MAHATMA EDUCATION SOCIETY’S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE
(Autonomous)

NEW PANVEL

PROJECT REPORT ON

“ Used Car Price Prediction”

IN PARTIAL FULFILLMENT OF

MASTER OF SCIENCE DATA ANALYTICS PART - II

SEMESTER III – 2025-26

PROJECT GUIDE
Prof. Omkar Sherkhane

SUBMITTED BY: Sushant Kishan Rathod

Mahatma Education Society's

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE

(Autonomous)

Re-accredited "A" Grade by NAAC (3rd Cycle)

Project Completion Certificate

THIS IS TO CERTIFY THAT

Sushant Kishan Rathod

of **M.Sc. Data Analytics Part - II** has completed the project titled **Used Car Price Prediction** of subject **Machine Learning** under our guidance and supervision during the academic year 2025-26 in the department of Data Analytics.

Project Guide Course Coordinator Head of the Department

Pillai College of Arts, Commerce & Science (Autonomous)
New Panvel

Introduction

A used car, a pre-owned vehicle, or a secondhand car, is a vehicle that has previously had one or more retail owners. Used cars are sold through a variety of outlets, including franchise and independent car dealers, rental car companies, buy here pay here dealerships, leasing offices, auctions, and private party sales. Some car retailers offer "no-haggle prices," "certified" used cars, and extended service plans or warranties.

Used car pricing reports typically produce three forms of the pricing information.

- **Dealer or retail price is the price expected to pay if buying from a licensed new-car or used-car dealer.**
- **Dealer trade-in price or wholesale price is the price a shopper should expect to receive from a dealer if trading in a car. This is also the price that a dealer will typically pay for a car at a dealer wholesale auction.**
- **Private-party price is the price expected to pay if buying from an individual. A private-party seller is hoping to get more money than they would with a trade-in to a dealer. A private-party buyer is hoping to pay less than the dealer retail price.**

TOOLS USED :

- **Jupyter Notebook**
- **Visual Studio (Flask)**

COLUMN NAMES AND DESCRIPTION :

Column Name	Column Description
Name	This column contains name of the car
Company	This column contains details about company/ brand of the car
Year	This column contain Manufacturing year of the car
Price	This column contains the price of the car
Kms_driven	This column contains the kms that the car has driven
Fuel_type	The column contain the fuel type of the car

