RXT', 'Renault Lodgy 85', 'Renault Scala RxL',
'Skoda Fabia', 'Skoda Fabia 1.2L', 'Skoda Fabia Classic',
'Skoda Laura', 'Skoda Octavia Classic', 'Skoda Rapid Elegance',
'Skoda Superb 1.8', 'Skoda Yeti Ambition', 'Tata Aria Pleasure',
'Tata Bolt XM', 'Tata Indica', 'Tata Indica V2', 'Tata Indica eV2',
'Tata Indigo CS', 'Tata Indigo LS', 'Tata Indigo LX',
'Tata Indigo Marina', 'Tata Indigo eCS', 'Tata Manza',
'Tata Manza Aqua', 'Tata Manza Aura', 'Tata Manza ELAN',
'Tata Nano', 'Tata Nano Cx', 'Tata Nano GenX', 'Tata Nano LX',
'Tata Nano Lx', 'Tata Sumo Gold', 'Tata Sumo Grande',
'Tata Sumo Victa', 'Tata Tiago Revotorq', 'Tata Tiago Revotron',
'Tata Tigor Revotron', 'Tata Venture EX', 'Tata Vista Quadrajet',

'Tata Zest Quadrajet', 'Tata Zest XE', 'Tata Zest XM',
'Toyota Corolla', 'Toyota Corolla Altis', 'Toyota Corolla H2',
'Toyota Etios', 'Toyota Etios G', 'Toyota Etios GD',
'Toyota Etios Liva', 'Toyota Fortuner', 'Toyota Fortuner 3.0',
'Toyota Innova 2.0', 'Toyota Innova 2.5', 'Toyota Qualis',
'Volkswagen Jetta Comfortline', 'Volkswagen Jetta Highline',
'Volkswagen Passat Diesel', 'Volkswagen Polo',
'Volkswagen Polo Comfortline', 'Volkswagen Polo Highline',
'Volkswagen Polo Highline1.2L', 'Volkswagen Polo Trendline',
'Volkswagen Vento Comfortline', 'Volkswagen Vento Highline',
'Volkswagen Vento Konekt', 'Volvo S80 Summum'], dtype=object),
array(['Audi', 'BMW', 'Chevrolet', 'Datsun', 'Fiat', 'Force', 'Ford',
'Hindustan', 'Honda', 'Hyundai', 'Jaguar', 'Jeep', 'Land',
'Mahindra', 'Maruti', 'Mercedes', 'Mini', 'Mitsubishi', 'Nissan',
'Renault', 'Skoda', 'Tata', 'Toyota', 'Volkswagen', 'Volvo'],
dtype=object),

array(['Diesel', 'LPG', 'Petrol'], dtype=object)]

column_trans= make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),remainder='passthrough')

Before linear regression we apply OHE in these coloumn ['name','company','fuel_type']

4. Modeling and Error Analysis

In this we find the error from the given data and then solve those error

Error In The Given Data

-Year has non-year values

-Year object to int

-Price has Ask for price

-Price object to int

-Kms_driven has kms with integer

-Kms_driven has NaN values

-Fuel_type has NaN values

-Keep first 3 words of name

CODE:

```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 892 entries, 0 to 891
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        892 non-null    object
 1   company     892 non-null    object
 2   year        892 non-null    object
 3   Price       892 non-null    object
 4   kms_driven  840 non-null    object
 5   fuel_type   837 non-null    object
```

dtypes: object(6)
memory usage: 41.9+ KB

car['year'].unique()

array(['2007', '2006', '2018', '2014', '2015', '2012', '2013', '2016',
'2010', '2017', '2008', '2011', '2019', '2009', '2005', '2000',
'...', '150k', 'TOUR', '2003', 'r 15', '2004', 'Zest', '/-Rs',
'sale', '1995', 'ara)', '2002', 'SELL', '2001', 'tion', 'odel',
'2 bs', 'arry', 'Eon', 'o...', 'ture', 'emi', 'car', 'able', 'no.',
'd...', 'SALE', 'digo', 'sell', 'd Ex', 'n...', 'e...', 'D...',
', Ac', 'go .', 'k...', 'o c4', 'zire', 'cent', 'Sumo', 'cab',
't xe', 'EV2', 'r...', 'zest'], dtype=object)

In the given data we find some false data present not only in year coloumn but in other coloumn also so first we have to clean the false data and make a new data.

