Dataset: 'cardata.csv'

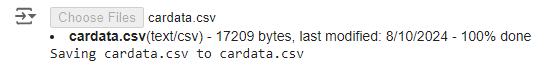
**LINEAR REGRESSION ALGORITHM**

If distribution is normal then linear regression can be applied.

**For uploading data in a googlecolab**

from google.colab import files

uploaded = files.upload()



**To see that uploaded file**

import pandas as pd

import io

df = pd.read\_csv(io.BytesIO(uploaded['cardata.csv']))

print(df)

car\_info.info()

car\_info.describe()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 301 entries, 0 to 300

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Car\_Name 301 non-null object

1 Year 301 non-null int64

2 Selling\_Price 301 non-null float64

3 Present\_Price 301 non-null float64

4 Kms\_Driven 301 non-null int64

5 Fuel\_Type 301 non-null object

6 Seller\_Type 301 non-null object

7 Transmission 301 non-null object

8 Owner 301 non-null int64

dtypes: float64(2), int64(3), object(4)

memory usage: 21.3+ KB

|  | **Year** | **Selling\_Price** | **Present\_Price** | **Kms\_Driven** | **Owner** |
| --- | --- | --- | --- | --- | --- |
| **count** | 301.000000 | 301.000000 | 301.000000 | 301.000000 | 301.000000 |
| **mean** | 2013.627907 | 4.661296 | 7.628472 | 36947.205980 | 0.043189 |
| **std** | 2.891554 | 5.082812 | 8.644115 | 38886.883882 | 0.247915 |
| **min** | 2003.000000 | 0.100000 | 0.320000 | 500.000000 | 0.000000 |
| **25%** | 2012.000000 | 0.900000 | 1.200000 | 15000.000000 | 0.000000 |
| **50%** | 2014.000000 | 3.600000 | 6.400000 | 32000.000000 | 0.000000 |
| **75%** | 2016.000000 | 6.000000 | 9.900000 | 48767.000000 | 0.000000 |
| **max** | 2018.000000 | 35.000000 | 92.600000 | 500000.000000 | 3.000000 |

To change column name to lower case

Car\_info.columns= [ cols.lower() for cols in car\_info.columns ]

->car\_info.columns=[ cols.lower() for cols in car\_info.columns ]

car\_info.head()

| **car\_name** | **year** | **selling\_price** | **present\_price** | **kms\_driven** | **fuel\_type** | **seller\_type** | **transmission** | **owner** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | ritz | 2014 | 3.35 | 5.59 | 27000 | Petrol | Dealer | Manual | 0 |
| **1** | sx4 | 2013 | 4.75 | 9.54 | 43000 | Diesel | Dealer | Manual | 0 |
| **2** | ciaz | 2017 | 7.25 | 9.85 | 6900 | Petrol | Dealer | Manual | 0 |
| **3** | wagon r | 2011 | 2.85 | 4.15 | 5200 | Petrol | Dealer | Manual | 0 |
| **4** | swift | 2014 | 4.60 | 6.87 | 42450 | Diesel | Dealer | Manual | 0 |

**To display the values in fuel type**

* print(car\_info['fuel\_type'].value\_counts())

fuel\_type

Petrol 239

Diesel 60

CNG 2

Name: count, dtype: int64

* **To display the style**

import matplotlib.pyplot as plt

from matplotlib import style

style.available // It will list out available style

['Solarize\_Light2',

'\_classic\_test\_patch',

'\_mpl-gallery',

'\_mpl-gallery-nogrid',

'bmh',

'classic',

'dark\_background',

'fast',

'fivethirtyeight',

'ggplot',

'grayscale',

'seaborn-v0\_8',

'seaborn-v0\_8-bright',

'seaborn-v0\_8-colorblind',

'seaborn-v0\_8-dark',

'seaborn-v0\_8-dark-palette',

'seaborn-v0\_8-darkgrid',

'seaborn-v0\_8-deep',

'seaborn-v0\_8-muted',

'seaborn-v0\_8-notebook',

'seaborn-v0\_8-paper',

'seaborn-v0\_8-pastel',

'seaborn-v0\_8-poster',

'seaborn-v0\_8-talk',

'seaborn-v0\_8-ticks',

'seaborn-v0\_8-white',

'seaborn-v0\_8-whitegrid',

'tableau-colorblind10']