In [2]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
%matplotlib inline
mpl.style.use('ggplot')
```

In [3]:
```python
car=pd.read_csv('cars.csv')
```

In [4]:
```python
car.head()
```

Out[4]:

	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
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

In [5]:
```python
car.shape
```

Out[5]: (5352, 6)

In [6]:
```python
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5352 entries, 0 to 5351
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        5352 non-null   object
 1   company     5352 non-null   object
 2   year        5352 non-null   object
 3   Price       5352 non-null   object
 4   kms_driven  5040 non-null   object
 5   fuel_type   5022 non-null   object
dtypes: object(6)
memory usage: 251.0+ KB
```

Type *Markdown* and LaTeX: α^2

In [7]:
```python
car.tail()
```

Out[7]:

	name	company	year	Price	kms_driven	fuel_type
5347	Ta	Tara	zest	3,10,000	NaN	NaN
5348	Tata Zest XM Diesel	Tata	2018	2,60,000	27,000 kms	Diesel
5349	Mahindra Quanto C8	Mahindra	2013	3,90,000	40,000 kms	Diesel
5350	Honda Amaze 1.2 E i VTEC	Honda	2014	1,80,000	Petrol	NaN
5351	Chevrolet Sail 1.2 LT ABS	Chevrolet	2014	1,60,000	Petrol	NaN

Type *Markdown* and LaTeX: α^2

Cleaning Data

year has many non-year values

In [8]:
```
car=car[car['year'].str.isnumeric()]
```

year is in object. Change to integer

In [9]:
```
car['year']=car['year'].astype(int)
```

Price has Ask for Price

In [10]:
```
car=car[car['Price']!='Ask For Price']
```

Price has commas in its prices and is in object

In [11]:
```
car['Price']=car['Price'].str.replace(',','').astype(int)
```

kms_driven has object values with kms at last.

In [12]:
```
car['kms_driven']=car['kms_driven'].str.split().str.get(0).str.replace(',','')
```

It has nan values and two rows have 'Petrol' in them

