



# CAR PRICE PREDICTION – PREDICTING THE PRICES OF USED CARS

Submitted by:  
NATASHA PODDAR

## **ACKNOWLEDGMENT**

The data used in the project was scraped from the following websites :

- CARDEKHO.com
- CARS24.com

# INTRODUCTION

- Business Problem Framing

With the covid 19 impact in the market, there have been a lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper. With the change in market due to covid 19 impact, there are difficulties in calculating the valuation of previous car prices. This problem had two steps, first was to scrape the data from various websites and second was to build various models for price prediction.

## Conceptual Background of the Domain Problem

A basic understanding on the car industry and the various factors which help in determining the price of a car is needed

- Review of Literature

COVID-19 has affected day to day life and is slowing down the global economy. Postponement of new vehicle purchases by customers and reduced exports have resulted in the aftermarket receiving increased attention from auto component suppliers, But with the increase in social distancing and other various factors, there is an increase in the sale of used cars. The automobile business, which had been speeding up throughout the previous two decades and has now gone into the reverse rig, has seen the worst situation come to light. Moreover, the current situation of the worldwide pandemic, with the lockdown, has induced work hardships and a declining demand in automobiles. Hence this segment has gained a lot of potential.

## Motivation for the Problem Undertaken

This issue is very realistic and common in today's world and one should know to deal with such situations in the future

# Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

Firstly missing values were checked and

Correlation with all independent variables and wrt target were checked

Outliers were identified through zscore but they were not removed as the data had more than 20% outliers

Skewness was checked and tools were applied to control them and scale the data

Multi colinearity was checked and worked upon

Models were applied to train and test the model

.

- Data Sources and their formats

The data used for the project was scraped from CARDEKHO & CARS24 website. The same was converted in an excel file.

```
In [206]: year=[]
aa=driver.find_elements(By.XPATH, '//h2[@class="_3FpCg"]')
for i in aa:
    year.append(i.text.split()[0])

In [207]: brand=[]
aaa=driver.find_elements(By.XPATH, '//h2[@class="_3FpCg"]')
for i in aaa:
    brand.append(i.text.split()[1])

In [208]: model=[]
aaaa=driver.find_elements(By.XPATH, '//h2[@class="_3FpCg"]')
for i in aaaa:
    model.append(i.text.split()[2:])

In [209]: km=[]
ss=driver.find_elements(By.XPATH, '//ul[@class="bVR0c"]')
for i in ss:
    km.append(i.text.split()[0])

In [210]: make=[]
sss=driver.find_elements(By.XPATH, '//ul[@class="bVR0c"]')
for i in sss:
```