So the issue is found in this data are :

Now we have to start cleaning our data but before that we have to make a backup of the data

backup=car.copy()

car[car['year'].str.isnumeric()]

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80,000	45,000 kms	Petrol
1	Mahindra Jeep CL550 MDI	Mahindra	2006	4,25,000	40 kms	Diesel
2	Maruti Suzuki Alto 800 Vxi	Maruti	2018	Ask For Price	22,000 kms	Petrol

	name	company	year	Price	kms_driven	fuel_type	
3	*Hyundai Grand i10 Magna 1.2 Kappa VTVT*	*Hyundai*	*2014*	*3,25,000*	*28,000 kms*	*Petrol*
4	*Ford EcoSport Titanium 1.5L TDCi*	*Ford*	*2014*	*5,75,000*	*36,000 kms*	*Diesel*
...	*...*	*...*	*...*	*...*	*...*	*...*
886	*Toyota Corolla Altis*	*Toyota*	*2009*	*3,00,000*	*1,32,000 kms*	*Petrol*
888	*Tata Zest XM Diesel*	*Tata*	*2018*	*2,60,000*	*27,000 kms*	*Diesel*
889	*Mahindra Quanto C8*	*Mahindra*	*2013*	*3,90,000*	*40,000 kms*	*Diesel*
890	*Honda Amaze 1.2 E i VTEC*	*Honda*	*2014*	*1,80,000*	*Petrol*	*NaN*
891	*Chevrolet Sail 1.2 LT ABS*	*Chevrolet*	*2014*	*1,60,000*	*Petrol*	*NaN*

842 rows × 6 columns

car=car[car['year'].str.isnumeric()]

car['year'].astype(int)

0 2007

1 2006

2 2018

3 2014

4 2014

...

886 2009

888 2018

889 2013

890 2014

891 2014

Name: year, Length: 842, dtype: int64

car['year']=car['year'].astype(int)

car[car['Price']!="Ask For Price"]

	name	company	year	Price	kms_driven	fuel_type	
0	*Hyundai Santro Xing XO eRLX Euro III*	*Hyundai*	*2007*	*80,000*	*45,000 kms*	*Petrol*
1	*Mahindra Jeep CL550 MDI*	*Mahindra*	*2006*	*4,25,000*	*40 kms*	*Diesel*
3	*Hyundai Grand i10 Magna 1.2 Kappa VTVT*	*Hyundai*	*2014*	*3,25,000*	*28,000 kms*	*Petrol*
4	*Ford EcoSport Titanium 1.5L TDCi*	*Ford*	*2014*	*5,75,000*	*36,000 kms*	*Diesel*
6	*Ford Figo*	*Ford*	*2012*	*1,75,000*	*41,000 kms*	*Diesel*
...	*...*	*...*	*...*	*...*	*...*	*...*

	name	company	year	Price	kms_driven	fuel_type	
886	Toyota Corolla Altis	Toyota	2009	3,00,000	1,32,000 kms	Petrol
888	Tata Zest XM Diesel	Tata	2018	2,60,000	27,000 kms	Diesel
889	Mahindra Quanto C8	Mahindra	2013	3,90,000	40,000 kms	Diesel
890	Honda Amaze 1.2 E i VTEC	Honda	2014	1,80,000	Petrol	NaN
891	Chevrolet Sail 1.2 LT ABS	Chevrolet	2014	1,60,000	Petrol	NaN

819 rows × 6 columns

```
car=car[car['Price']!="Ask For Price"]
```

```
car['Price'].str.replace(',','').astype(int)
```

```
0 80000
1 425000
3 325000
4 575000
6 175000
...
886 300000
888 260000
889 390000
890 180000
891 160000
```

Name: Price, Length: 819, dtype: int64

car['Price']=car['Price'].str.replace(',','').astype(int)

car['kms_driven'].str.split(' ').str.get(0).str.replace(',','')

0 45000

1 40

3 28000

4 36000

6 41000

...

886 132000

888 27000

889 40000

890 Petrol

891 Petrol

Name: kms_driven, Length: 819, dtype: object

car['kms_driven']=car['kms_driven'].str.split(' ').str.get(0).str.replace(',','')

car[car['kms_driven'].str.isnumeric()]

	name	company	year	Price	kms_driven	fuel_type
0	*Hyundai Santro Xing XO eRLX Euro III*	*Hyundai*	*2007*	*80000*	*45000*	*Petrol*
1	*Mahindra Jeep CL550 MDI*	*Mahindra*	*2006*	*425000*	*40*	*Diesel*

	name	company	year	Price	kms_driven	fuel_type
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	325000	28000	Petrol
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	575000	36000	Diesel
6	Ford Figo	Ford	2012	175000	41000	Diesel
...	...	...	...	...	...	...
883	Maruti Suzuki Ritz VXI ABS	Maruti	2011	270000	50000	Petrol
885	Tata Indica V2 DLE BS III	Tata	2009	110000	30000	Diesel
886	Toyota Corolla Altis	Toyota	2009	300000	132000	Petrol
888	Tata Zest XM Diesel	Tata	2018	260000	27000	Diesel
889	Mahindra Quanto C8	Mahindra	2013	390000	40000	Diesel