In [13]:
```
car=car[car['kms_driven'].str.isnumeric()]
```

In [14]:
```
car['kms_driven']=car['kms_driven'].astype(int)
```

fuel_type has nan values

In [15]:
```
car=car[~car['fuel_type'].isna()]
```

In [16]:
```
car.shape
```

Out[16]:
```
(4896, 6)
```

Company does not need any cleaning now. Changing car names. Keeping only the first three words

In [17]:
```
car['name']=car['name'].str.split().str.slice(start=0,stop=3).str.join(' ')
```

Resetting the index of the final cleaned data

In [18]:
```
car=car.reset_index(drop=True)
```

Cleaned Data

In [19]:
```
car
```

Out[19]:

	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
...	...	...	...	...	...	...
4891	Maruti Suzuki Ritz	Maruti	2011	270000	50000	Petrol
4892	Tata Indica V2	Tata	2009	110000	30000	Diesel
4893	Toyota Corolla Altis	Toyota	2009	300000	132000	Petrol
4894	Tata Zest XM	Tata	2018	260000	27000	Diesel
4895	Mahindra Quanto C8	Mahindra	2013	390000	40000	Diesel

4896 rows × 6 columns

In [20]:
```
car.to_csv('Cleaned_Car_data.csv')
```

In [21]:
```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4896 entries, 0 to 4895
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        4896 non-null   object
 1   company     4896 non-null   object
 2   year        4896 non-null   int32 
 3   Price       4896 non-null   int32 
 4   kms_driven  4896 non-null   int32 
 5   fuel_type   4896 non-null   object
dtypes: int32(3), object(3)
memory usage: 172.3+ KB
```

In [22]:
```
car.describe(include='all')
```

Out[22]:

	name	company	year	Price	kms_driven	fuel_type
count	4896	4896	4896.000000	4.896000e+03	4896.000000	4896
unique	254	25	NaN	NaN	NaN	3
top	Maruti Suzuki Swift	Maruti	NaN	NaN	NaN	Petrol
freq	306	1326	NaN	NaN	NaN	2568
mean	NaN	NaN	2012.444853	4.117176e+05	46275.531863	NaN
std	NaN	NaN	4.000948	4.749417e+05	34279.907007	NaN