fuel\_info=car\_info['fuel\_type']

seller\_info=car\_info['seller\_type']

transmission\_info=car\_info['transmission']

selling\_price=car\_info['selling\_price']

* print(fuel\_info)

0 Petrol

1 Diesel

2 Petrol

3 Petrol

4 Diesel

...

296 Diesel

297 Petrol

298 Petrol

299 Diesel

300 Petrol

* Name: fuel\_type, Length: 301, dtype: object
* **To display in chart**

style.use('ggplot')

chart=plt.figure(figsize=(20,5))

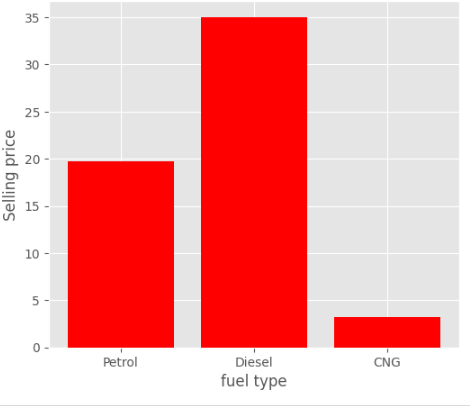
chart.suptitle('categorical chart info')

plt.subplot(1,3,1)

plt.bar(fuel\_info,selling\_price,color='Red')

plt.xlabel("fuel type")

plt.ylabel("Selling price")



style.use('ggplot')

chart=plt.figure(figsize=(20,5))

chart.suptitle('categorical chart info')

plt.subplot(1,3,2)

plt.bar(seller\_info,selling\_price,color='green')

plt.xlabel("selling type")

plt.ylabel("Selling price")

plt.subplot(1,3,3)

plt.bar(transmission\_info,selling\_price,color='yellow')

plt.xlabel("transmission type")

plt.ylabel("Selling price")

**To display petrol data**

petrol\_data=car\_info.groupby('fuel\_type').get\_group('Petrol')

petrol\_data.describe()

->



**To display Diesel data**

Diesel\_data=car\_info.groupby('fuel\_type').get\_group('Diesel')

Diesel\_data.describe()



**// To display fuel-type,seller-type,transmission**

print(car\_info['fuel\_type'],car\_info['seller\_type'],car\_info['transmission'])

0 Petrol

1 Diesel

2 Petrol

3 Petrol

4 Diesel

...

296 Diesel

297 Petrol

298 Petrol

299 Diesel

300 Petrol

Name: fuel\_type, Length: 301, dtype: object 0 Dealer

1 Dealer

2 Dealer

3 Dealer

4 Dealer

...

296 Dealer

297 Dealer

298 Dealer

299 Dealer

300 Dealer

Name: seller\_type, Length: 301, dtype: object 0 Manual

1 Manual

2 Manual

3 Manual

4 Manual

...

296 Manual

297 Manual

298 Manual

299 Manual

300 Manual

Name: transmission, Length: 301, dtype: object

**Before changing car\_info**

Car\_Name Year Selling\_Price Present\_Price Kms\_Driven Fuel\_Type \

0 ritz 2014 3.35 5.59 27000 Petrol

1 sx4 2013 4.75 9.54 43000 Diesel

2 ciaz 2017 7.25 9.85 6900 Petrol

3 wagon r 2011 2.85 4.15 5200 Petrol

4 swift 2014 4.60 6.87 42450 Diesel

.. ... ... ... ... ... ...