817 rows × 6 columns

```
car=car[car['kms_driven'].str.isnumeric()]
```

```
car[~car['fuel_type'].isna()]
```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80000	45000	Petrol
1	Mahindra Jeep CL550 MDI	Mahindra	2006	425000	40	Diesel

	name	company	year	Price	kms_driven	fuel_type
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	325000	28000	Petrol
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	575000	36000	Diesel
6	Ford Figo	Ford	2012	175000	41000	Diesel
...	...	...	...	...	...	...
883	Maruti Suzuki Ritz VXI ABS	Maruti	2011	270000	50000	Petrol
885	Tata Indica V2 DLE BS III	Tata	2009	110000	30000	Diesel
886	Toyota Corolla Altis	Toyota	2009	300000	132000	Petrol
888	Tata Zest XM Diesel	Tata	2018	260000	27000	Diesel
889	Mahindra Quanto C8	Mahindra	2013	390000	40000	Diesel

816 rows × 6 columns

```
car=car[~car['fuel_type'].isna()]
```

```
car['name'].str.split(' ').str.slice(0,3).str.join(' ')
```

0 Hyundai Santro Xing

1 Mahindra Jeep CL550

3 Hyundai Grand i10

4 Ford EcoSport Titanium

6 Ford Figo

...

883 Maruti Suzuki Ritz

885 Tata Indica V2

886 Toyota Corolla Altis

888 Tata Zest XM

889 Mahindra Quanto C8

Name: name, Length: 816, dtype: object

```
car['name']=car['name'].str.split(' ').str.slice(0,3).str.join(' ')
```

```
car.reset_index(drop=True)
```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	80000	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	425000	40	Diesel
2	Hyundai Grand i10	Hyundai	2014	325000	28000	Petrol
3	Ford EcoSport Titanium	Ford	2014	575000	36000	Diesel
4	Ford Figo	Ford	2012	175000	41000	Diesel
...	...	...	...	...	...	...
811	Maruti Suzuki Ritz	Maruti	2011	270000	50000	Petrol
812	Tata Indica V2	Tata	2009	110000	30000	Diesel
813	Toyota Corolla Altis	Toyota	2009	300000	132000	Petrol
814	Tata Zest XM	Tata	2018	260000	27000	Diesel
815	Mahindra Quanto C8	Mahindra	2013	390000	40000	Diesel

816 rows × 6 columns

car=car.reset_index(drop=True)

Now i have to store the clean data

car.to_csv('Cleaned Car.csv')

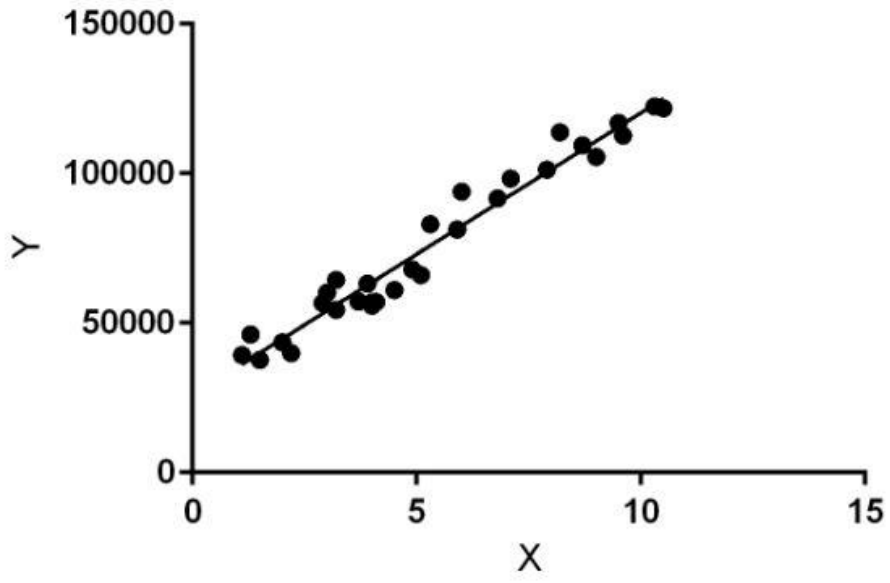
5. *Advanced Modeling and Feature Engineering*

The model is used in this project is Linear Regression Model

Linear Regression :

Linear regression is a machine learning algorithm based on supervised learning.

It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.



Some other thing used in this model is :

- *OHE : One Hot Encoder*

- *Coefficient Of Determination (r2 Linear Regression)*

  One Hot Encoder (OHE): A one-hot is a group of bits among which the legal combinations of values are only those with a single high bit and all the others low. A similar implementation in which all bits are '1' except one '0' is sometimes called one-cold.