min	NaN	NaN	1995.000000	3.000000e+04	0.000000	NaN
25%	NaN	NaN	2010.000000	1.750000e+05	27000.000000	NaN
50%	NaN	NaN	2013.000000	2.999990e+05	41000.000000	NaN
75%	NaN	NaN	2015.000000	4.912500e+05	56818.500000	NaN
max	NaN	NaN	2019.000000	8.500003e+06	400000.000000	NaN

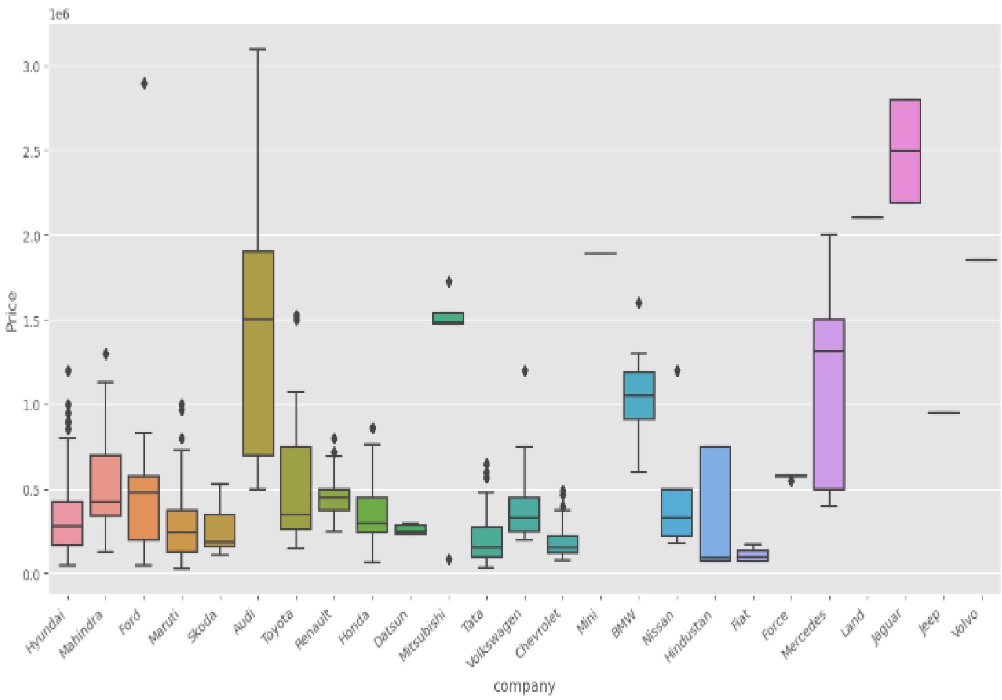
Checking relationship of Company with Price

In [24]:
```
car['company'].unique()
```

Out[24]:
```
array(['Hyundai', 'Mahindra', 'Ford', 'Maruti', 'Skoda', 'Audi', 'Toy
ota',
       'Renault', 'Honda', 'Datsun', 'Mitsubishi', 'Tata', 'Volkswage
n',
       'Chevrolet', 'Mini', 'BMW', 'Nissan', 'Hindustan', 'Fiat', 'Fo
rce',
       'Mercedes', 'Land', 'Jaguar', 'Jeep', 'Volvo'], dtype=object)
```

In [25]:
```
import seaborn as sns
```

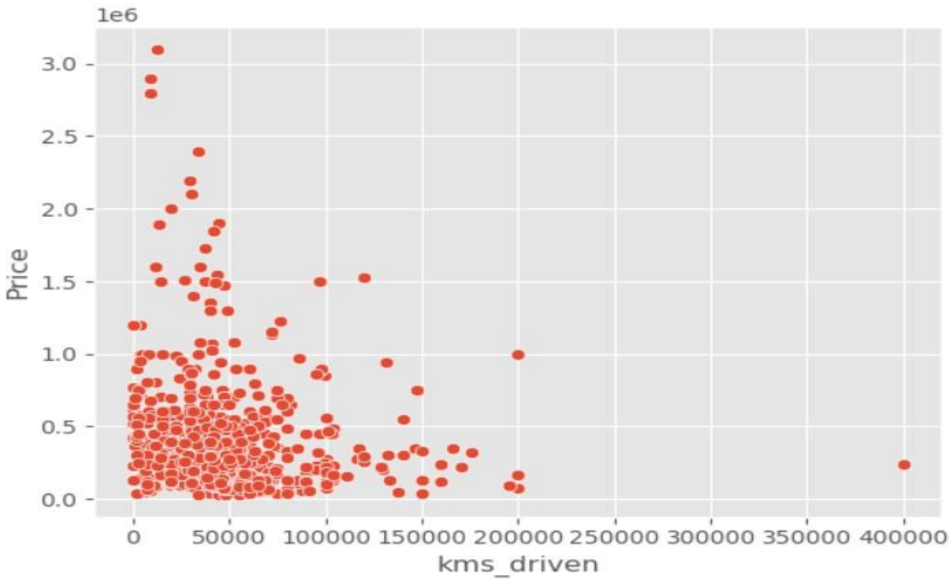
In [26]:
```
plt.subplots(figsize=(15,7))
ax=sns.boxplot(x='company',y='Price',data=car)
ax.set_xticklabels(ax.get_xticklabels(),rotation=40,ha='right')
plt.show()
```



Checking relationship of kms_driven with Price

In [27]:
```
sns.scatterplot(x='kms_driven',y='Price',data=car)
```

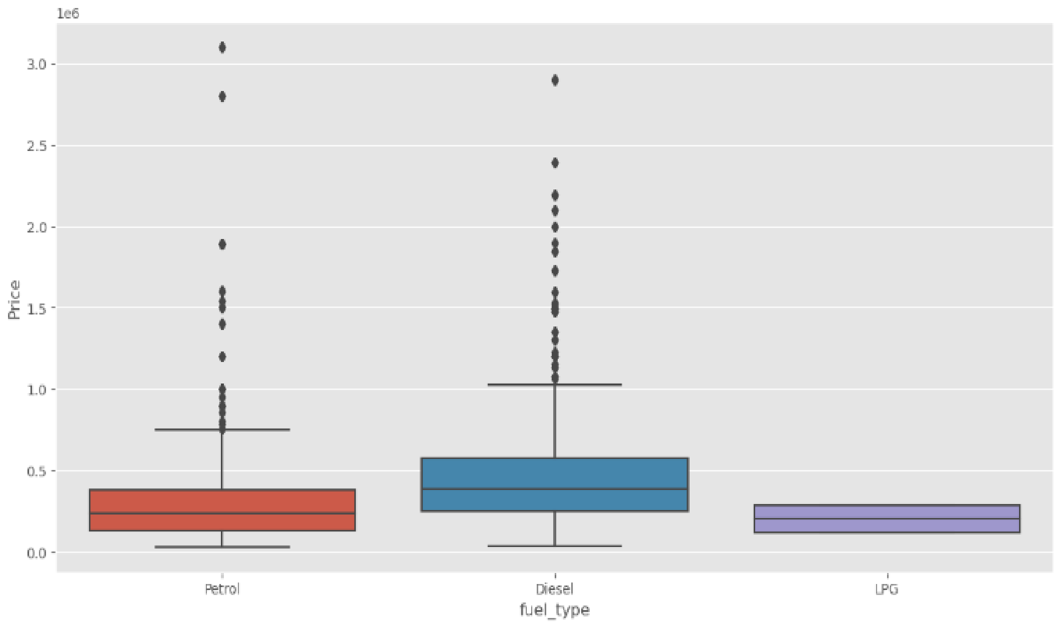
Out[27]: <Axes: xlabel='kms_driven', ylabel='Price'>



Checking relationship of Fuel Type with Price

n [28]:
```
plt.subplots(figsize=(14,7))
sns.boxplot(x='fuel_type',y='Price',data=car)
```

ut[28]: <Axes: xlabel='fuel_type', ylabel='Price'>



Extracting Training Data

In [30]:
```
X=car[['name','company','year','kms_driven','fuel_type']]
y=car['Price']
```

In [31]:
```
X
```

Out[31]:

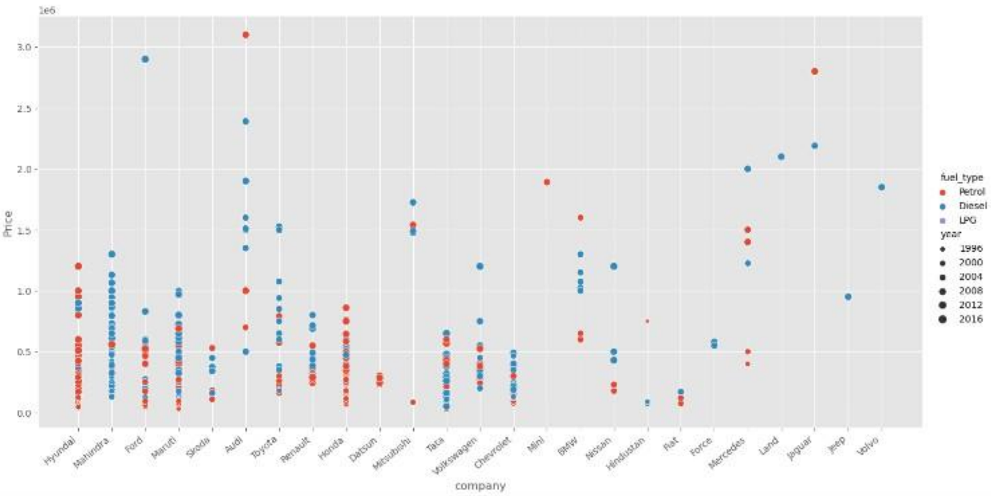
	name	company	year	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	40	Diesel
2	Hyundai Grand i10	Hyundai	2014	28000	Petrol
3	Ford EcoSport Titanium	Ford	2014	36000	Diesel
4	Ford Figo	Ford	2012	41000	Diesel
...	...	...	...	...	...
4891	Maruti Suzuki Ritz	Maruti	2011	50000	Petrol
