USED CAR PRICE PREDICTION USING MACHINE LEARNING REGRESSION ALGORITHEMS

BY: *Potti mohan vamsi krishna kanth*
B.Tech(cse)-3rd year student
at Diginique Techlabs,
Amity University Gurugram.
Mentor*: *Bandenawaz Bagwan

ABSTRACT

Determining whether the listed prices of a used car is a challenging task, due to the many factors that drive a used car price on the market. The focus of this project is developing machine learning models that can 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. Our results show that Decision tree regression model, Random Forest model and K-nearest neighbors algorithm with linear regression yield the best results and also created with web application to verify the price of a car using features.

INTRODUCTION ABOUT MACHINE LEARNING

Machine learning is a branch of artificial intelligence(AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

These are three types of machine learning:
1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

1.**supervised learning:**

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labeled data. Even though the data needs to be labeled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances.

In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labeled parameters required for the problem.

The algorithm then finds relationships between the parameters given, essentially establishing a causes and effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.
This solution is then deployed for use with the final dataset, which it learns from in the same way as the training dataset. This means that supervised machine learning algorithms will continue to improve even after being deployed, discovering new patterns and relationships as it trains itself on new data.

2. **unsupervised learning**:

Unsupervised machine learning holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.

In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.

The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

3.**reinforcement learning:**

It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. favorable outputs are encouraged or ‘reinforced’, and non-favorable outputs are discouraged or ‘punished’.

Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not.

In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result.

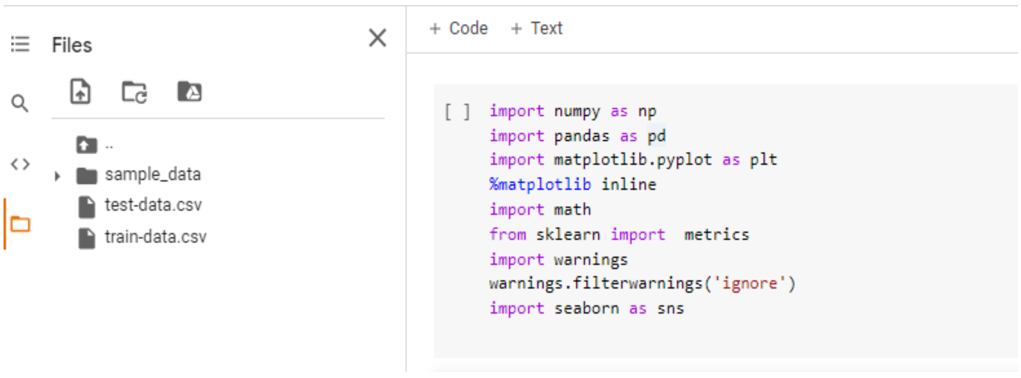
In typical reinforcement learning use-cases, such as finding the shortest route between two points on a map, the solution is not an absolute value. Instead, it takes on a score of effectiveness, expressed in a percentage value. The higher this percentage value is, the more reward is given to the algorithm. Thus, the program is trained to give the best possible solution for the best possible reward.

PROBLEM STATEMENT

Used car price prediction. predict the car price of unknown car using multiple algorithms.

METHODOLOGY

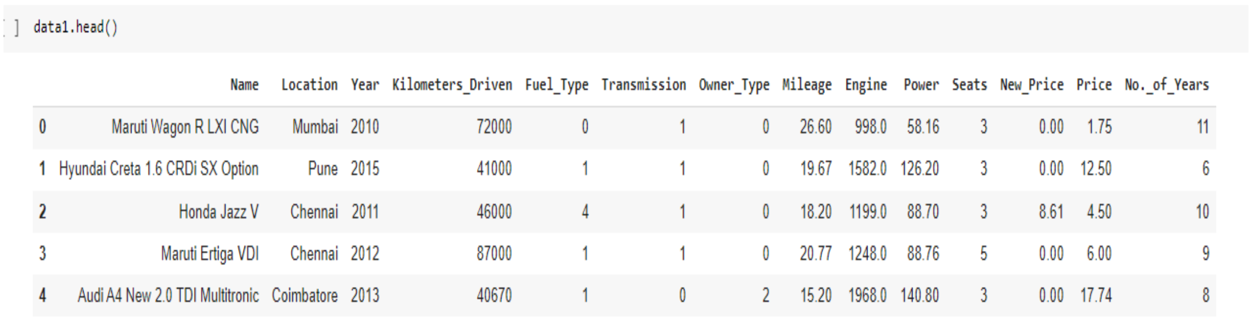
STEP 1. Importing libraries and datasets



```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     %matplotlib inline
     import math
     from sklearn import  metrics
     import warnings
     warnings.filterwarnings('ignore')
     import seaborn as sns
```

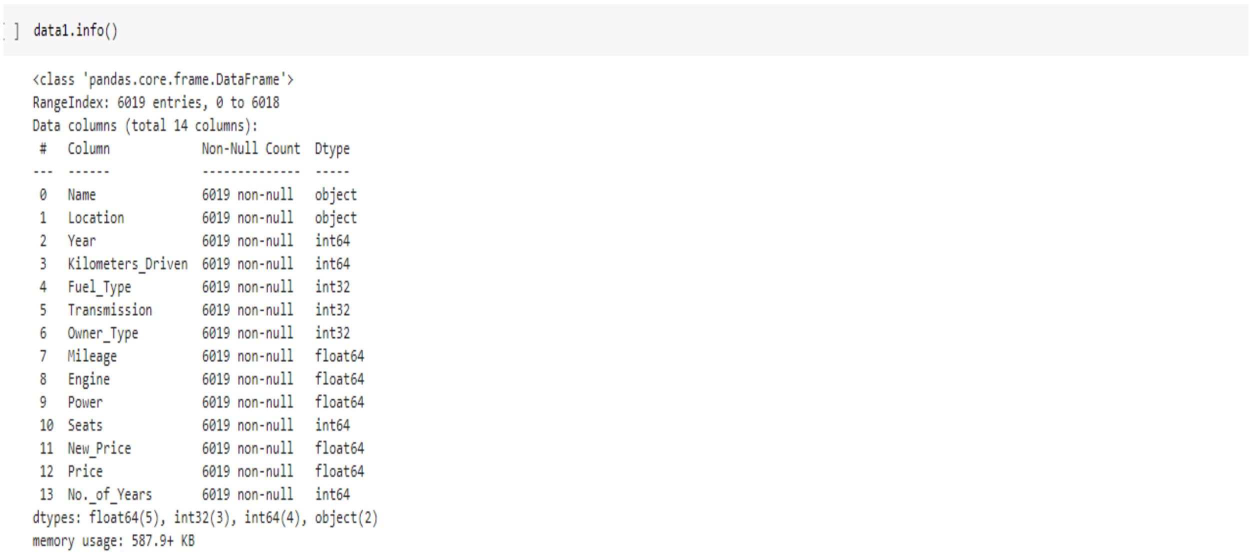
STEP 2. After Removing impurities of our train and test datasets.

Train Dataset:



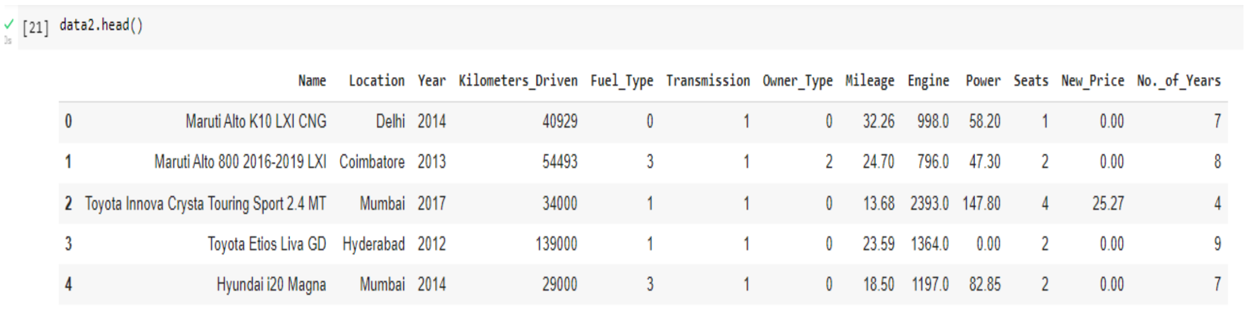
```
] data1.head()
```

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price	No._of_Years
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	0	1	0	26.60	998.0	58.16	3	0.00	1.75	11
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	1	1	0	19.67	1582.0	126.20	3	0.00	12.50	6
2	Honda Jazz V	Chennai	2011	46000	4	1	0	18.20	1199.0	88.70	3	8.61	4.50	10
3	Maruti Ertiga VDI	Chennai	2012	87000	1	1	0	20.77	1248.0	88.76	5	0.00	6.00	9
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	1	0	2	15.20	1968.0	140.80	3	0.00	17.74	8