  R2 Linear Regression: R-Squared (R^2 or the coefficient of determination) is a statistical measure that determines the proportion of variance in the dependent variable that can be explained by the independent variable.

Formula : R2 = 1-RSS/TSS

R2 = Coefficient Of Determination *TSS = Total Sum Of Square*
RSS = Sum Of Square Of Residual

Code:

X

	name	company	year	kms_driven	fuel_type
0	*Hyundai Santro Xing*	*Hyundai*	*2007*	*45000*	*Petrol*
1	*Mahindra Jeep CL550*	*Mahindra*	*2006*	*40*	*Diesel*
2	*Hyundai Grand i10*	*Hyundai*	*2014*	*28000*	*Petrol*
3	*Ford EcoSport Titanium*	*Ford*	*2014*	*36000*	*Diesel*
4	*Ford Figo*	*Ford*	*2012*	*41000*	*Diesel*
...	*...*	*...*	*...*	*...*	*...*
811	*Maruti Suzuki Ritz*	*Maruti*	*2011*	*50000*	*Petrol*
812	*Tata Indica V2*	*Tata*	*2009*	*30000*	*Diesel*
813	*Toyota Corolla Altis*	*Toyota*	*2009*	*132000*	*Petrol*
814	*Tata Zest XM*	*Tata*	*2018*	*27000*	*Diesel*
815	*Mahindra Quanto C8*	*Mahindra*	*2013*	*40000*	*Diesel*

815 rows × 5 columns

Y

0 80000

1 425000

2 325000

3 575000

4 175000

...

811 270000

812 110000

813 300000

814 260000

815 390000

Name: Price, Length: 815, dtype: int64

Now we have to use the train_test_split & linear regression

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.2)
```

Here r2_score will measures the linear regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
```

Now we have to change X_train & X_test using OHE

ohe = OneHotEncoder()

ohe.fit(x[['name','company','fuel_type']])

Transform X_train & X_test using coloumn transfer & pipeline

Now we have to transform column data using column transfer Using this we only transform the data of name , column & fuel type and rest of the column is passthrough.Also we the pass the categories in OHE

*ohe.categories_*

'Ford Endeavor 4x4', 'Ford Fiesta', 'Ford Fiesta SXi', 'Ford Figo', 'Ford Figo Diesel', 'Ford Figo Duratorq', 'Ford Figo Petrol', 'Ford Fusion 1.4', 'Ford Ikon 1.3', 'Ford Ikon 1.6', 'Hindustan Motors Ambassador', 'Honda Accord', 'Honda Amaze', 'Honda Amaze 1.2', 'Honda Amaze 1.5', 'Honda Brio', 'Honda Brio V', 'Honda Brio VX', 'Honda City', 'Honda City 1.5', 'Honda City SV', 'Honda City VX', 'Honda City ZX', 'Honda Jazz S', 'Honda Jazz VX', 'Honda Mobilio', 'Honda Mobilio S', 'Honda WR V', 'Hyundai Accent', 'Hyundai Accent Executive', 'Hyundai Accent GLE', 'Hyundai Accent GLX', 'Hyundai Creta', 'Hyundai Creta 1.6', 'Hyundai Elantra 1.8', 'Hyundai Elantra SX', 'Hyundai Elite i20', 'Hyundai Eon', 'Hyundai Eon D', 'Hyundai Eon Era', 'Hyundai Eon Magna', 'Hyundai Eon Sportz', 'Hyundai Fluidic Verna', 'Hyundai Getz', 'Hyundai Getz GLE', 'Hyundai Getz Prime', 'Hyundai Grand i10', 'Hyundai Santro', 'Hyundai Santro AE', 'Hyundai Santro Xing', 'Hyundai Sonata Transform', 'Hyundai Verna', 'Hyundai Verna 1.4', 'Hyundai Verna 1.6', 'Hyundai Verna Fluidic', 'Hyundai Verna Transform', 'Hyundai Verna VGT', 'Hyundai Xcent Base', 'Hyundai Xcent SX', 'Hyundai i10', 'Hyundai i10 Era', 'Hyundai i10 Magna', 'Hyundai i10 Sportz', 'Hyundai i20', 'Hyundai i20 Active', 'Hyundai i20 Asta', 'Hyundai i20 Magna', 'Hyundai i20 Select', 'Hyundai i20 Sportz', 'Jaguar XE XE', 'Jaguar XF 2.2', 'Jeep Wrangler Unlimited', 'Land Rover Freelander', 'Mahindra Bolero DI', 'Mahindra Bolero Power', 'Mahindra Bolero SLE', 'Mahindra Jeep CL550', 'Mahindra Jeep MM', 'Mahindra KUV100', 'Mahindra KUV100 K8', 'Mahindra Logan', 'Mahindra Logan Diesel', 'Mahindra Quanto C4', 'Mahindra Quanto C8', 'Mahindra Scorpio', 'Mahindra Scorpio 2.6', 'Mahindra Scorpio LX', 'Mahindra Scorpio S10', 'Mahindra Scorpio S4', 'Mahindra Scorpio SLE', 'Mahindra Scorpio SLX', 'Mahindra Scorpio VLX', 'Mahindra Scorpio Vlx', 'Mahindra Scorpio W', 'Mahindra TUV300 T4', 'Mahindra TUV300 T8', 'Mahindra Thar CRDe', 'Mahindra XUV500', 'Mahindra XUV500 W10', 'Mahindra XUV500 W6', 'Mahindra XUV500 W8', 'Mahindra Xylo D2', 'Mahindra Xylo E4',