296 city 2016 9.50 11.60 33988 Diesel

297 brio 2015 4.00 5.90 60000 Petrol

298 city 2009 3.35 11.00 87934 Petrol

299 city 2017 11.50 12.50 9000 Diesel

300 brio 2016 5.30 5.90 5464 Petrol

Seller\_Type Transmission Owner

0 Dealer Manual 0

1 Dealer Manual 0

2 Dealer Manual 0

3 Dealer Manual 0

4 Dealer Manual 0

.. ... ... ...

296 Dealer Manual 0

297 Dealer Manual 0

298 Dealer Manual 0

299 Dealer Manual 0

300 Dealer Manual 0

**//To replace text value(for fuel\_type,seller\_type,transmission) to numerical value in dataset , because computer understands 0 and 1**

car\_info.replace({'fuel\_type':{'Petrol':0,'Diesel':1,'CNG':2}},inplace=True)

car\_info.replace({'seller\_type':{'Dealer':1, 'Individual':2}},inplace=True)

car\_info.replace({'transmission':{'Manual':1, 'Automatic':2}},inplace=True)

**o/p:**

car\_name year selling\_price present\_price kms\_driven fuel\_type \

0 ritz 2014 3.35 5.59 27000 0

1 sx4 2013 4.75 9.54 43000 1

2 ciaz 2017 7.25 9.85 6900 0

3 wagon r 2011 2.85 4.15 5200 0

4 swift 2014 4.60 6.87 42450 1

.. ... ... ... ... ... ...

296 city 2016 9.50 11.60 33988 1

297 brio 2015 4.00 5.90 60000 0

298 city 2009 3.35 11.00 87934 0

299 city 2017 11.50 12.50 9000 1

300 brio 2016 5.30 5.90 5464 0

seller\_type transmission owner

0 1 1 0

1 1 1 0

2 1 1 0

3 1 1 0

4 1 1 0

.. ... ... ...

296 1 1 0

297 1 1 0

298 1 1 0

299 1 1 0

300 1 1 0

[301 rows x 9 columns]

**Car\_info.describe()**

yearselling\_pricepresent\_pricekms\_drivenfuel\_typeseller\_typetransmissionownercount301.000000301.000000301.000000301.000000301.000000301.000000301.000000301.000000mean2013.6279074.6612967.62847236947.2059800.2126251.3521591.1328900.043189std2.8915545.0828128.64411538886.8838820.4258010.4784390.3400210.247915min2003.0000000.1000000.320000500.0000000.0000001.0000001.0000000.00000025%2012.0000000.9000001.20000015000.0000000.0000001.0000001.0000000.00000050%2014.0000003.6000006.40000032000.0000000.0000001.0000001.0000000.00000075%2016.0000006.0000009.90000048767.0000000.0000002.0000001.0000000.000000max2018.00000035.00000092.600000500000.0000002.0000002.0000002.0000003.000000

// to display present price and seller price

// to regression algorithm we need seaborn library

// graph will be displayed

import seaborn as sea

import matplotlib.pyplot as plt

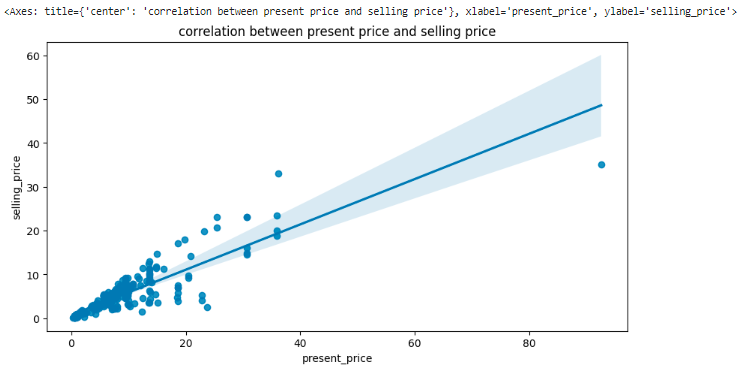
from matplotlib import style

fig=plt.figure(figsize=(10,5))

plt.title('correlation between present price and selling price')

sea.regplot(x='present\_price',y='selling\_price',data=car\_info)