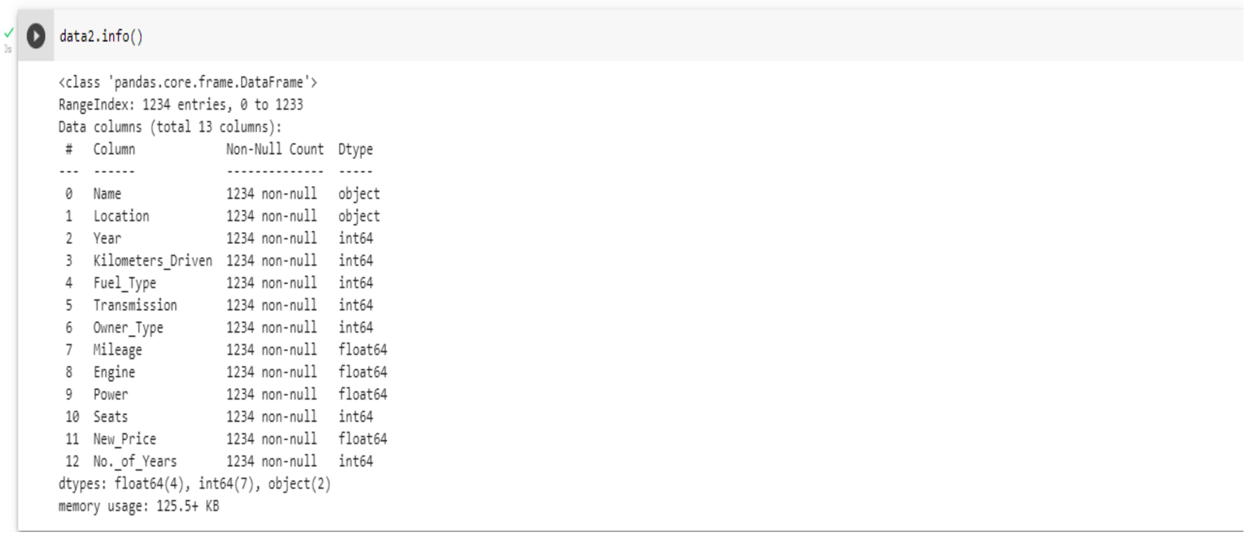
```
] data1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Name               6019 non-null   object
 1   Location           6019 non-null   object
 2   Year               6019 non-null   int64
 3   Kilometers_Driven  6019 non-null   int64
 4   Fuel_Type          6019 non-null   int32
 5   Transmission       6019 non-null   int32
 6   Owner_Type         6019 non-null   int32
 7   Mileage            6019 non-null   float64
 8   Engine             6019 non-null   float64
 9   Power              6019 non-null   float64
 10  Seats              6019 non-null   int64
 11  New_Price          6019 non-null   float64
 12  Price              6019 non-null   float64
 13  No._of_Years       6019 non-null   int64
dtypes: float64(5), int32(3), int64(4), object(2)
memory usage: 587.9+ KB
```

Test Dataset:



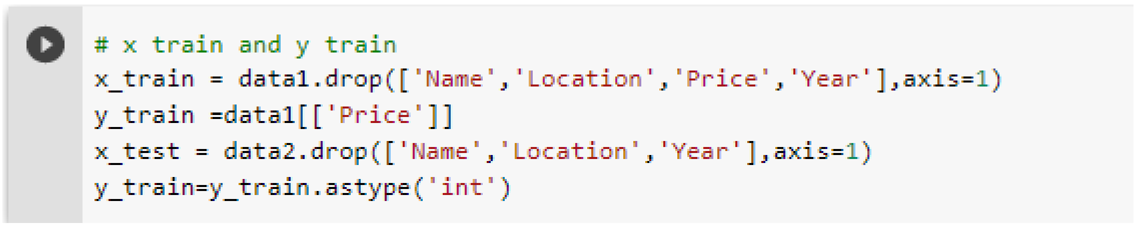
```
[21] data2.head()
```

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	No._of_Years
0	Maruti Alto K10 LXI CNG	Delhi	2014	40929	0	1	0	32.26	998.0	58.20	1	0.00	7
1	Maruti Alto 800 2016-2019 LXI	Coimbatore	2013	54493	3	1	2	24.70	796.0	47.30	2	0.00	8
2	Toyota Innova Crysta Touring Sport 2.4 MT	Mumbai	2017	34000	1	1	0	13.68	2393.0	147.80	4	25.27	4
3	Toyota Etios Liva GD	Hyderabad	2012	139000	1	1	0	23.59	1364.0	0.00	2	0.00	9
4	Hyundai i20 Magna	Mumbai	2014	29000	3	1	0	18.50	1197.0	82.85	2	0.00	7



```
data2.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1234 entries, 0 to 1233
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Name               1234 non-null   object
 1   Location           1234 non-null   object
 2   Year               1234 non-null   int64
 3   Kilometers_Driven  1234 non-null   int64
 4   Fuel_Type          1234 non-null   int64
 5   Transmission       1234 non-null   int64
 6   Owner_Type         1234 non-null   int64
 7   Mileage            1234 non-null   float64
 8   Engine             1234 non-null   float64
 9   Power              1234 non-null   float64
 10  Seats              1234 non-null   int64
 11  New_Price          1234 non-null   float64
 12  No._of_Years       1234 non-null   int64
dtypes: float64(4), int64(7), object(2)
memory usage: 125.5+ KB
```

STEP 3: Training and test Using csv file



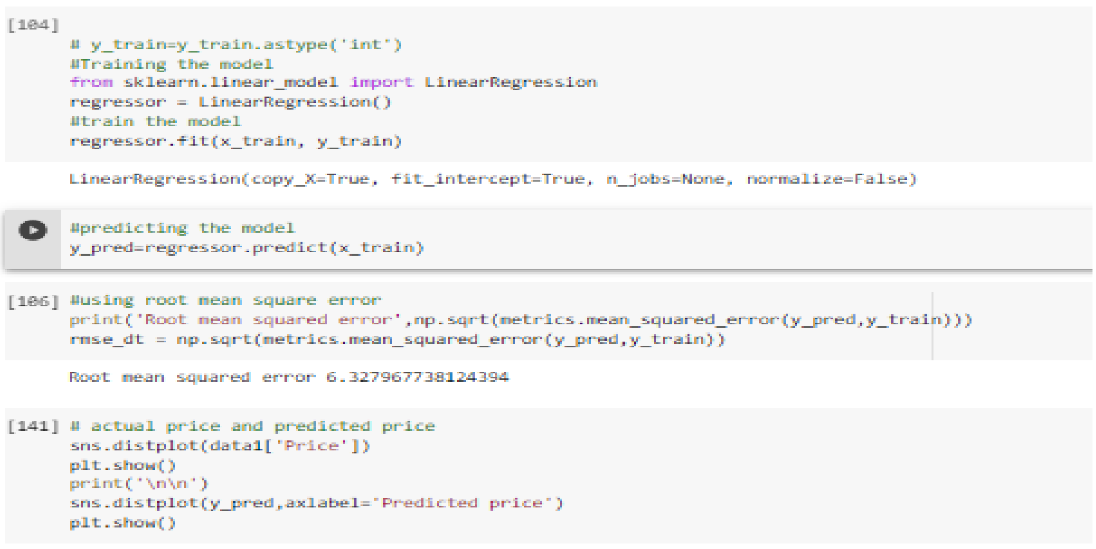
```
# x train and y train
x_train = data1.drop(['Name','Location','Price','Year'],axis=1)
y_train =data1[['Price']]
x_test = data2.drop(['Name','Location','Year'],axis=1)
y_train=y_train.astype('int')
```

STEP 4: Used Algorithms for Predicting Used car price

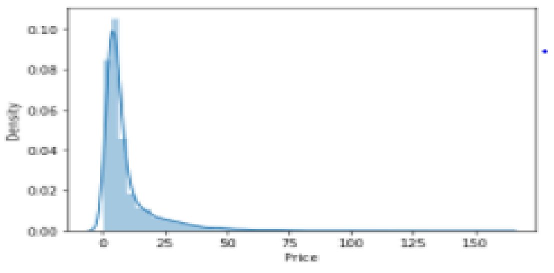
1.**Linear regression algorithm:**

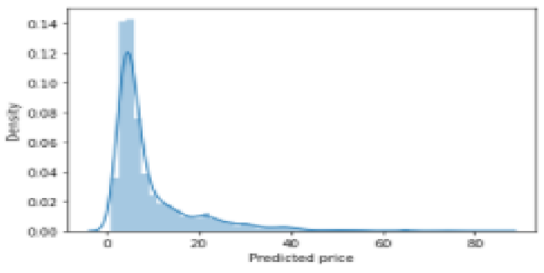
Linear Regression is a machine learning algorithm based on **supervised learning.** It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

Train the data using Linear regression and predict the price using features of a car. And we used here Root mean square error. And visualize the predicted price and actual price.