'Mahindra Xylo E8', 'Maruti Suzuki 800', 'Maruti Suzuki A', 'Maruti Suzuki Alto', 'Maruti Suzuki Baleno', 'Maruti Suzuki Celerio', 'Maruti Suzuki Ciaz', 'Maruti Suzuki Dzire', 'Maruti Suzuki Eeco', 'Maruti Suzuki Ertiga', 'Maruti Suzuki Esteem', 'Maruti Suzuki Estilo', 'Maruti Suzuki Maruti', 'Maruti Suzuki Omni', 'Maruti Suzuki Ritz', 'Maruti Suzuki S', 'Maruti Suzuki SX4', 'Maruti Suzuki Stingray', 'Maruti Suzuki Swift', 'Maruti Suzuki Versa', 'Maruti Suzuki Vitara', 'Maruti Suzuki Wagon', 'Maruti Suzuki Zen', 'Mercedes Benz A', 'Mercedes Benz B', 'Mercedes Benz C', 'Mercedes Benz GLA', 'Mini Cooper S', 'Mitsubishi Lancer 1.8', 'Mitsubishi Pajero Sport', 'Nissan Micra XL', 'Nissan Micra XV', 'Nissan Sunny', 'Nissan Sunny XL', 'Nissan Terrano XL', 'Nissan X Trail', 'Renault Duster', 'Renault Duster 110', 'Renault Duster 110PS', 'Renault Duster 85', 'Renault Duster 85PS', 'Renault Duster RxL', 'Renault Kwid', 'Renault Kwid 1.0', 'Renault Kwid RXT', 'Renault Lodgy 85', 'Renault Scala RxL', 'Skoda Fabia', 'Skoda Fabia 1.2L', 'Skoda Fabia Classic', 'Skoda Laura', 'Skoda Octavia Classic', 'Skoda Rapid Elegance', 'Skoda Superb 1.8', 'Skoda Yeti Ambition', 'Tata Aria Pleasure', 'Tata Bolt XM', 'Tata Indica', 'Tata Indica V2', 'Tata Indica eV2', 'Tata Indigo CS', 'Tata Indigo LS', 'Tata Indigo LX', 'Tata Indigo Marina', 'Tata Indigo eCS', 'Tata Manza', 'Tata Manza Aqua', 'Tata Manza Aura', 'Tata Manza ELAN', 'Tata Nano', 'Tata Nano Cx', 'Tata Nano GenX', 'Tata Nano LX', 'Tata Nano Lx', 'Tata Sumo Gold', 'Tata Sumo Grande', 'Tata Sumo Victa', 'Tata Tiago Revotorq', 'Tata Tiago Revotron', 'Tata Tigor Revotron', 'Tata Venture EX', 'Tata Vista Quadrajet', 'Tata Zest Quadrajet', 'Tata Zest XE', 'Tata Zest XM', 'Toyota Corolla', 'Toyota Corolla Altis', 'Toyota Corolla H2', 'Toyota Etios', 'Toyota Etios G', 'Toyota Etios GD', 'Toyota Etios Liva', 'Toyota Fortuner', 'Toyota Fortuner 3.0', 'Toyota Innova 2.0', 'Toyota Innova 2.5', 'Toyota Qualis', 'Volkswagen Jetta Comfortline', 'Volkswagen Jetta Highline', 'Volkswagen Passat Diesel', 'Volkswagen Polo', 'Volkswagen Polo Comfortline', 'Volkswagen Polo Highline', 'Volkswagen Polo Highline1.2L', 'Volkswagen Polo Trendline', 'Volkswagen Vento Comfortline', 'Volkswagen Vento Highline', 'Volkswagen Vento Konekt', 'Volvo S80 Summum'], dtype=object), array(['Audi', 'BMW', 'Chevrolet', 'Datsun', 'Fiat', 'Force', 'Ford', 'Hindustan', 'Honda', 'Hyundai', 'Jaguar', 'Jeep', 'Land', 'Mahindra', 'Maruti', 'Mercedes', 'Mini', 'Mitsubishi', 'Nissan', 'Renault', 'Skoda', 'Tata', 'Toyota', 'Volkswagen', 'Volvo'], dtype=object), array(['Diesel', 'LPG', 'Petrol'], dtype=object)]

column_trans= make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),remainder='passthrough')