4892	Tata Indica V2	Tata	2009	30000	Diesel
4893	Toyota Corolla Altis	Toyota	2009	132000	Petrol
4894	Tata Zest XM	Tata	2018	27000	Diesel
4895	Mahindra Quanto C8	Mahindra	2013	40000	Diesel

4890 rows × 5 columns

Relationship of Price with FuelType, Year and Company mixed

In [29]:
```
ax=sns.relplot(x='company',y='Price',data=car,hue='fuel_type',size='year',height
ax.set_xticklabels(rotation=40,ha='right')
```

Out[29]:
```
<seaborn.axisgrid.FacetGrid at 0x13dfb45ead0>
```



Applying Train Test Split

In [33]:
```python
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)
```

In [34]:
```python
from sklearn.linear_model import LinearRegression
```

In [35]:
```python
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score
```

Creating an OneHotEncoder object to contain all the possible categories

In [36]:
```python
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
```

Out[36]:
```
▾ OneHotEncoder
OneHotEncoder()
```

Creating a column transformer to transform categorical columns

In [37]:
```python
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),
                                    remainder='passthrough')
```

Linear Regression Model

In [38]:
```python
lr=LinearRegression()
```

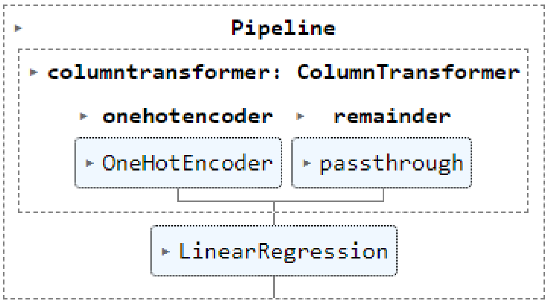
Making a pipeline

In [39]:
```python
pipe=make_pipeline(column_trans,lr)
```

Fitting the model

In [40]:
```python
pipe.fit(X_train,y_train)
```

Out[40]:



In [41]:
```
y_pred=pipe.predict(X_test)
```

In [42]:
```
r2_score(y_test,y_pred)
```

Out[42]: 0.7730912051445293

Finding the model with a random state of TrainTestSplit where the model was found to give almost 0.92 as r2_score

Checking R2 Score

In [ ]:
```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_stat
    lr=LinearRegression()
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