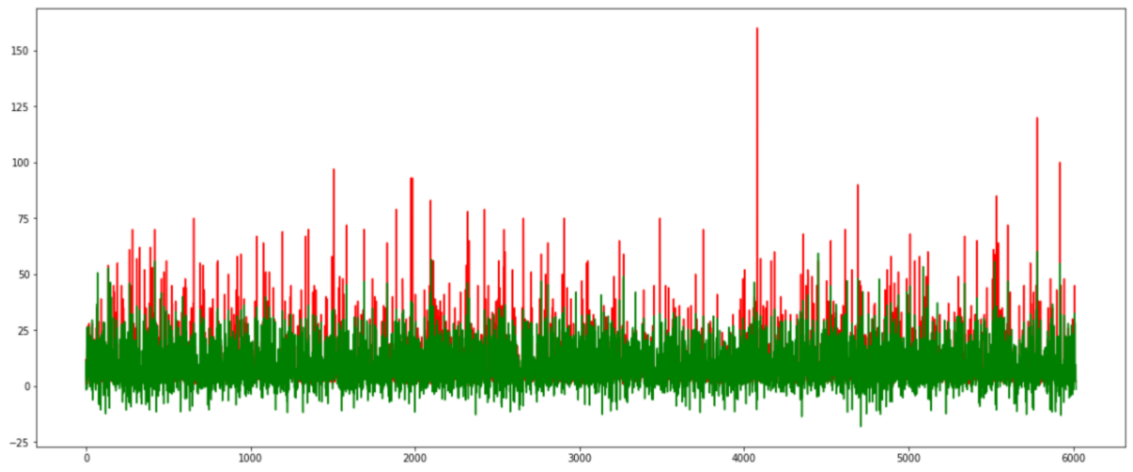
```
[104]
      # y_train=y_train.astype('int')
      #Training the model
      from sklearn.linear_model import LinearRegression
      regressor = LinearRegression()
      #train the model
      regressor.fit(x_train, y_train)

      LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

      #predicting the model
      y_pred=regressor.predict(x_train)

[106] #using root mean square error
      print('Root mean squared error',np.sqrt(metrics.mean_squared_error(y_pred,y_train)))
      rmse_dt = np.sqrt(metrics.mean_squared_error(y_pred,y_train))

      Root mean squared error 6.327967738124394

[141] # actual price and predicted price
      sns.distplot(data1['Price'])
      plt.show()
      print('\n\n')
      sns.distplot(y_pred,axlabel='Predicted price')
      plt.show()
```



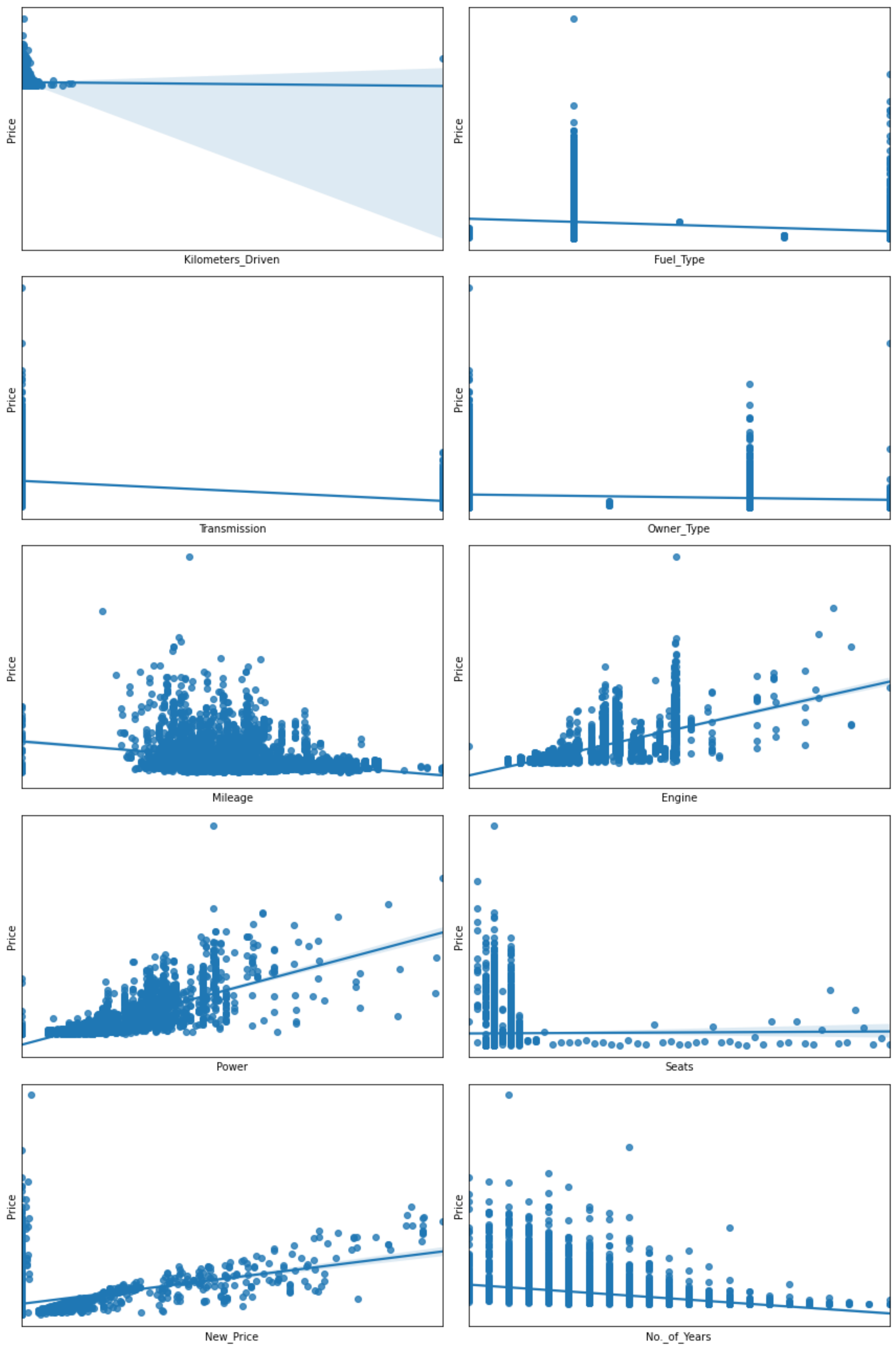


Visualizing difference between actual price and predicted price

```
plt.figure(figsize = (20,8))
plt.plot(y_train['Price'], 'r')
plt.plot(y_pred,'g')
plt.show()
```

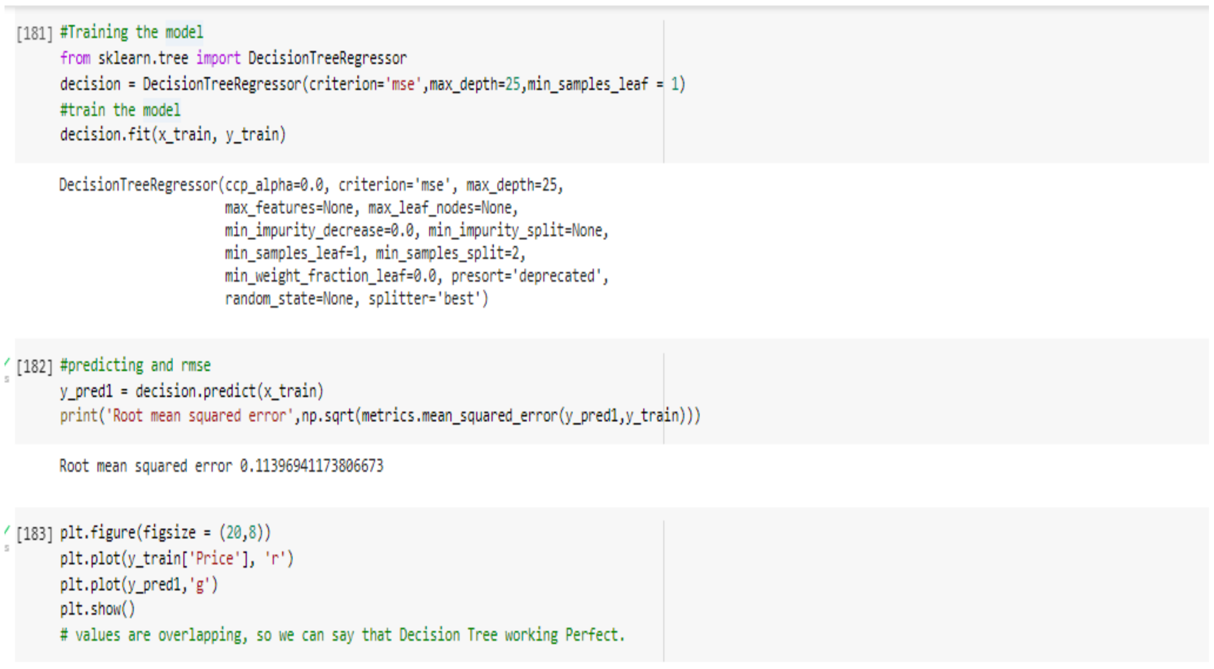


```
180] fig = plt.figure(figsize=(12,18))
     for i in range(len(x_train.columns)):
         fig.add_subplot(5,2,i+1)
         sns.regplot(x=x_train.columns[i], y='Price', data=data1)
         plt.xlabel(x_train.columns[i])
         plt.xticks(())
         plt.yticks(())
     plt.tight_layout()
     plt.show()
```