Before linear regression we apply OHE in these coloumn ['name','company','fuel_type']

Now we have to train linear regression model

lr=LinearRegression()

Now the input data is given to the pipeline first it goes to the column transformer then OHE will perform then it transform the given column.Now all the column & transform column will be given to the linear regression

```
pipe=make_pipeline(column_trans,lr)
```

```
pipe.fit(x_train,y_train)
```

```
Pipeline(steps=[('columntransformer', ColumnTransformer(remainder='passthrough',
transformers=[('onehotencoder', OneHotEncoder(categories=[array(['Audi A3
Cabriolet', 'Audi A4 1.8', 'Audi A4 2.0', 'Audi A6 2.0', 'Audi A8', 'Audi Q3 2.0', 'Audi Q5
2.0', 'Audi Q7', 'BMW 3 Series', 'BMW 5 Series', 'BMW 7 Series', 'BMW X1', 'BMW
X1 sDrive20d', 'BMW X1 xDrive20d', 'Chevrolet Beat', 'Chevrolet Beat... array(['Audi',
'BMW', 'Chevrolet', 'Datsun', 'Fiat', 'Force', 'Ford', 'Hindustan', 'Honda', 'Hyundai',
'Jaguar', 'Jeep', 'Land', 'Mahindra', 'Maruti', 'Mercedes', 'Mini', 'Mitsubishi', 'Nissan',
'Renault', 'Skoda', 'Tata', 'Toyota', 'Volkswagen', 'Volvo'], dtype=object),
array(['Diesel', 'LPG', 'Petrol'], dtype=object)]), ['name', 'company', 'fuel_type'])])),
('linearregression', LinearRegression())])
```

Now we have to predict & store in y

```
y_pred=pipe.predict(x_test)
```

```
y_pred
```

```
array([ 230144.46315668, 611844.47045219, 253171.72290007, 231291.71486861,
589830.312828 , 499020.78852601, 373431.94867907, 389474.61146253,
158727.96561675, 226574.1395564 , 18946.3488765 , 343287.29680047,
1191041.65527796, 215892.72594278, 373541.52363361, 110558.2356806 ,
1891895.63089208, 228662.71713552, 6139.78838712, 36665.14657372,
1410877.77380855, 1480235.07043961, 199962.21432345, -79286.7112126 ,
543746.49162824, 280818.85381555, 403150.48335501, 296817.51791763,
276080.30308063, 463338.28811035, 612972.00592937, 161871.40329273,
306264.34511222, 231175.93245289, 242613.48514084, 664560.70896006,
217602.20245635, 268214.10611541, 208136.93354911, 217602.20245635,
110488.76623117, 154925.43739358, 503329.58843321, 127658.53767359,
698611.88805933, 360699.83292719, 468008.34019004, 379411.44997381,
312780.04801786, 446663.93492596, 143835.49639098, 15865.05539931,
237439.75352297, 305310.07358013, 203509.07247338, 34218.05660992,
377282.25289665, 252231.62925474, 360398.79864634, 224185.43146649,
486261.07326125, 273789.38839305, 216622.22830994, 314555.75024258,
```

65643.81687953, 1891895.63089208, 296433.62369916, 384370.88480105, 299842.44188115, 223793.50538404, 462478.15520082, 87284.76024929, 347367.09168233, 433545.50647558, 409532.36979257, 216622.22830994, 232587.158746 , 363899.34382997, 259011.47248983, 275275.44053997, 271963.26626188, 858493.24572294, 700235.50340249, 576297.28530885, 270207.49957664, 363735.8320663 , 327159.95789728, 331863.6593663 , 121415.70786721, 246231.28738835, 223050.43788962, 300658.04543772, 76327.26212968, 764469.23001864, 303058.18304952, 312865.74797944, 422972.46852032, 176902.2568452 , 295213.44627115, 485195.8750367 , 577956.85306913, 602467.42285176, 103890.74942932, 209347.19291448, 422527.65885902, 453413.66715815, 391683.51629346, 775403.40565592, 289708.05496385, 1213504.65379928, 318139.71339422, 232587.158746 , 320662.20289716, 456104.5324366 , 203555.38543966, 478947.29728145, 235758.15568691, 488895.27546174, 1644610.67612292, 366293.72643574, 299661.2456451 , -92969.35941643, 1891895.63089208, 257861.28384838, 641465.51485743, 729157.48907416, 89588.05751197, 348930.92595612, 197196.57065558, 272728.95589896, 234919.41537943, -39457.74666665, 450322.39399081, 304607.84971943, 39776.93111869, 727739.87584678, 281707.19366673, 126104.59871583, 440165.35000399, 557177.66740492, 723092.16213723, 472215.74302682, 672468.26809531, 335174.14023079, 168104.12945304, 592293.94231388, 335174.14023079, 542923.46249329, 272125.36164388, 1525415.78879682, 167317.38017057, 580405.50919691, 1242535.0773416 , 226574.1395564 , 636036.74882163, 321323.00499769, 358621.13208462, 1232716.30467384, 186966.46720908, 810812.89049307, 282411.13110717, 277167.12772009, 76257.79268026])