In [ ]:
```
np.argmax(scores)
```

In [ ]:
```
scores[np.argmax(scores)]
```

In [ ]:
```
pipe.predict(pd.DataFrame(columns=X_test.columns,data=np.array(['Maruti Suzuki Sw
```

The best model is found at a certain random state

In [49]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np
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)
```

Out[49]: 0.8462901176591714

In [ ]:

In [54]:
```
import pickle
```

In [55]:
```
pickle.dump(pipe,open('Model.pkl','wb'))
```

In [56]:
```
df=pickle.load(open('Model.pkl','rb'))
```

pipe.predict(pd.DataFrame(columns=['name','company','year','kms_driven','fuel_type'],data=np.array(['Maruti Suzuki Swift','Maruti',2019,100,'Petrol']).reshape(1,5)))

Out[57]: array([440993.81577112])

Index.html

```
<!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+NcPb1dKGj7Sk"
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 ><br><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()"><b>Predict
Price</b></button>
               </div>
            </form>
            <br>
            <div class="row">
               <div class="col-12" style="text-align: center">
                  <h3><span id="prediction"></span></h3>
               </div>
            </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>

<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
        crossorigin="anonymous"></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>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
        integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
        crossorigin="anonymous"></script>
</body>
</html>
```

Style.css

```
body {
    background-
image:url(https://axleaddict.com/.image/t_share/MTk4NTE5NTU3MDY3OTA4NTQ3/man-
gives-tour-of-the-most-exclusive-car-dealership-in-the-country.jpg); /* Set a
light background color for the body */
    background-size: cover; /* Cover the entire container with the background
image */
    background-position: center center; /* Center the background image */
    background-repeat: no-repeat; /* Do not repeat the background image */
    border-radius: 10px; /* Add rounded corners to the container */
    padding: 30px; /* Add padding inside the container */
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.1); /* Add a subtle shadow effect */
    text-align: center; /* Center align text inside the container */
    color: #ffffff; /* Set white text color */
    padding-left: 23%;

}


.container {
    margin-top:auto; /* Adjust the margin from the top */
    background-color:transparent; /* Set the background image for the
container */
    background-size: cover; /* Cover the entire container with the background
image */
    background-position: center center; /* Center the background image */
    background-repeat: no-repeat; /* Do not repeat the background image */
    border-radius: 10px; /* Add rounded corners to the container */
    padding: 30px; /* Add padding inside the container */
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.1); /* Add a subtle shadow effect */
    text-align: right; /* Center align text inside the container */
    color: hwb(0 0% 100%); /* Set white text color */
    font-family: 'Times New Roman', Times, serif;
    font-weight:bold;
    font-size: large;
    height: auto; /* Set the height of the container */
    width: 50% /* Set the width of the container */
}
.card-header{
    font-family: 'Times New Roman', Times, serif;
    font-weight:bold;
    font-size: 50%;
    background-color: transparent;
}
.row{
    margin-top: 0px;
    width:900px;
```

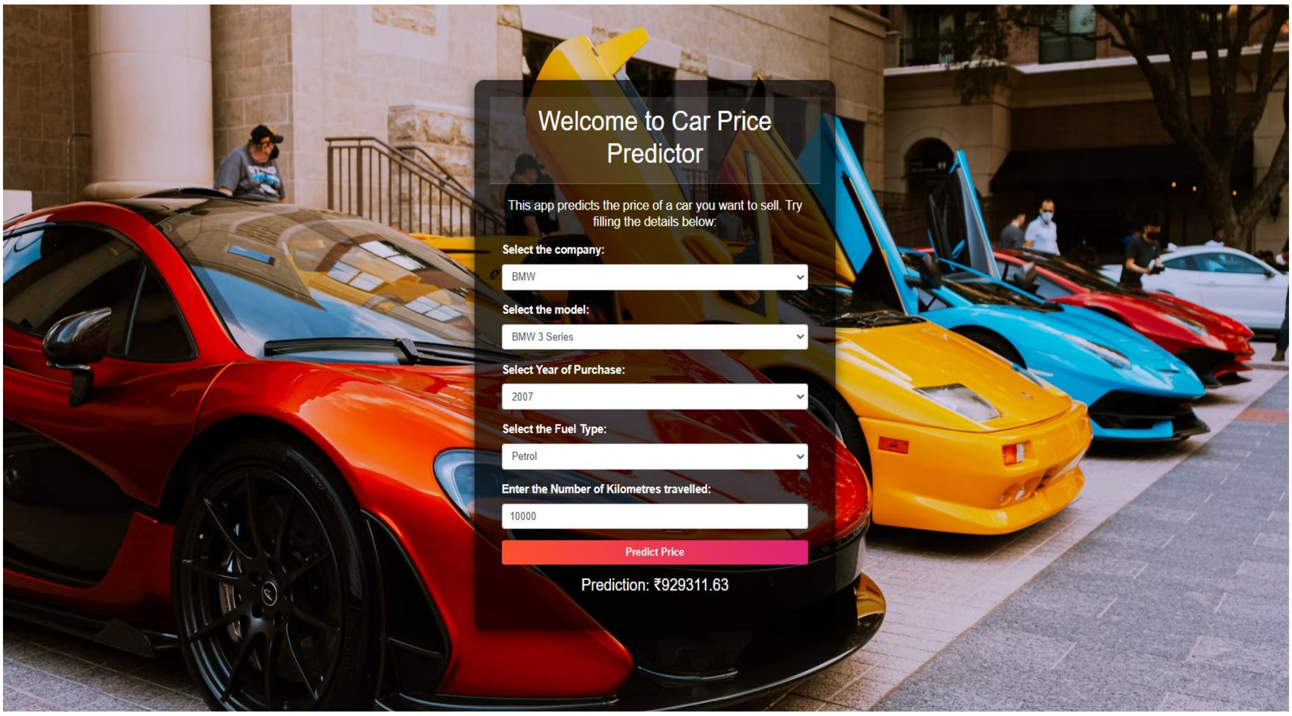
```
    padding-right: 100px;