	A	B	C	D	E	F	G	H	I	J	K	L	M			
	SOURCE	YEAR	BRAND	MODEL	CATEGORY	OWNERSH	KM	MAKE	TYPE	OPTION AVAI	ZDP AVAILABLE	LOCATION	PRICE			
1	CARS24	2019	Maruti	Swift	HATCHBACK	1st	12046	Petrol	Manual	YES	YES	DELHI	538799.00			
2	CARS24	2019	Maruti	Swift	HATCHBACK	1st	11404	Petrol	Manual	YES	YES	DELHI	534399.00			
3	CARS24	2018	Hyundai	I TEN	HATCHBACK	1st	6875	Petrol	Manual	YES	YES	DELHI	556899.00			
4	CARS24	2018	Datsun	Redi Go	HATCHBACK	1st	15576	Petrol	Manual	YES	YES	DELHI	223699.00			
5	CARS24	2018	Hyundai	I TEN	HATCHBACK	2nd	12362	Petrol	Manual	YES	YES	DELHI	521999.00			
6	CARS24	2019	Maruti	Swift	HATCHBACK	1st	14679	Petrol	Manual	YES	YES	DELHI	556199.00			
7	CARS24	2020	Maruti	Figgo	HATCHBACK	1st	16633	Petrol	Manual	YES	YES	DELHI	557199.00			
8	CARS24	2019	Maruti	Swift	HATCHBACK	1st	18981	Petrol	Manual	YES	YES	DELHI	563699.00			
9	CARS24	2015	Hyundai	Eon	HATCHBACK	1st	76800	Petrol	Manual	YES	YES	DELHI	236999.00			
10	CARS24	2012	Maruti	Ritz	HATCHBACK	2nd	32749	Petrol	Manual	YES	YES	DELHI	225299.00			
11	CARS24	2013	Maruti	Wagon R	HATCHBACK	2nd	38343	Petrol	Manual	YES	YES	DELHI	276699.00			
12	CARS24	2021	Hyundai	AURA	HATCHBACK	1st	19104	Petrol + CNG	Manual	YES	YES	DELHI	808399.00			
13	CARS24	2014	Mercedes-Benz	C Class	LUXURY SEDAN	1st	37531	Petrol	Automatic	YES	YES	DELHI	1929699.00			
14	CARS24	2020	KIA	SELTOS	SUBCOMPACT CROSSOVER SUV	1st	8241	Petrol	Automatic	YES	YES	DELHI	1722599.00			
15	CARS24	2019	Hyundai	I TEN	HATCHBACK	1st	20011	Petrol	Manual	YES	YES	DELHI	509899.00			
16	CARS24	2020	Maruti	Swift	HATCHBACK	1st	12031	Petrol	Manual	YES	YES	DELHI	592499.00			
17	CARS24	2019	Maruti	Swift	HATCHBACK	1st	24812	Petrol	Manual	YES	YES	DELHI	544599.00			
18	CARS24	2020	Maruti	Swift	HATCHBACK	1st	36255	Petrol + CNG	Manual	YES	YES	DELHI	569799.00			
19	CARS24	2011	Maruti	Wagon R	HATCHBACK	2nd	28876	Petrol	Manual	YES	YES	DELHI	281499.00			
20	CARS24	2022	Hyundai	Santro	HATCHBACK	1st	2315	Petrol + CNG	Manual	YES	YES	DELHI	688499.00			
21	CARS24	2020	Maruti	Swift	HATCHBACK	1st	17435	Petrol	Manual	YES	YES	DELHI	559999.00			
22	CARS24	2019	KIA	SELTOS	SUBCOMPACT CROSSOVER SUV	2nd	12757	Petrol	Automatic	YES	YES	DELHI	1667299.00			
23	CARS24	2012	Hyundai	Eon	HATCHBACK	2nd	43868	Petrol	Manual	YES	YES	DELHI	159999.00			
24	CARS24	2021	MG	HECTOR	SUV	1st	3587	Petrol	Manual	YES	YES	DELHI	1771299.00			
25	CARS24	2022	KIA	CARENS	MPV	1st	1892	Petrol	Manual	YES	YES	DELHI	1666999.00			
26	CARS24	2011	Maruti	Swift	HATCHBACK	3rd	59419	Petrol	Manual	YES	YES	DELHI	285699.00			
27	CARS24	2021	Tata	ALTROZ	HATCHBACK	1st	7789	Petrol	Manual	YES	YES	DELHI	720176.00			
28	CARS24	2019	Maruti	Swift	HATCHBACK	2nd	45291	Petrol	Manual	YES	YES	DELHI	545199.00			
29	CARS24	2021	Maruti	Swift	HATCHBACK	1st	5234	Petrol	Manual	YES	YES	DELHI	611299.00			
30	CARS24	2021	Nissan	MAGNITE	SUBCOMPACT CROSSOVER SUV	1st	9540	Petrol	Manual	YES	YES	DELHI	537399.00			
31	CARS24	2022	Hyundai	VENUE	SUBCOMPACT CROSSOVER SUV	1st	9463	Petrol	Manual	YES	YES	DELHI	968199.00			
32	CARS24	2012	Maruti	Swift Dzire	SEDAN	1st	61580	Petrol	Manual	YES	YES	DELHI	383699.00			
33	CARS24	2022	Renault	Kiger	SUBCOMPACT CROSSOVER SUV	1st	3271	Petrol	Manual	YES	YES	DELHI	843899.00			
34	CARS24	2020	KIA	SELTOS	SUBCOMPACT CROSSOVER SUV	1st	33981	Diesel	Automatic	YES	YES	DELHI	1369099.00			
35	CARS24	2013	Honda	Brio	HATCHBACK	1st	44620	Petrol	Manual	YES	YES	DELHI	280099.00			
36	CARS24	2021	Hyundai	I TWENTY	HATCHBACK	1st	5992	Petrol	Manual	YES	YES	DELHI	1059099.00			
37	CARS24	2012	Hyundai	I TEN	HATCHBACK	1st	21379	Petrol	Manual	YES	YES	DELHI	244199.00			
38	CARS24	2021	Maruti	Wagon R	HATCHBACK	2nd	2762	Petrol	Manual	YES	YES	DELHI	534799.00			
39	CARS24	2019	Maruti	Wagon R	HATCHBACK	1st	3373	Petrol	Manual	YES	YES	DELHI	507099.00			
40	CARS24	2020	Maruti	Baleno	HATCHBACK	2nd	10838	Petrol	Manual	YES	YES	DELHI	603599.00			
41	CARS24	2020	Maruti	Santro	HATCHBACK	1st	3856	Petrol	Manual	YES	YES	DELHI	458543.00			