Now we have to see the r2_score //r2_score = measure linear regression

r2_score(y_test,y_pred)

0.657937638603565

In Different train and test split data we get different r2 score Then we perform random r2 score in train and test split data & choose the data which has maximum score.

```
for i in range(10):
  x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.2)
```

```
  lr=LinearRegression()
  pipe=make_pipeline(column_trans,lr)
  pipe.fit(x_train,y_train)
  y_pred=pipe.predict(x_test)
  print(r2_score(y_test,y_pred), i)
```

```
0.6604513854128435 0
0.7507650461605244 1
0.6024190275339651 2
0.7708743114289502 3
0.5567281375008425 4
0.5173532834744307 5
0.7453733606751184 6
0.4554845105596823 7
0.8074409480015773 8
0.5910794492084795 9
```

Now We use random_state=i to getting the highest value. After that we have to store the value to storing the value we use score & then append the value

```
score=[]
for i in range(1000):
  x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.2,random_state=i)
  lr=LinearRegression()
  pipe=make_pipeline(column_trans,lr)
  pipe.fit(x_train,y_train)
  y_pred=pipe.predict(x_test)
  score.append(r2_score(y_test,y_pred))
```

Now we have to check the greatest r2 score

np.argmax(score)

661

Checking greatest Score

score[np.argmax(score)]

0.889770931767991

Checking random r2 score

x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.2,random_state=np.argmax(score))

lr=LinearRegression()

pipe=make_pipeline(column_trans,lr)

pipe.fit(x_train,y_train)

y_pred=pipe.predict(x_test)

r2_score(y_test,y_pred)

0.889770931767991

Now we have to import pickle

import pickle

pickle.dump(pipe,open('LinearRegressionModel.pkl','wb'))

pipe.predict(pd.DataFrame([['Jaguar XE XE','Jaguar',2016,8500,'Petrol']],columns=['name','company','year','kms_driven','fuel_type']))

array([2801297.40028831])

Now model is found

6. Deployment and Productionization

The final phase of this project.is Deployment, here we are deploying the whole machine learning pipeline into a production system, into a real-time scenario.

In this final stage we need to deploy this machine learning pipeline to put of research available to end users for use. The model will be deployed in real world scenario where it takes continuous raw input from real world and predict the output.

Prediction Page Building:

In this step the tool we are using to build our prediction page is HTMl . With the help of Html we can create the front end of the car prediction page where the car price is predicted.

HTML CODE :

```
<!DOCTYPE html>
<html lang="en">
<head xmlns="http://www.w3.org/1999/xhtml">
    <meta charset="UTF-8">
    <title>Car Price Predictor</title>
    <link rel="stylesheet" href="static/css/style.css">
    <link rel="stylesheet" type="text/css"
          href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.11.2/css/all.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
            integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
            crossorigin="anonymous"></script>

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
          integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7S
k" crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js"></script>
```