}
.col-md-10 form-group{
    position:center;
    text-align: right;
    padding-left: 500px;
    padding-top: 50px;

}
.card-body {

    background-image:url(https://hdqwalls.com/wallpapers/light-abstract-
simple-background-iv.jpg);/* Set a blue background color for the header and
body */
    background-size: cover; /* Cover the entire container with the background
image */
    background-position: center center; /* Center the background image */
    background-repeat: no-repeat;
    color:#fffefe; /* Set white text color */
    font-size: 30px;
    font-family: 'Times New Roman', Times, serif;
    font-weight: bold;
    border-radius: 10px 10px 0 0; /* Add rounded corners to the top of the
card */
}

.card-body {
    padding-top: 50px;
    padding: 20px; /* Add padding inside the card body */
}

.card-body input[type="text"], .card-body select {

    width: 100%; /* Make input fields and select elements full-width */
    margin-bottom: 10px; /* Add spacing between input fields and select
elements */
    height: 100%;
    font-size: 100%;
}

.btn-primary {
    background-color: #28a74600; /* Set a green background color for primary
buttons */
    border-color: #28a74600; /* Set border color same as background color */
}


.btn-primary:hover {
```

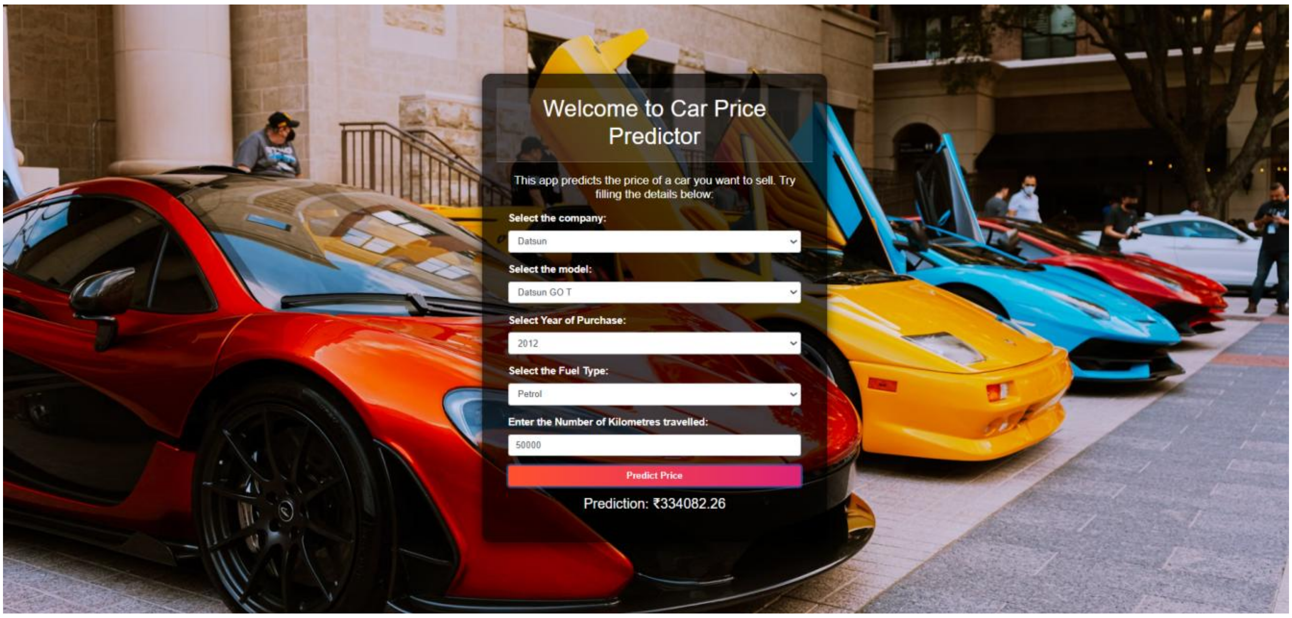
```css
    background-color: #fffffffd; /* Darken the background color on hover */
    border-color: hwb(0 100% 0% / 0.947); /* Darken the border color on hover
*/
}

#prediction {
    font-size: 100%; /* Increase font size for the prediction text */
    font-weight: bold; /* Make the prediction text bold */
}
form {
    font-size: 30px; /* Adjust the font size as needed */
    padding-left: 10%;

}

label {
    display: block;
    margin-bottom: 2px;
}

input, select, textarea {
    font-size: 50%;
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