Price
Kilometers_Driven
Price
Fuel_Type
Price
Transmission
Price
Owner_Type
Price
Mileage
Price
Engine
Price
Power
Price
Seats
Price
New_Price
Price
No._of_Years

2. **Decision Tree Regression:**

Decision Tree is one of the most commonly used, practical approaches for supervised learning. It can be used to solve both Regression and Classification tasks with the latter being put more into practical application.
Train the data using decision tree regression and predict the price using features of a car. And we used here Root mean square error. And visualize the predicted price and actual price.



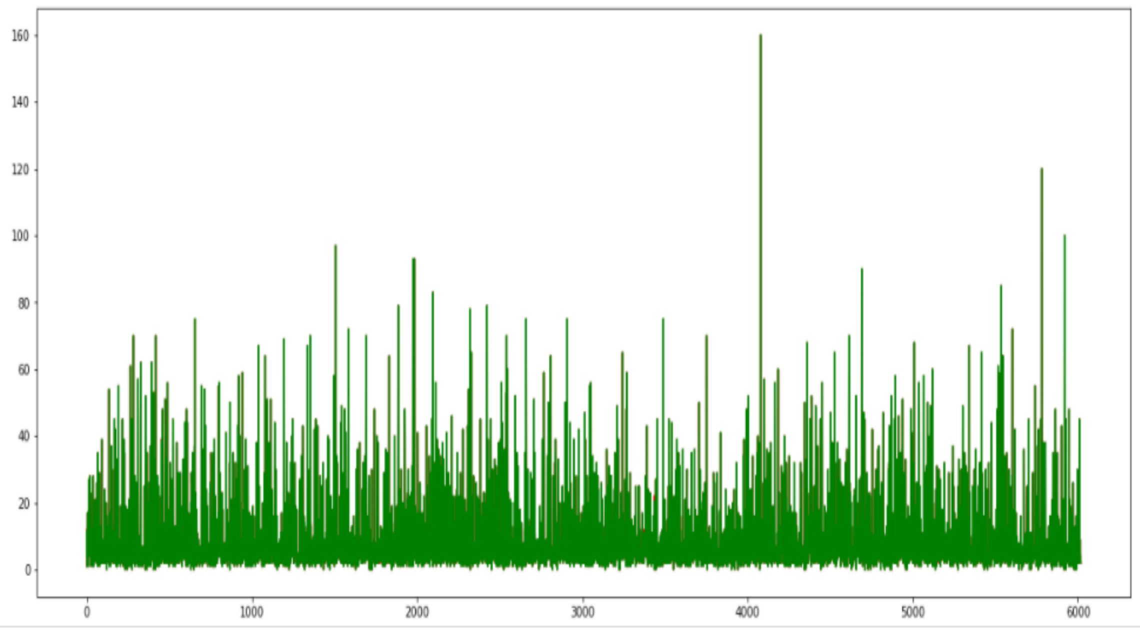
```
[181] #Training the model
      from sklearn.tree import DecisionTreeRegressor
      decision = DecisionTreeRegressor(criterion='mse',max_depth=25,min_samples_leaf = 1)
      #train the model
      decision.fit(x_train, y_train)
```

```
      DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=25,
                            max_features=None, max_leaf_nodes=None,
                            min_impurity_decrease=0.0, min_impurity_split=None,
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, presort='deprecated',
                            random_state=None, splitter='best')
```

```
[182] #predicting and rmse
      y_pred1 = decision.predict(x_train)
      print('Root mean squared error',np.sqrt(metrics.mean_squared_error(y_pred1,y_train)))
```

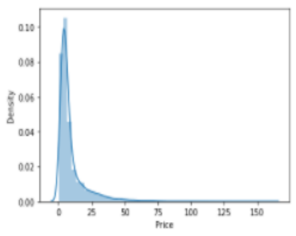
```
      Root mean squared error 0.11396941173806673
```

```
[183] plt.figure(figsize = (20,8))
      plt.plot(y_train['Price'], 'r')
      plt.plot(y_pred1,'g')
      plt.show()
      # values are overlapping, so we can say that Decision Tree working Perfect.
```

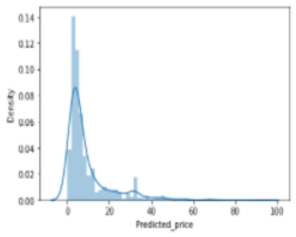


Here, we added predicted price cell

```
[186] sns.distplot(data1['Price'])
      plt.show()
```



```
[187] sns.distplot(data2['Predicted_price'])
      plt.show()
```



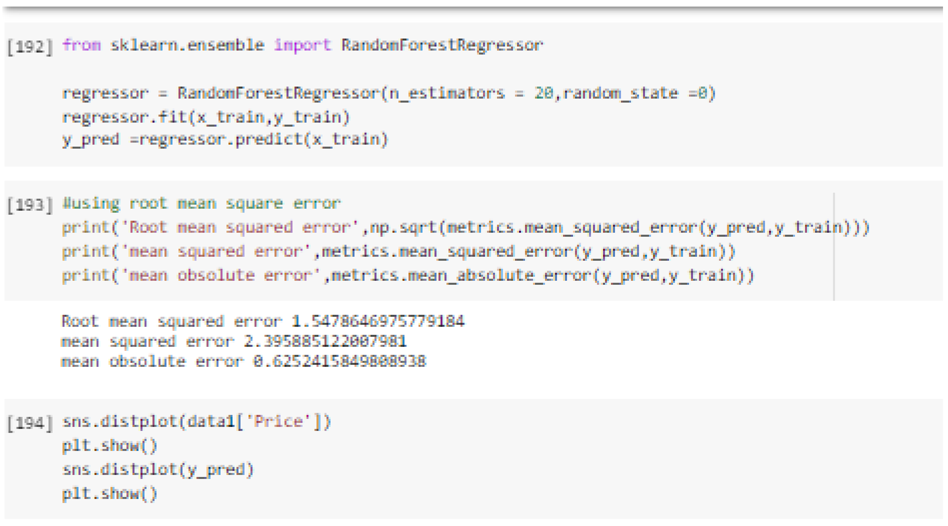
```
[188] data2
```

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	No._of_Years	Predicted_price
0	Maruti Alto K10 LXI CNG	Delhi	2014	40929	0	1	0	32.26	998.0	58.20	1	0.00	7	3.0
1	Maruti Alto 800 2016-2019 LXI	Coimbatore	2013	54493	3	1	2	24.70	796.0	47.30	2	0.00	8	2.0
2	Toyota Innova Crysta Touring Sport 2.4 MT	Mumbai	2017	34000	1	1	0	13.68	2393.0	147.80	4	25.27	4	17.0
3	Toyota Etios Liva GD	Hyderabad	2012	139000	1	1	0	23.59	1364.0	0.00	2	0.00	9	2.0
4	Hyundai i20 Magna	Mumbai	2014	29000	3	1	0	18.50	1197.0	82.85	2	0.00	7	4.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1229	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	1	1	0	20.54	1598.0	103.60	2	0.00	10	3.0
1230	Volkswagen Polo GT TSI	Mumbai	2015	59000	3	0	0	17.21	1197.0	103.60	2	0.00	6	6.0
1231	Nissan Micra Diesel XV	Kolkata	2012	28000	1	1	0	23.08	1461.0	63.10	2	0.00	9	2.0
1232	Volkswagen Polo GT TSI	Pune	2013	52262	3	0	3	17.20	1197.0	103.60	2	0.00	8	4.0
1233	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	1	0	0	10.00	2148.0	170.00	2	0.00	7	27.0

1234 rows × 14 columns

3. **Random forest regression:**

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.
Train the data using random forest regressor and predict the price using features of a car. And we used here Root mean square error, mean square error, mean absolute error.
And visualize the predicted price and actual price.