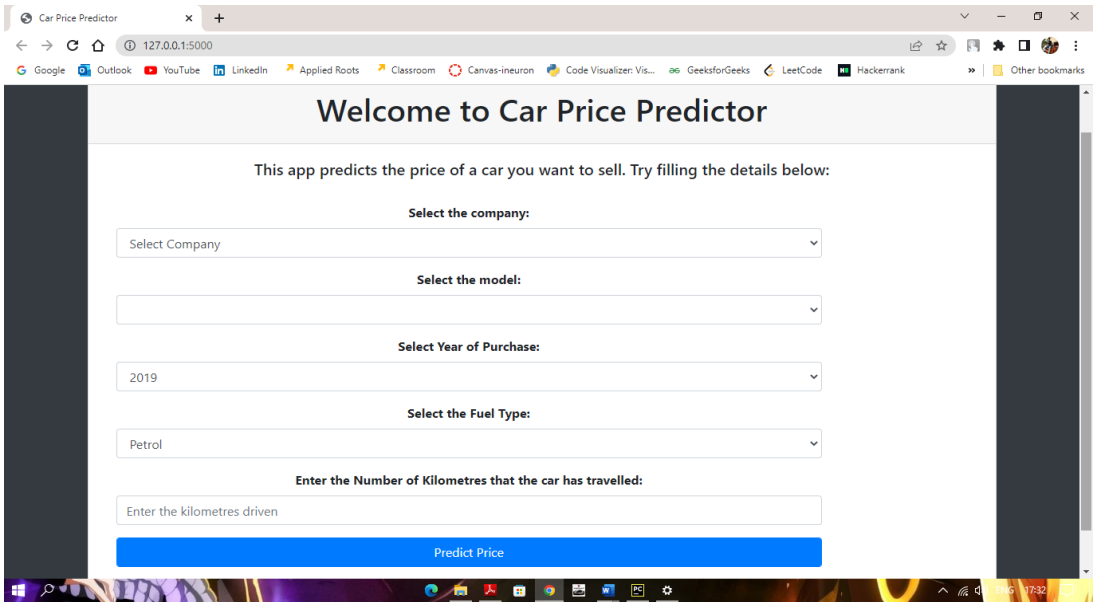
```
</head>
<body class="bg-dark">

<div class="container">
   <div class="row">
      <div class="card mt-50" style="width: 100%; height: 100%">
         <div class="card-header" style="text-align: center">
            <h1>Welcome to Car Price Predictor</h1>
         </div>
         <div class="card-body">
            <div class="col-12" style="text-align: center">
               <h5>This app predicts the price of a car you want to sell. Try filling the
details below: </h5>
            </div>
            <br>
            <form method="post" accept-charset="utf-8" name="Modelform">
               <div class="col-md-10 form-group" style="text-align: center">
                  <label><b>Select the company:</b> </label><br>
                  <select class="selectpicker form-control" id="company"
name="company" required="1"
                        onchange="load_car_models(this.id,'car_models')">
                     {% for company in companies %}
                     <option value="{{ company }}">{{ company }}</option>
                     {% endfor %}
                  </select>
               </div>
               <div class="col-md-10 form-group" style="text-align: center">
                  <label><b>Select the model:</b> </label><br>
                  <select class="selectpicker form-control" id="car_models"
name="car_models" required="1">
                  </select>
               </div>
               <div class="col-md-10 form-group" style="text-align: center">
                  <label><b>Select Year of Purchase:</b> </label><br>
                  <select class="selectpicker form-control" id="year" name="year"
required="1">
                     {% for year in years %}
                     <option value="{{ year }}">{{ year }}</option>
                     {% endfor %}
                  </select>
               </div>
               <div class="col-md-10 form-group" style="text-align: center">
                  <label><b>Select the Fuel Type:</b> </label><br>
                  <select class="selectpicker form-control" id="fuel_type"
name="fuel_type" required="1">
                     {% for fuel in fuel_types %}
```

```
                    <option value="{{ fuel }}">{{ fuel }}</option>
                    {% endfor %}
                </select>
            </div>
            <div class="col-md-10 form-group" style="text-align: center">
                <label><b>Enter the Number of Kilometres that the car has
travelled:</b> </label><br>
                <input type="text" class="form-control" id="kilo_driven"
name="kilo_driven"
                    placeholder="Enter the kilometres driven ">
            </div>
            <div class="col-md-10 form-group" style="text-align: center">
                <button  class="btn btn-primary form-control"
onclick="send_data()">Predict Price</button>
            </div>
        </form>
        <br>
        <div class="row">
            <div class="col-12" style="text-align: center">
                <h4><span id="prediction"></span></h4>
            </div>
        </div>
    </div>
  </div>
</div>

<script>

  function load_car_models(company_id,car_model_id)
  {
    var company=document.getElementById(company_id);
    var car_model= document.getElementById(car_model_id);
    console.log(company.value);
    car_model.value="";
    car_model.innerHTML="";
    {% for company in companies %}
        if( company.value == "{{ company }}")
        {
            {% for model in car_models %}
                {% if company in model %}

                    var newOption= document.createElement("option");
                    newOption.value="{{ model }}";
                    newOption.innerHTML="{{ model }}";
                    car_model.options.add(newOption);
```

```
                {% endif %}
            {% endfor %}
        }
    {% endfor %}
}

function form_handler(event) {
    event.preventDefault(); // Don't submit the form normally
}
function send_data()
{
    document.querySelector('form').addEventListener("submit",form_handler);

    var fd=new FormData(document.querySelector('form'));

    var xhr= new XMLHttpRequest({mozSystem: true});

    xhr.open('POST','/predict',true);
    document.getElementById('prediction').innerHTML="Wait! Predicting Price.....";
    xhr.onreadystatechange = function(){
        if(xhr.readyState == XMLHttpRequest.DONE){
            document.getElementById('prediction').innerHTML="Prediction:
₹"+xhr.responseText;

        }
    };

    xhr.onload= function(){};

    xhr.send(fd);
}
</script>
</body>
</html>
```

Output :

- ***Without Prediction***



-With Prediction