<Axes: title={'center': 'correlation between present price and selling price'}, xlabel='present\_price', ylabel='selling\_price'>



**Now machine learning model is going to be generated.**

**// fixing input and output variable, here selling price is needed so it is considered as output variable so out\_data variable is taken**

**// for input data, we don’t require car\_name and selling price so we are //removing**

Input\_data=car\_info.drop(columns=[‘car\_name’,’selling\_price’])

Out\_data=car\_info[‘selling\_price’]

print(out\_data)

output:

0 3.35

1 4.75

2 7.25

3 2.85

4 4.60

...

296 9.50

297 4.00

298 3.35

299 11.50

300 5.30

Name: selling\_price, Length: 301, dtype: float64

**// To print input and output data after executing above query**

print(input\_data.shape)

print(out\_data.shape)

(301, 7)

(301,)

==========

// To normarlize data we use standard scaler

// To have same normalized data

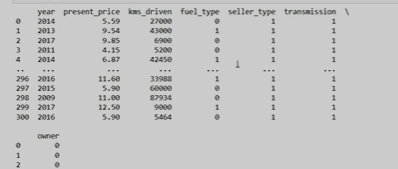
from sklearn.preprocessing import StandardScaler

scaler=StandardScaler() // using standardscaler we are making some input

input\_data=scaler.fit\_transform(input\_data) // normalizing input data

print(input\_data)

Before Normalizing:



After Normalizing

output: [[ 0.128897 -0.23621461 -0.25622446 ... -0.73728539 -0.39148015

-0.17450057]

[-0.21751369 0.22150462 0.1559105 ... -0.73728539 -0.39148015

-0.17450057]

[ 1.16812909 0.25742689 -0.77396901 ... -0.73728539 -0.39148015

-0.17450057]

...

[-1.60315648 0.39068691 1.31334003 ... -0.73728539 -0.39148015

-0.17450057]

[ 1.16812909 0.56450434 -0.7198763 ... -0.73728539 -0.39148015

-0.17450057]

[ 0.8217184 -0.20029235 -0.81095812 ... -0.73728539 -0.39148015

-0.17450057]]

// creating input and output data for training and testing

from sklearn.model\_selection import train\_test\_split // for training and test //split we use

input\_data\_train,input\_data\_test,output\_data\_train, output\_data\_test=train\_test\_split(input\_data,out\_data,test\_size=0.3)

print(input\_data\_train.shape)

print(output\_data\_train.shape)

print(input\_data\_test.shape)

print(input\_data\_test.shape)

//ouput

(210, 7)

(210,)

(91, 7)

(91, 7)

From sklearn.linear\_model import LinearRegression

Fron sklearn import metrics // to import metrics

model= LinearRegression()

model.fit(input\_data\_train,output\_data\_train) // fitting the model // train it

/ from sklearn.linear\_model import LinearRegression

from sklearn import metrics

model= LinearRegression()

model.fit(input\_data\_train,output\_data\_train)

/

LinearRegression

LinearRegression()

predicted\_sellingprice=model.predict(input\_data\_test)

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

print("mean absolute error:",metrics. mean\_squared\_error(predicted\_sellingprice,output\_data\_test))

print("mean squared error:",metrics. mean\_absolute\_error(predicted\_sellingprice, output\_data\_test))

print("R2 score:",metrics.r2\_score(predicted\_sellingprice, output\_data\_test))

mean absolute error: 3.989831769771728 // this should be less- Hence it will be efficient

mean squared error: 1.3011068460179196 // this should be less- Hence it will be efficient

R2 score: 0.7751257347279921 // this should be more- Hence it will be efficient, this is overall performance of machine learning score.