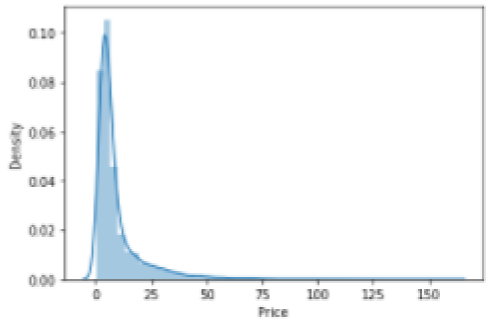


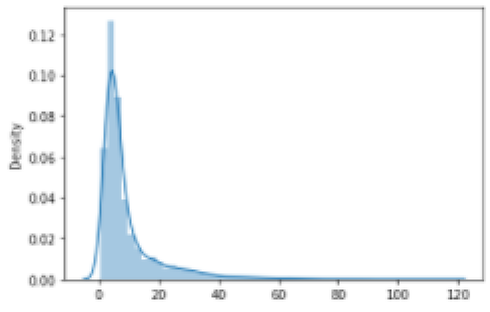
```
[192] from sklearn.ensemble import RandomForestRegressor

      regressor = RandomForestRegressor(n_estimators = 20,random_state =0)
      regressor.fit(x_train,y_train)
      y_pred =regressor.predict(x_train)
```

```
[193] #using root mean square error
      print('Root mean squared error',np.sqrt(metrics.mean_squared_error(y_pred,y_train)))
      print('mean squared error',metrics.mean_squared_error(y_pred,y_train))
      print('mean obsolute error',metrics.mean_absolute_error(y_pred,y_train))
```

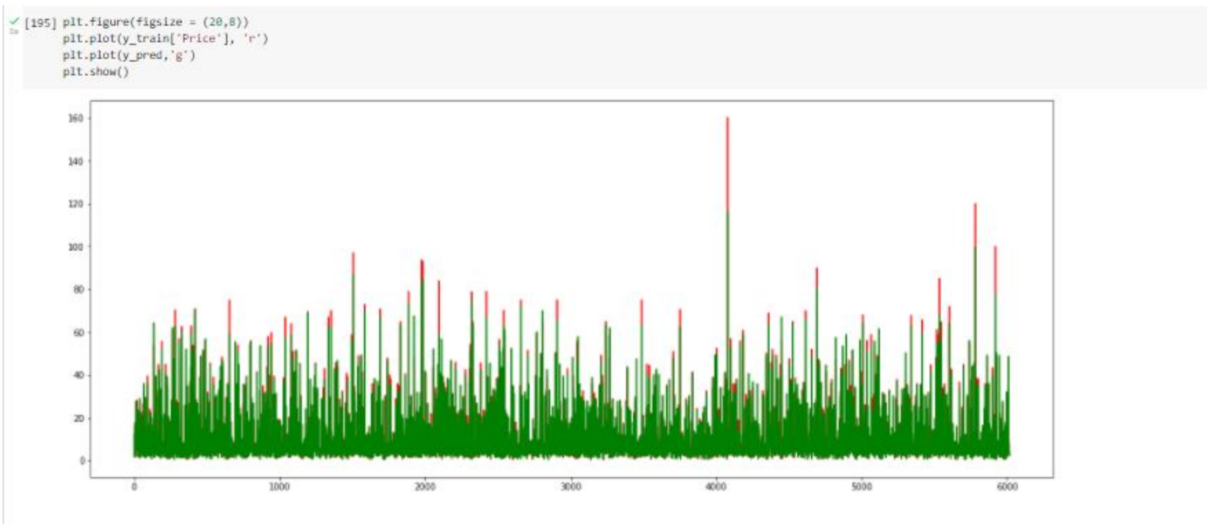
```
      Root mean squared error 1.5478646975779184
      mean squared error 2.395885122007981
      mean obsolute error 0.6252415849808938
```

```
[194] sns.distplot(data1['Price'])
      plt.show()
      sns.distplot(y_pred)
      plt.show()
```





Difference between actual price and predicted price using Random forest



```
[195] plt.figure(figsize = (20,8))
      plt.plot(y_train['Price'], 'r')
      plt.plot(y_pred,'g')
      plt.show()
```

4.**K-nearest neighbors:**

KNN regression is a non-parametric method that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same neighborhood.

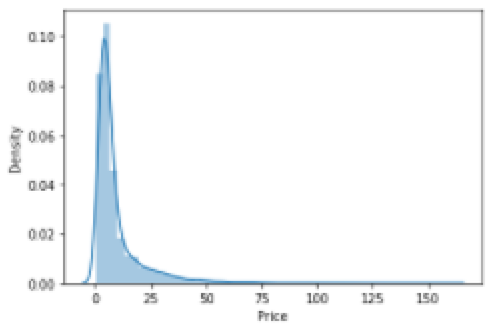
Train the data using K-nearest neighbors and predict the price using features of a car. And we used here Root mean square error. And visualize the predicted price and actual price.

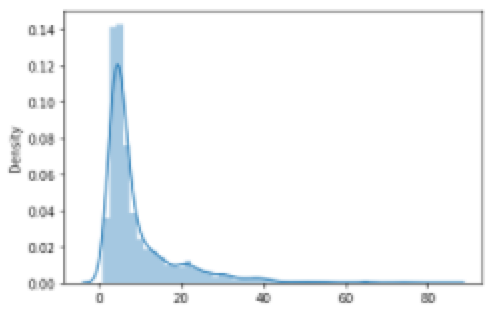
```
[196] from sklearn.neighbors import KNeighborsRegressor
      knn = KNeighborsRegressor()
      knn.fit(x_train,y_train['Price'])

      KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                          metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                          weights='uniform')
```

```
[197] y_pred =knn.predict(x_train)#x_test
```

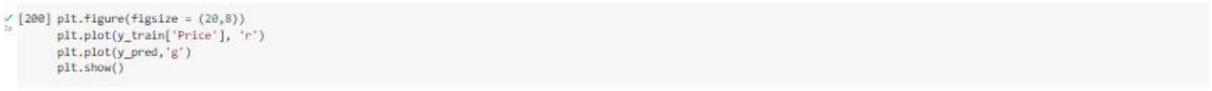
```
[198] print('Root mean squared error',np.sqrt(metrics.mean_squared_error(y_pred,y_train)))

      Root mean squared error 5.84564381625299
```

```
[199] sns.distplot(data1['Price'])
      plt.show()
      sns.distplot(y_pred)
      plt.show()
```

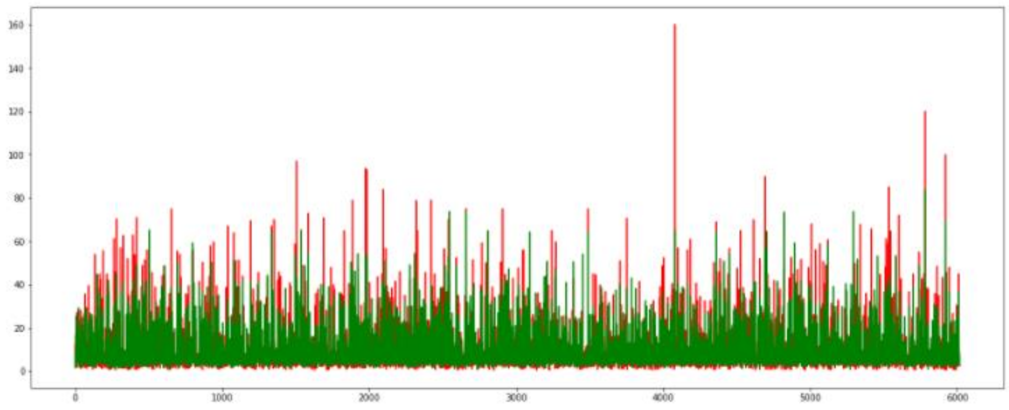




Difference between actual price and predicted price using K-nearest neighbor



```
[200] plt.figure(figsize = (20,8))
      plt.plot(y_train['Price'], 'r')
      plt.plot(y_pred,'g')
      plt.show()
```



WEB APPLICATION USING FLASK:

Flask is an API of Python that allows us to build up web-applications. Flask's framework is more explicit than Django's framework and is also easier to learn because it has less base code to implement a simple web-Application. A Web-Application Framework or Web Framework is the collection of modules and libraries that helps the developer to write applications without writing the low-level codes such as protocols, thread management, etc. Flask is based on WSGI (Web Server Gateway Interface) toolkit and Jinja2 template engine.

Creating Web application using flask based on ml with Used car price prediction and included with decision tree algorithm model.

Step 0**:** Gathering requirements and importing the libraries in the python files**:**

Flask==1.1.1

gunicorn==20.0.4

pandas==0.25.2

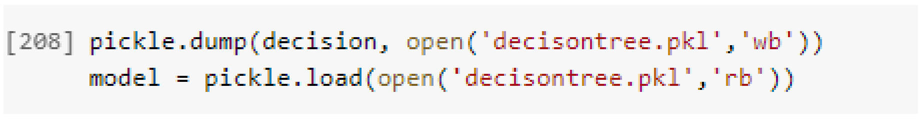
scikit-learn==0.23.2

numpy==1.17.3

pickle4==0.0.1

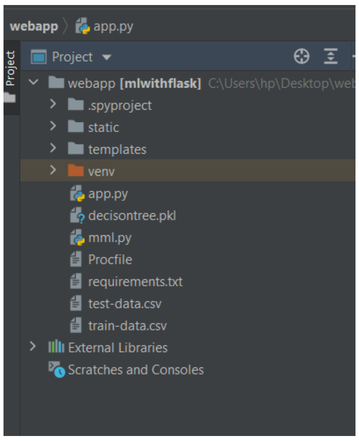
sklearn==0.0

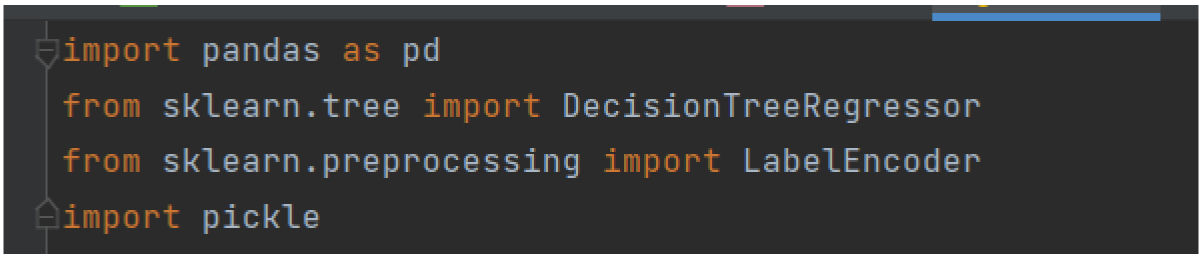
Step 1: import pickle and download the pkl(binary or byte)file



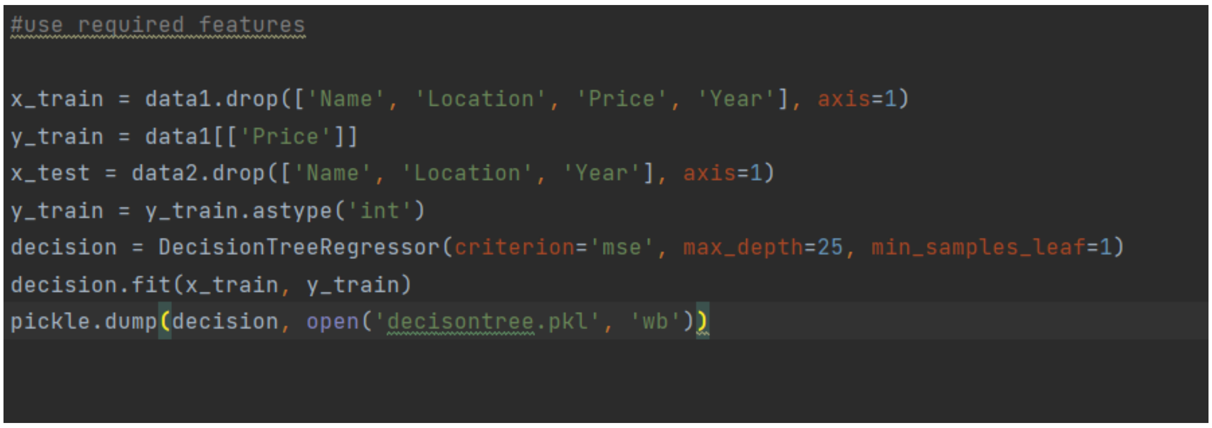
```
[208] pickle.dump(decision, open('decisontree.pkl','wb'))
      model = pickle.load(open('decisontree.pkl','rb'))
```

Step 2: Adding datasets with pkl file in project and import libraries in .py file





```
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import LabelEncoder
import pickle
```

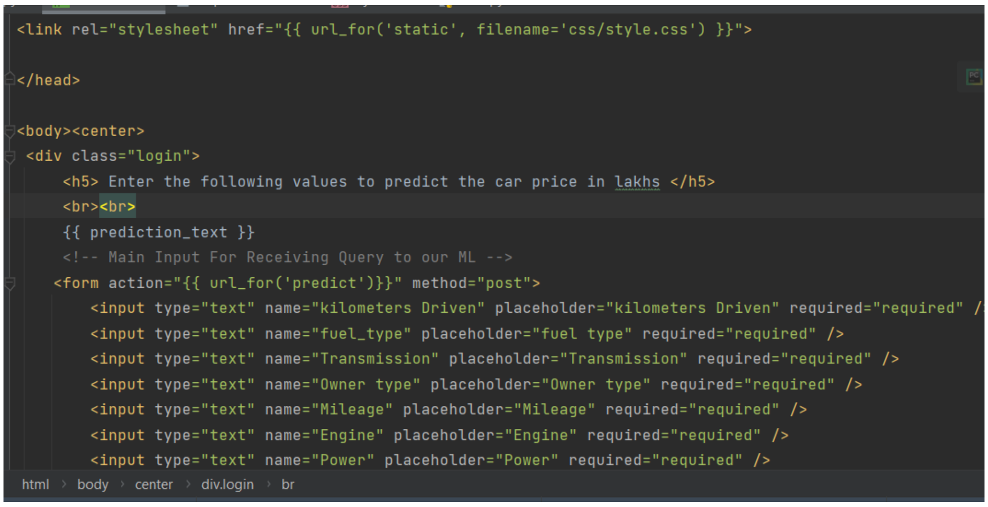


```
#use required features

x_train = data1.drop(['Name', 'Location', 'Price', 'Year'], axis=1)
y_train = data1[['Price']]
x_test = data2.drop(['Name', 'Location', 'Year'], axis=1)
y_train = y_train.astype('int')
decision = DecisionTreeRegressor(criterion='mse', max_depth=25, min_samples_leaf=1)
decision.fit(x_train, y_train)
pickle.dump(decision, open('decisontree.pkl', 'wb'))
```

Step 3: Creating HTML and CSS file.

HTML (Hyper Text Markup Language):

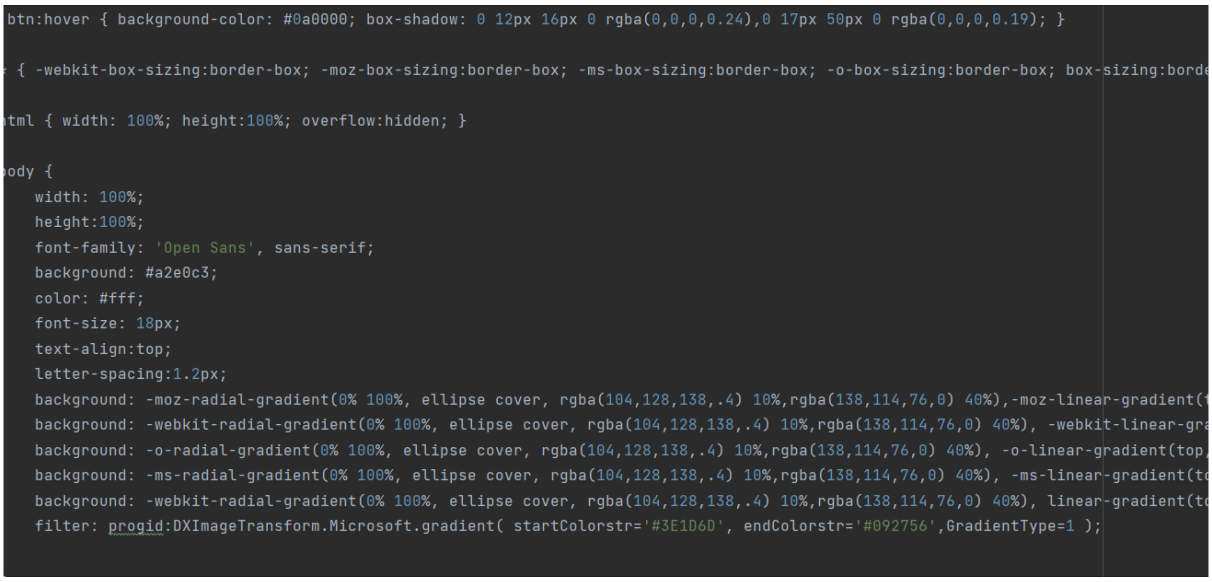
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.



```
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body><center>
 <div class="login">
     <h5> Enter the following values to predict the car price in lakhs </h5>
     <br><br>
     {{ prediction_text }}
     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}" method="post">
        <input type="text" name="kilometers Driven" placeholder="kilometers Driven" required="required" /
        <input type="text" name="fuel_type" placeholder="fuel type" required="required" />
        <input type="text" name="Transmission" placeholder="Transmission" required="required" />
        <input type="text" name="Owner type" placeholder="Owner type" required="required" />
        <input type="text" name="Mileage" placeholder="Mileage" required="required" />
        <input type="text" name="Engine" placeholder="Engine" required="required" />
        <input type="text" name="Power" placeholder="Power" required="required" />
```

CSS (Cascading Style Sheets):

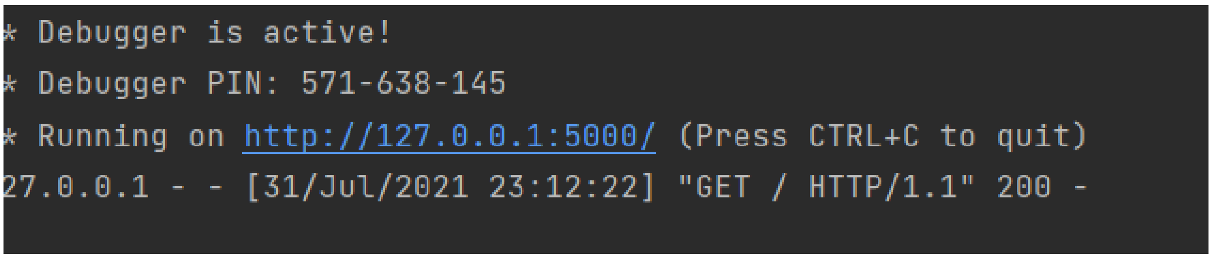
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files



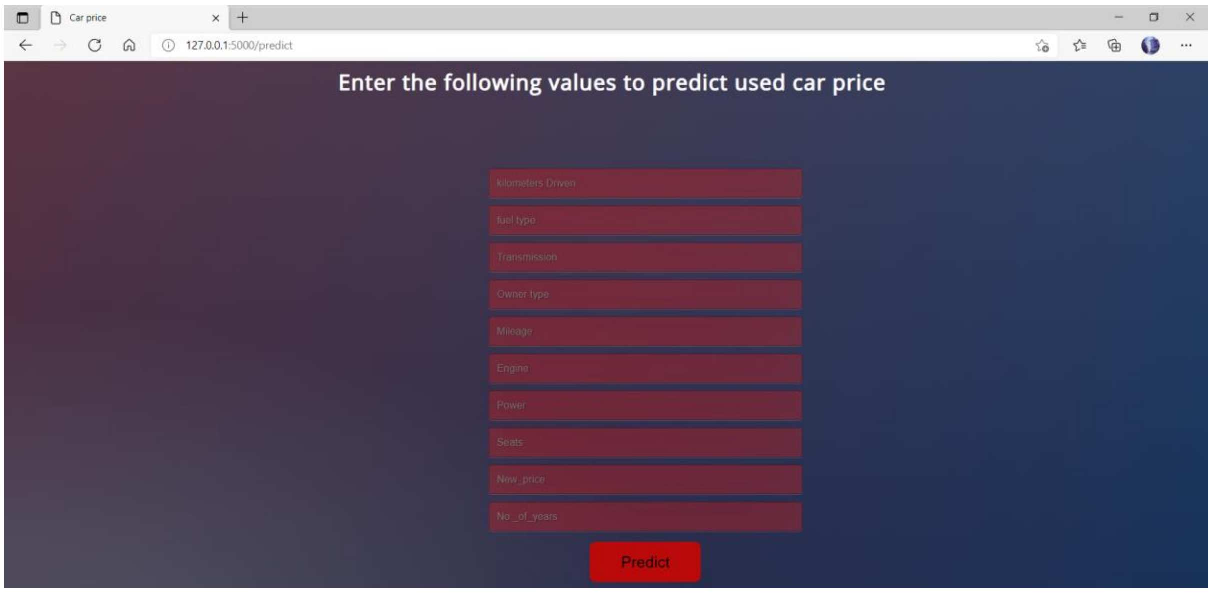
```
btn:hover { background-color: #0a0000; box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24),0 17px 50px 0 rgba(0,0,0,0.19); }

 { -webkit-box-sizing:border-box; -moz-box-sizing:border-box; -ms-box-sizing:border-box; -o-box-sizing:border-box; box-sizing:borde

tml { width: 100%; height:100%; overflow:hidden; }

ody {
   width: 100%;
   height:100%;
   font-family: 'Open Sans', sans-serif;
   background: #a2e0c3;
   color: #fff;
   font-size: 18px;
   text-align:top;
   letter-spacing:1.2px;
   background: -moz-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%),-moz-linear-gradient(t
   background: -webkit-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -webkit-linear-gra
   background: -o-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -o-linear-gradient(top,
   background: -ms-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -ms-linear-gradient(to
   background: -webkit-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), linear-gradient(to
   filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#3E1D6D', endColorstr='#092756',GradientType=1 );
```

Step 4: Web App Interface:

We get a local host link in the terminal.



```
* Debugger is active!
* Debugger PIN: 571-638-145
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
27.0.0.1 - - [31/Jul/2021 23:12:22] "GET / HTTP/1.1" 200 -
```



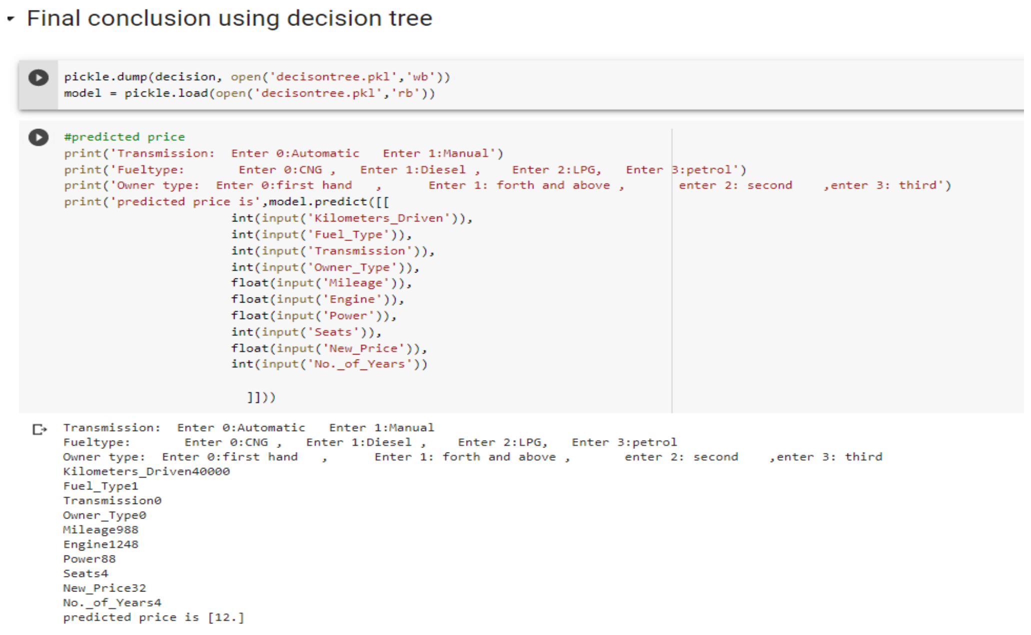
Enter the following values to predict used car price

Conclusion:

The best suitable algorithm for USED CAR PRICE PREDICTION is decision tree algorithm. To get accurate output of our car price.

Because root mean square error is 0.11.

Algorithm Name	**Root mean square error value**
Linear regression	6.327967738124394
Decision tree Regression	0.11396941173806673
Random forest Regression	1.5478646975779184
K-nearest neighbors	5.84564301625299



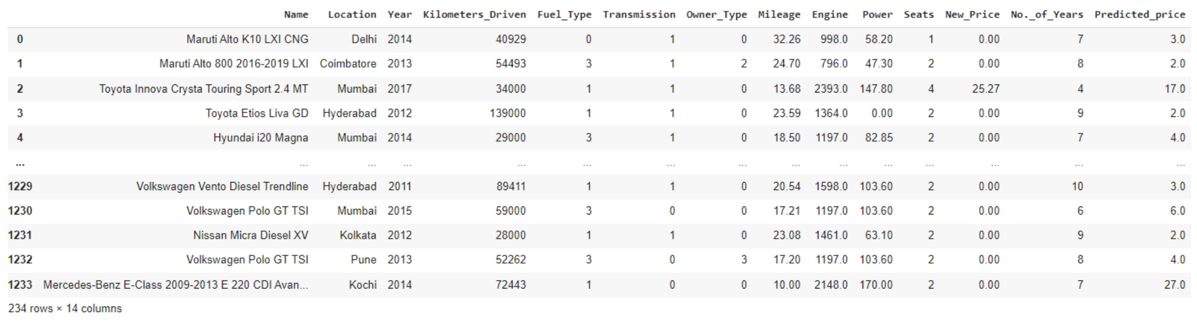
▾ Final conclusion using decision tree

```
pickle.dump(decision, open('decisontree.pkl','wb'))
model = pickle.load(open('decisontree.pkl','rb'))
```

```
#predicted price
print('Transmission:  Enter 0:Automatic   Enter 1:Manual')
print('Fueltype:       Enter 0:CNG ,   Enter 1:Diesel ,    Enter 2:LPG,   Enter 3:petrol')
print('Owner type:  Enter 0:first hand   ,      Enter 1: forth and above ,       enter 2: second    ,enter 3: third')
print('predicted price is',model.predict([[
                      int(input('Kilometers_Driven')),
                      int(input('Fuel_Type')),
                      int(input('Transmission')),
                      int(input('Owner_Type')),
                      float(input('Mileage')),
                      float(input('Engine')),
                      float(input('Power')),
                      int(input('Seats')),
                      float(input('New_Price')),
                      int(input('No._of_Years'))

                        ]]))
```

```
Transmission:  Enter 0:Automatic   Enter 1:Manual
Fueltype:       Enter 0:CNG ,   Enter 1:Diesel ,    Enter 2:LPG,   Enter 3:petrol
Owner type:  Enter 0:first hand   ,      Enter 1: forth and above ,       enter 2: second    ,enter 3: third
Kilometers_Driven40000
Fuel_Type1
Transmission0
Owner_Type0
Mileage988
Engine1248
Power88
Seats4
New_Price32
No._of_Years4
predicted price is [12.]
```

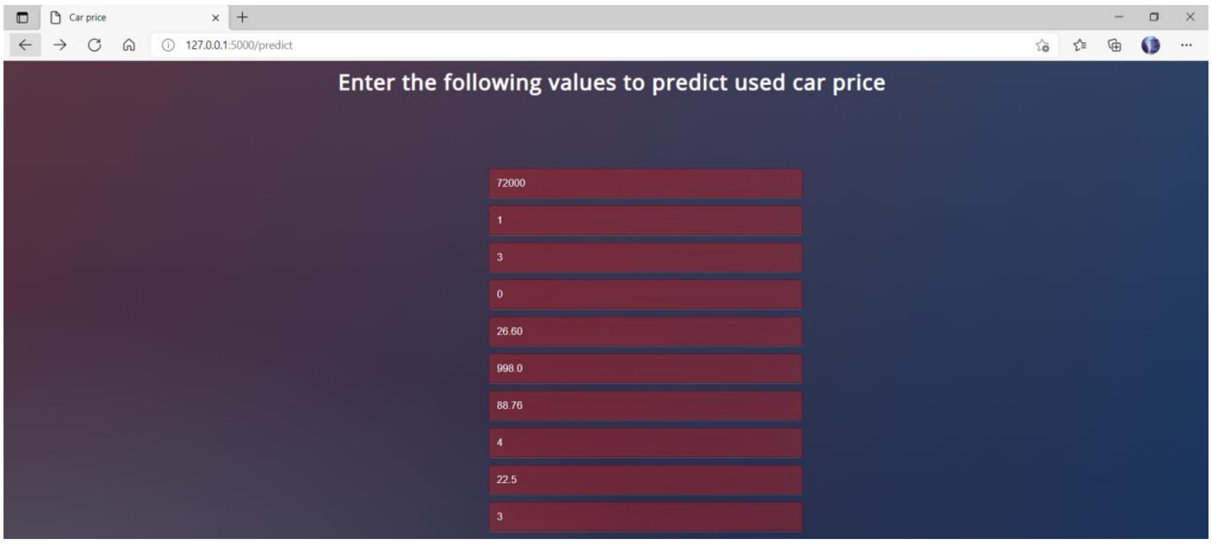
Updated with another cell for predicted price.



	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	No._of_Years	Predicted_price
0	Maruti Alto K10 LXI CNG	Delhi	2014	40929	0	1	0	32.26	998.0	58.20	1	0.00	7	3.0
1	Maruti Alto 800 2016-2019 LXI	Coimbatore	2013	54493	3	1	2	24.70	796.0	47.30	2	0.00	8	2.0
2	Toyota Innova Crysta Touring Sport 2.4 MT	Mumbai	2017	34000	1	1	0	13.68	2393.0	147.80	4	25.27	4	17.0
3	Toyota Etios Liva GD	Hyderabad	2012	139000	1	1	0	23.59	1364.0	0.00	2	0.00	9	2.0
4	Hyundai i20 Magna	Mumbai	2014	29000	3	1	0	18.50	1197.0	82.85	2	0.00	7	4.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1229	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	1	1	0	20.54	1598.0	103.60	2	0.00	10	3.0
1230	Volkswagen Polo GT TSI	Mumbai	2015	59000	3	0	0	17.21	1197.0	103.60	2	0.00	6	6.0
1231	Nissan Micra Diesel XV	Kolkata	2012	28000	1	1	0	23.08	1461.0	63.10	2	0.00	9	2.0
1232	Volkswagen Polo GT TSI	Pune	2013	52262	3	0	3	17.20	1197.0	103.60	2	0.00	8	4.0
1233	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	1	0	0	10.00	2148.0	170.00	2	0.00	7	27.0

234 rows × 14 columns

Web application:



Enter the following values to predict used car price

72000

1

3

0

26.60

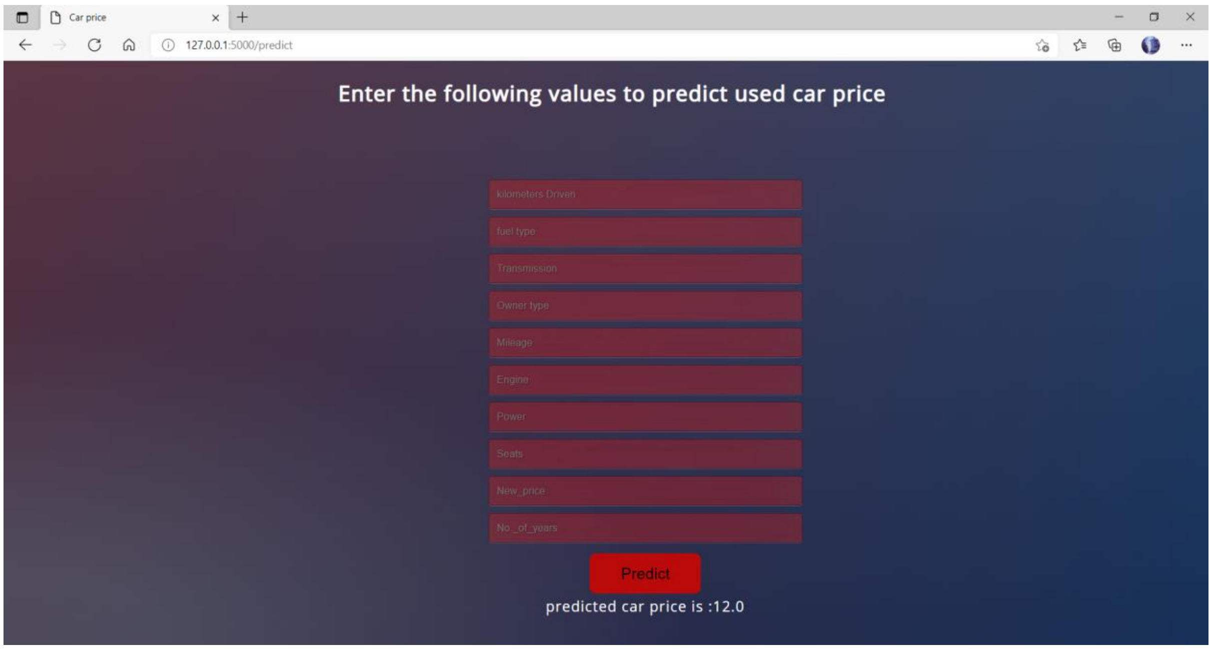
998.0

88.76

4

22.5

3



Enter the following values to predict used car price

kilometers Driven

fuel type

Transmission

Owner type

Mileage

Engine

Power

Seats

New_price

No_of_years

predicted car price is :12.0