- Data Preprocessing Done:
  1. Duplicate values check
  2. Unique & Count of all columns were checked
  3. Missing values were imputed
  4. Columns which had more than 40% data missing were removed
  5. Catagorical data was Encoded
  6. Skewness removal through Power Transform and scaling of the data
  7. VIF Check -for multicollinearity
  8. Correlation check
  9. Graphical Univariate, Bivariate & Multivariate Analysis
  - 10 Outliers check -ZSCORE

- Data Inputs- Logic- Output Relationships  
Mostly all the columns had low correlation with the target column, both positive and negative in nature

- Hardware and Software Requirements and Tools Used
  1. Pandas – For Data Reading and understanding
  2. Label Encoder –(SK LEARN) – For Encoding the categorical data into numerical ones
  3. Zscore(SCIPY)-For checking & removal of outliers
  4. Power Transform ()- Skewness removal
  5. Duplicate- To check for duplicate Values
  6. CORR-To check Correlation
  7. VIF -To check for multicollinearity
  8. Numpy- For mathematical operations
  9. LOGITSIC REGRESSION (SKLEARN) – Training & Testing the model
  10. KNN REGRESSOR (SKLEARN) – Training & Testing the model
  11. DECISION TREE REGRESSOR (SKLEARN) – Training & Testing the model
  12. RANDOM FOREST REGRESSOR (SKLEARN) – Training & Testing the model
  13. GRADIENT BOOSTING REGRESSOR (SKLEARN) – Training & Testing the model
  14. CROSS VAL SCORE – Regularizing the model
  15. GRID SEARCH CV- Hyper Tuning the Model for higher accuracy
  16. SEABORN- VISUALIZATION LIBRARY – HISTPLOTS, DISTPLOTS, SCATTERPLOTS, COUNTPLOTS, BOXPLOTS and other graphs
  17. MATPLOTLIB.PY PLOT -Visualization tool

## **Model/s Development and Evaluation**

- Identification of possible problem-solving approaches (methods)
  1. Firstly missing values were checked .
  2. Correlation with all independent variables and wrt target were checked
  3. Outliers were identified but they were NOT removed through zscore as the data had more than 20% outliers present
  4. Skewness was checked and tools were applied to control them and scale the data
  5. Multi colinearity was checked and worked upon
  6. Models were applied to train and test the model
- Testing of Identified Approaches (Algorithms)
  1. LINEAR REGRESSION
  2. KNN REGRESSOR
  3. DECISION TREE REGRESSOR
  4. RANDOM FOREST REGRESSOR
  5. GRADIENT BOOSTING REGRESSOR

# Run and Evaluate selected models

The image displays two screenshots of a Jupyter Notebook titled "CAR PREDICTION MODEL". The notebook is running on a local host (localhost:8889) and is using Python 3 (ipykernel). The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and saving.