    box-sizing: 50px;
}
.btn.btn-primary.form-control b {
    font-size: 25px; /* Adjust the font size as needed */
    margin-top: 5%;
    /* Additional styling for the button text if needed */

}
```

Application.py

```
from flask import Flask,render_template,request,redirect
from flask_cors import CORS,cross_origin
import pickle
import pandas as pd
import numpy as np

app=Flask(__name__)
cors=CORS(app)
model=pickle.load(open('Model.pkl','rb'))
car=pd.read_csv('Cleaned_Car_data.csv')

@app.route('/',methods=['GET','POST'])
def index():
    companies=sorted(car['company'].unique())
    car_models=sorted(car['name'].unique())
    year=sorted(car['year'].unique(),reverse=True)
    fuel_type=car['fuel_type'].unique()

    companies.insert(0,'Select Company')
    return render_template('index1.html',companies=companies,
car_models=car_models, years=year,fuel_types=fuel_type)


@app.route('/predict',methods=['POST'])
@cross_origin()
def predict():

    company=request.form.get('company')

    car_model=request.form.get('car_models')
    year=request.form.get('year')
    fuel_type=request.form.get('fuel_type')
    driven=request.form.get('kilo_driven')

    prediction=model.predict(pd.DataFrame(columns=['name', 'company', 'year',
'kms_driven', 'fuel_type'],
                              data=np.array([car_model,company,year,driven,fuel_typ
e]).reshape(1, 5)))
    print(prediction)

    return str(np.round(prediction[0],2))



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

Output:





Conclusion:

Used car price prediction is a valuable application of machine learning and data analytics that aims to forecast the market value of pre-owned vehicles based on various factors. The process involves leveraging historical data, statistical models, and machine learning algorithms to estimate the price of a used car.