**First Screenshot:** The notebook shows a code cell with the following Python code:

```
from sklearn.linear_model import LinearRegression

In [403]: lr=LinearRegression()

In [404]: for i in range(0,100):
           x_train,x_test,y_train,y_test=train_test_split(xnew,y,test_size=0.20,random_state=i)
           lr.fit(x_train,y_train)
           predr=lr.predict(x_train)
           pred=lr.predict(x_test)
           print('At random_state',i), 'the training accuracy test is', r2_score(y_train,pedr))
           print('At random_state',i), 'the testing accuracy test is', r2_score(y_test,pred)
           print('\n')
```

The output of the code cell shows the training and testing accuracy for random states 0 to 4:

```
At random_state {0} the training accuracy test is 0.16723124325851746
At random_state {0} the testing accuracy test is 0.1456655798226819

At random_state {1} the training accuracy test is 0.15599034868277306
At random_state {1} the testing accuracy test is 0.19426502544739577

At random_state {2} the training accuracy test is 0.15378318235172395
At random_state {2} the testing accuracy test is 0.196911841156716

At random_state {3} the training accuracy test is 0.15254554796168973
At random_state {3} the testing accuracy test is 0.1984046517258362

At random_state {4} the training accuracy test is 0.16666106071745934
At random_state {4} the testing accuracy test is 0.1477314086013567
```

Below the output, the text states: "AT RANDOM STATE 47, WE HAVE THE HIGHEST ACCURACY OF 21.45%, HENCE WE WILL PROCEED WITH THAT".

**Second Screenshot:** The notebook shows the same code cell, but the output is truncated, showing only the results for random state 11:

```
At random_state {11} the training accuracy test is 0.15626829085916893
At random_state {11} the testing accuracy test is 0.1929845489344001
```

Below the output, the text states: "AT RANDOM STATE 47, WE HAVE THE HIGHEST ACCURACY OF 21.45%, HENCE WE WILL PROCEED WITH THAT".

The code cell in the second screenshot includes the following Python code:

```
In [405]: x_train,x_test,y_train,y_test=train_test_split(xnew,y,test_size=0.2,random_state=47)

In [406]: lr.fit(x_train,y_train)

Out[406]: LinearRegression()
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [407]: PRELR=lr.predict(x_test)

In [408]: print(r2_score(y_test,PRELR))

0.21453899265302023

In [409]: from sklearn.metrics import mean_squared_error,mean_absolute_error

In [410]: print(mean_squared_error(y_test,PRELR))

180707400960.58194

In [411]: print(mean_absolute_error(y_test,PRELR))

285131.48208522936

In [412]: print(sqrt(mean_squared_error(y_test,PRELR)))

425096.9312528402
```



Home Page - Select or create x CAR PREDICTION MODEL - Jupyter x DATA COLLECTION - CAR PREDICTION x COVID-19 Impact on Used Car x +

localhost:8889/notebooks/CAR%20PREDICTION%20MODEL%20.ipynb

Jupyter CAR PREDICTION MODEL Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

### DECISION TREE REGRESSOR

```
In [351]: dtc=DecisionTreeRegressor()

In [352]: for i in range(0,100):
            x_train,x_test,y_train,y_test=train_test_split(xnew,y,test_size=0.20,random_state=i)
            dtc.fit(x_train,y_train)
            predtc=dtc.predict(x_train)
            predtc1=dtc.predict(x_test)
            print('At random_state',i), 'the training accuracy test is', r2_score(y_train,predtc))
            print('At random_state',i), 'the testing accuracy test is', r2_score(y_test,predtc1))
            print('\n')
```

At random\_state {0} the training accuracy test is 0.9989461778253284  
At random\_state {0} the testing accuracy test is 0.6403490388769837

At random\_state {1} the training accuracy test is 0.9990361462976899  
At random\_state {1} the testing accuracy test is 0.6346947729452908

At random\_state {2} the training accuracy test is 0.9993816030159495  
At random\_state {2} the testing accuracy test is 0.5609649317163037

At random\_state {3} the training accuracy test is 0.9992489675822144  
At random\_state {3} the testing accuracy test is 0.3296052547443419

At random\_state {4} the training accuracy test is 0.999114386119175  
At random\_state {4} the testing accuracy test is 0.49122384140140996

**AT RANDOM STATE 99, WE HAVE THE HIGHEST ACCURACY OF 77.89%**

DATA COLLEC...ipynb CAR PREDICT...ipynb Show all x

Home Page - Select or create x CAR PREDICTION MODEL - Jupyter x DATA COLLECTION - CAR PREDICTION x COVID-19 Impact on Used Car x +

localhost:8889/notebooks/CAR%20PREDICTION%20MODEL%20.ipynb

Jupyter CAR PREDICTION MODEL Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

At random\_state {4} the training accuracy test is 0.999114386119175  
At random\_state {4} the testing accuracy test is 0.49122384140140996

**AT RANDOM STATE 99, WE HAVE THE HIGHEST ACCURACY OF 77.89%**

```
In [370]: x_train,x_test,y_train,y_test=train_test_split(xnew,y,test_size=0.2,random_state=99)

In [371]: dtc.fit(x_train,y_train)

Out[371]: DecisionTreeRegressor()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [372]: PREDTC=dtc.predict(x_test)

In [373]: print(r2_score(y_test,PREDTC))
0.7987341670117821

In [374]: print(mean_squared_error(y_test,PREDTC))
64426896033.6702

In [375]: print(mean_absolute_error(y_test,PREDTC))
120228.2240673887

In [376]: print(sqrt(mean_squared_error(y_test,PREDTC)))
253824.53788723855
```

DATA COLLEC...ipynb CAR PREDICT...ipynb Show all x

Home Page - Select or create x CAR PREDICTION MODEL - Jupyter x DATA COLLECTION - CAR PREDICTION x COVID-19 Impact on Used Car x +

localhost:8889/notebooks/CAR%20PREDICTION%20MODEL%20.ipynb

Jupyter CAR PREDICTION MODEL Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

### KNN REGRESSOR

```
In [353]: knn=KNeighborsRegressor()

In [354]: for i in range(0,100):
            x_train,x_test,y_train,y_test=train_test_split(xnew,y,test_size=0.20,random_state=i)
            knn.fit(x_train,y_train)
            preknn=knn.predict(x_train)
            preknn1=knn.predict(x_test)
            print('At random_state',{i}, 'the training accuracy test is', r2_score(y_train,preknn))
            print('At random_state',{i}, 'the testing accuracy test is', r2_score(y_test,preknn1))
            print('\n')
```

At random\_state {0} the training accuracy test is 0.6449931475249209  
At random\_state {0} the testing accuracy test is 0.3639154661911914

At random\_state {1} the training accuracy test is 0.6687152036966533  
At random\_state {1} the testing accuracy test is 0.4315135593966356

At random\_state {2} the training accuracy test is 0.661543750196812  
At random\_state {2} the testing accuracy test is 0.4378864678967497

At random\_state {3} the training accuracy test is 0.631979131702658  
At random\_state {3} the testing accuracy test is 0.4968604840905839

At random\_state {4} the training accuracy test is 0.6573656045127805  
At random\_state {4} the testing accuracy test is 0.30879144072318854

**AT RANDOM STATE 16, WE HAVE THE HIGHEST ACCURACY OF 60.20%**

Home Page - Select or create x CAR PREDICTION MODEL - Jupyter x DATA COLLECTION - CAR PREDICTION x COVID-19 Impact on Used Car x +

localhost:8889/notebooks/CAR%20PREDICTION%20MODEL%20.ipynb

Jupyter CAR PREDICTION MODEL Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

At random\_state {4} the training accuracy test is 0.6573656045127805  
At random\_state {4} the testing accuracy test is 0.30879144072318854

**AT RANDOM STATE 16, WE HAVE THE HIGHEST ACCURACY OF 60.20%**

```
In [377]: x_train,x_test,y_train,y_test=train_test_split(xnew,y,test_size=0.2,random_state=16)

In [378]: knn.fit(x_train,y_train)

Out[378]: KNeighborsRegressor()
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [379]: PREKNN=knn.predict(x_test)

In [380]: print(r2_score(y_test,PREKNN))
0.602068253167668

In [381]: print(mean_squared_error(y_test,PREKNN))
140300061140.57812

In [382]: print(mean_absolute_error(y_test,PREKNN))
185926.03942238266

In [383]: print(sqrt(mean_squared_error(y_test,PREKNN)))
374566.4976216882
```

Home Page - Select or create x CAR PREDICTION MODEL - Jupyter x DATA COLLECTION - CAR PREDICTION x COVID-19 Impact on Used Car x +

localhost:8889/notebooks/CAR%20PREDICTION%20MODEL%20.ipynb

Jupyter CAR PREDICTION MODEL Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

### RANDOM FOREST REGRESSOR

```
In [355]: rm=RandomForestRegressor()

In [356]: for i in range(0,100):
            x_train,x_test,y_train,y_test=train_test_split(xnew,y,test_size=0.20,random_state=i)
            rm.fit(x_train,y_train)
            prerm=rm.predict(x_train)
            prerm1=rm.predict(x_test)
            print('At random_state',i), 'the training accuracy test is', r2_score(y_train,prerm))
            print('At random_state',i), 'the testing accuracy test is', r2_score(y_test,prerm1))
            print('\n')
```

At random\_state {0} the training accuracy test is 0.9555905847140681  
At random\_state {0} the testing accuracy test is 0.7869576314682576

At random\_state {1} the training accuracy test is 0.9630856377481753  
At random\_state {1} the testing accuracy test is 0.71805216482

At random\_state {2} the training accuracy test is 0.9586852617351039  
At random\_state {2} the testing accuracy test is 0.7249334449672771

At random\_state {3} the training accuracy test is 0.9637083184111567  
At random\_state {3} the testing accuracy test is 0.51873374536178

At random\_state {4} the training accuracy test is 0.9711400868227036  
At random\_state {4} the testing accuracy test is 0.6035867656998379

**AT RANDOM STATE 16, WE HAVE THE HIGHEST ACCURACY OF 86.79%**

DATA COLLEC...ipynb CAR PREDICT...ipynb Show all x

Home Page - Select or create x CAR PREDICTION MODEL - Jupyter x DATA COLLECTION - CAR PREDICTION x COVID-19 Impact on Used Car x +

localhost:8889/notebooks/CAR%20PREDICTION%20MODEL%20.ipynb

Jupyter CAR PREDICTION MODEL Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [384]: x_train,x_test,y_train,y_test=train_test_split(xnew,y,test_size=0.2,random_state=16)

In [385]: rm.fit(x_train,y_train)

Out[385]: RandomForestRegressor()

In [386]: PRERM=rm.predict(x_test)

In [387]: print(r2_score(y_test,PRERM))
0.8619638308522724

In [388]: print(mean_squared_error(y_test,PRERM))
48667851020.183655

In [389]: print(mean_absolute_error(y_test,PRERM))
110113.81299922575

In [390]: print(sqrt(mean_squared_error(y_test,PRERM)))
220607.91241517983
```

**AT RANDOM STATE 16, WE HAVE THE HIGHEST ACCURACY OF 86.79%**

DATA COLLEC...ipynb CAR PREDICT...ipynb Show all x

Home Page - Select or create x CAR PREDICTION MODEL - Jupyter x DATA COLLECTION - CAR PREDICTION x COVID-19 Impact on Used Car x +

localhost:8889/notebooks/CAR%20PREDICTION%20MODEL%20.ipynb

Jupyter CAR PREDICTION MODEL Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

### GRADIENT BOOSTING REGRESSOR

```
In [357]: xg=GradientBoostingRegressor()

In [358]: for i in range(0,100):
            x_train,x_test,y_train,y_test=train_test_split(xnew,y,test_size=0.20,random_state=i)
            xg.fit(x_train,y_train)
            prexg=xg.predict(x_test)
            print('At random_state' ,i), 'the training accuracy test is', r2_score(y_train,prexg))
            print('At random_state' ,i), 'the testing accuracy test is', r2_score(y_test,prexg1))
            print('\n')
```

At random\_state {0} the training accuracy test is 0.7438257509905621  
At random\_state {0} the testing accuracy test is 0.6892907515764062

At random\_state {1} the training accuracy test is 0.7956622034478977  
At random\_state {1} the testing accuracy test is 0.7111567585678187

At random\_state {2} the training accuracy test is 0.7773035010383585  
At random\_state {2} the testing accuracy test is 0.6677688478847015

At random\_state {3} the training accuracy test is 0.7815344441305353  
At random\_state {3} the testing accuracy test is 0.6598969308249277

At random\_state {4} the training accuracy test is 0.7976205584080982  
At random\_state {4} the testing accuracy test is 0.5607203111715279

**AT RANDOM STATE 27, WE HAVE THE HIGHEST ACCURACY OF 78.58%**

DATA COLLEC...ipynb CAR PREDICT...ipynb Show all x

Home Page - Select or create x CAR PREDICTION MODEL - Jupyter x DATA COLLECTION - CAR PREDICTION x COVID-19 Impact on Used Car x +

localhost:8889/notebooks/CAR%20PREDICTION%20MODEL%20.ipynb

Jupyter CAR PREDICTION MODEL Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

At random\_state {4} the training accuracy test is 0.7976205584080982  
At random\_state {4} the testing accuracy test is 0.5607203111715279

**AT RANDOM STATE 27, WE HAVE THE HIGHEST ACCURACY OF 78.58%**

```
In [391]: x_train,x_test,y_train,y_test=train_test_split(xnew,y,test_size=0.2,random_state=27)

In [392]: xg.fit(x_train,y_train)

Out[392]: GradientBoostingRegressor()
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [395]: PREXGB=xg.predict(x_test)

In [396]: print(r2_score(y_test,PREXGB))
0.7858162277821326

In [397]: print(mean_squared_error(y_test,PREXGB))
60647145306.8412

In [398]: print(mean_absolute_error(y_test,PREXGB))
151975.14756885506

In [399]: print(sqrt(mean_squared_error(y_test,PREXGB)))
246266.41124367976
```

DATA COLLEC...ipynb CAR PREDICT...ipynb Show all x

- Key Metrics for success in solving problem under consideration
  1. R2 SCORE
  2. MEAN SQUARED ERROR
  3. MEAN ABSOLUTE ERROR
  4. ROOT SQUARE MEAN ERROR
- Visualizations

Seaborn Library was used along with matplotlib Library for visualizations

Histplots, bar plots, count plots, swarmplots, boxplots etc were made and analysed
- Interpretation of the Results

RANDOM FOREST REGRESSOR had the highest model accuracy and the difference between CV MEAN SCORE & MODEL ACCURACY SCORE was also not very high hence we had hyper tuned the said model and saved the same

# CONCLUSION

- Key Findings and Conclusions of the Study

Random Forest Regressor without hyper tuning had a higher accuracy and the same model was selected and saved

- Learning Outcomes of the Study in respect of Data Science

Strong insights were derived from the various visualization tools which helped in understanding the various relationships between the target and other variables

The screenshot shows a Jupyter Notebook titled "CAR PREDICTION MODEL" with the following content:

**THE SCORE HAS NOT IMPROVED WRT BOTH CV MEAN SCORE AND MODEL ACCURACY SCORE, HENCE WE WILL SELECT AND SAVE 'rm' MODEL ONLY**

**SAVING THE BEST MODEL**

```
In [493]: import pickle

In [494]: filename='carprediction.pkl'
          pickle.dump(rm,open(filename,'wb'))

In [495]: loaded_model=pickle.load(open('carprediction.pkl','rb'))

In [496]: conclusion=pd.DataFrame([loaded_model.predict(x_test[:]),y_test[:]],index=['PREDICTED','ORIGINAL'])

In [497]: conclusion
```

Out[497]:

	0	1	2	3	4	5	6	7	8	9	...	1375	1376	1377	1378
PREDICTED	691982.0	374135.0	508783.99	405686.0	678751.0	670940.5	421152.2	904364.04	352731.0	507815.0	...	612908.01	257130.0	591589.0	392437.0
ORIGINAL	679699.0	392999.0	475000.00	380599.0	848099.0	738399.0	406899.0	655099.00	349299.0	498299.0	...	602599.00	524000.0	607049.0	300599.0

2 rows x 1385 columns

In [ ]: