To predict the selling price of a car

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.feature_extraction.text import TfidfVectorizer
from pandas.plotting import scatter_matrix
import math
import nltk
import pickle
```

getting the dataset

In [ ]:

```
car_data = pd.read_csv('car data.csv')
car_data
```

Out[ ]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transm
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	N
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	N
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	N
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	N
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	N
...	...	...	...	...	...	...	...	
296	city	2016	9.50	11.60	33988	Diesel	Dealer	N
297	brio	2015	4.00	5.90	60000	Petrol	Dealer	N
298	city	2009	3.35	11.00	87934	Petrol	Dealer	N
299	city	2017	11.50	12.50	9000	Diesel	Dealer	N
300	brio	2016	5.30	5.90	5464	Petrol	Dealer	N

301 rows × 9 columns

In [ ]:

```
car_data = pd.DataFrame(car_data)
```

In [ ]:

```
car_data.info()
```

```
<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
```

In [ ]:

```
car_data['Fuel_Type'].unique()
```

Out[ ]:

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

In [ ]:

```
count = car_data['Car_Name'].value_counts()
threshold = 7.5
repl = count[count <= threshold].index
print(car_data['Car_Name'].unique())
```

```
['ritz' 'sx4' 'ciaz' 'wagon r' 'swift' 'vitara brezza' 's cross'
 'alto 800' 'ertiga' 'dzire' 'alto k10' 'ignis' '800' 'baleno' 'omni'
 'fortuner' 'innova' 'corolla altis' 'etios cross' 'etios g' 'etios liva'
 'corolla' 'etios gd' 'camry' 'land cruiser' 'Royal Enfield Thunder 500'
 'UM Renegade Mojave' 'KTM RC200' 'Bajaj Dominar 400'
 'Royal Enfield Classic 350' 'KTM RC390' 'Hyosung GT250R'
 'Royal Enfield Thunder 350' 'KTM 390 Duke ' 'Mahindra Mojo XT300'
 'Bajaj Pulsar RS200' 'Royal Enfield Bullet 350'
 'Royal Enfield Classic 500' 'Bajaj Avenger 220' 'Bajaj Avenger 150'
 'Honda CB Hornet 160R' 'Yamaha FZ S V 2.0' 'Yamaha FZ 16'
 'TVS Apache RTR 160' 'Bajaj Pulsar 150' 'Honda CBR 150' 'Hero Extreme'
 'Bajaj Avenger 220 dtsi' 'Bajaj Avenger 150 street' 'Yamaha FZ  v 2.0'
 'Bajaj Pulsar  NS 200' 'Bajaj Pulsar 220 F' 'TVS Apache RTR 180'
 'Hero Passion X pro' 'Bajaj Pulsar NS 200' 'Yamaha Fazer '
 'Honda Activa 4G' 'TVS Sport ' 'Honda Dream Yuga '
 'Bajaj Avenger Street 220' 'Hero Splender iSmart' 'Activa 3g'
 'Hero Passion Pro' 'Honda CB Trigger' 'Yamaha FZ S '
 'Bajaj Pulsar 135 LS' 'Activa 4g' 'Honda CB Unicorn'
 'Hero Honda CBZ extreme' 'Honda Karizma' 'Honda Activa 125' 'TVS Jupyter'
 'Hero Honda Passion Pro' 'Hero Splender Plus' 'Honda CB Shine'
 'Bajaj Discover 100' 'Suzuki Access 125' 'TVS Wego' 'Honda CB twister'
 'Hero Glamour' 'Hero Super Splendor' 'Bajaj Discover 125' 'Hero Hunk'
 'Hero  Ignitor Disc' 'Hero  CBZ Xtreme' 'Bajaj  ct 100' 'i20' 'grand i10'
 'i10' 'eon' 'xcent' 'elantra' 'creta' 'verna' 'city' 'brio' 'amaze'
 'jazz']
```

In [ ]:

```
print(count)
```

```
city                       26
corolla altis              16
verna                      14
fortuner                   11
brio                       10
                           ..
Honda CB Trigger            1
Yamaha FZ S                 1
Bajaj Pulsar 135 LS         1
Activa 4g                   1
Bajaj Avenger Street 220    1
Name: Car_Name, Length: 98, dtype: int64
```

In [ ]:

```
car_cat = pd.get_dummies(car_data['Car_Name'].replace(repl,"Uncommmon"))
```

In [ ]:

```
car_data['Transmission'].unique()
```

Out[ ]:

```
array(['Manual', 'Automatic'], dtype=object)
```

In [ ]:

```
car_data['Owner'].unique()
```

Out[ ]:

```
array([0, 1, 3], dtype=int64)
```

In [ ]:

```
car_data['Seller_Type'].unique()
```

Out[ ]:

```
array(['Dealer', 'Individual'], dtype=object)
```

In [ ]:

```
for i in range(301):
    car_data.loc[i,['Year']] = 2023 - car_data.loc[i,['Year']]

car_data
```

Out[ ]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transm
0	ritz	9	3.35	5.59	27000	Petrol	Dealer	M
1	sx4	10	4.75	9.54	43000	Diesel	Dealer	M
2	ciaz	6	7.25	9.85	6900	Petrol	Dealer	M
3	wagon r	12	2.85	4.15	5200	Petrol	Dealer	M
4	swift	9	4.60	6.87	42450	Diesel	Dealer	M
...	...	...	...	...	...	...	...	
296	city	7	9.50	11.60	33988	Diesel	Dealer	M
297	brio	8	4.00	5.90	60000	Petrol	Dealer	M
298	city	14	3.35	11.00	87934	Petrol	Dealer	M
299	city	6	11.50	12.50	9000	Diesel	Dealer	M
300	brio	7	5.30	5.90	5464	Petrol	Dealer	M

301 rows × 9 columns

In [ ]:

```
car_data.isnull().sum()
```

Out[ ]:

```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

In [ ]:

```
type(car_data)
```

Out[ ]:

```
pandas.core.frame.DataFrame
```

In [ ]:

```
car_data['Depreciation'] = (car_data['Present_Price'] - car_data['Selling_Price'
# car_data['Depreciation_per_km'] = (car_data['Present_Price'] - car_data['Selli
car_data
```

Out[ ]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transm
0	ritz	9	3.35	5.59	27000	Petrol	Dealer	M
1	sx4	10	4.75	9.54	43000	Diesel	Dealer	M
2	ciaz	6	7.25	9.85	6900	Petrol	Dealer	M
3	wagon r	12	2.85	4.15	5200	Petrol	Dealer	M
4	swift	9	4.60	6.87	42450	Diesel	Dealer	M
...	...	...	...	...	...	...	...	
296	city	7	9.50	11.60	33988	Diesel	Dealer	M
297	brio	8	4.00	5.90	60000	Petrol	Dealer	M
298	city	14	3.35	11.00	87934	Petrol	Dealer	M
299	city	6	11.50	12.50	9000	Diesel	Dealer	M
300	brio	7	5.30	5.90	5464	Petrol	Dealer	M

301 rows × 10 columns

In [ ]:

```
car_data = pd.concat([car_data,car_cat], axis = 1)
car_data = pd.get_dummies(car_data, columns = ['Transmission',"Fuel_Type","Selle
car_data.drop(columns="Car_Name", inplace=True)
```

In [ ]:

```
type(car_data)
```

Out[ ]:

```
pandas.core.frame.DataFrame
```

In [ ]:

```
# from sklearn.compose import ColumnTransformer
# from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder
print(car_data)
```

```
     Year  Selling_Price  Present_Price  Kms_Driven  Owner  Depreciation  \
0       9           3.35           5.59       27000      0      0.248889
1      10           4.75           9.54       43000      0      0.479000
2       6           7.25           9.85        6900      0      0.433333
3      12           2.85           4.15        5200      0      0.108333
4       9           4.60           6.87       42450      0      0.252222
..    ...            ...            ...         ...    ...           ...
296     7           9.50          11.60       33988      0      0.300000
297     8           4.00           5.90       60000      0      0.237500
298    14           3.35          11.00       87934      0      0.546429
299     6          11.50          12.50        9000      0      0.166667
300     7           5.30           5.90        5464      0      0.085714

     Uncommmon  brio  ciaz  city  corolla altis  fortuner  grand i10  i20  \
0            1     0     0     0              0         0          0    0
1            1     0     0     0              0         0          0    0
2            0     0     1     0              0         0          0    0
3            1     0     0     0              0         0          0    0
4            1     0     0     0              0         0          0    0
..         ...   ...   ...   ...            ...       ...        ...  ...
296          0     0     0     1              0         0          0    0
297          0     1     0     0              0         0          0    0
298          0     0     0     1              0         0          0    0
299          0     0     0     1              0         0          0    0
300          0     1     0     0              0         0          0    0

     innova  verna  Transmission_Manual  Fuel_Type_Diesel  Fuel_Type_Petrol  \
0         0      0                    1                 0                 1
1         0      0                    1                 1                 0
2         0      0                    1                 0                 1
3         0      0                    1                 0                 1
4         0      0                    1                 1                 0
..      ...    ...                  ...               ...               ...
296       0      0                    1                 1                 0
297       0      0                    1                 0                 1
298       0      0                    1                 0                 1
299       0      0                    1                 1                 0
300       0      0                    1                 0                 1

     Seller_Type_Individual
0                         0
1                         0
2                         0
3                         0
4                         0
..                      ...
296                       0
297                       0
298                       0
299                       0
300                       0

[301 rows x 20 columns]
```

In [ ]:

```
correlation_matrix = car_data.corr()
correlation_matrix['Selling_Price']
```

Out[ ]:

```
Year                     -0.236141
Selling_Price             1.000000
Present_Price             0.878983
Kms_Driven                0.029187
Owner                    -0.088344
Depreciation              0.656466
Uncommmon                -0.526853
brio                      0.003058
ciaz                      0.097252
city                      0.167118
corolla altis             0.117753
fortuner                  0.538261
grand i10                 0.009198
i20                       0.003646
innova                    0.280812
verna                     0.062962
Transmission_Manual      -0.367128
Fuel_Type_Diesel          0.552339
Fuel_Type_Petrol         -0.540571
Seller_Type_Individual   -0.550724
Name: Selling_Price, dtype: float64
```

In [ ]:

```
# OneHotEncoder = OneHotEncoder()
```

In [ ]:

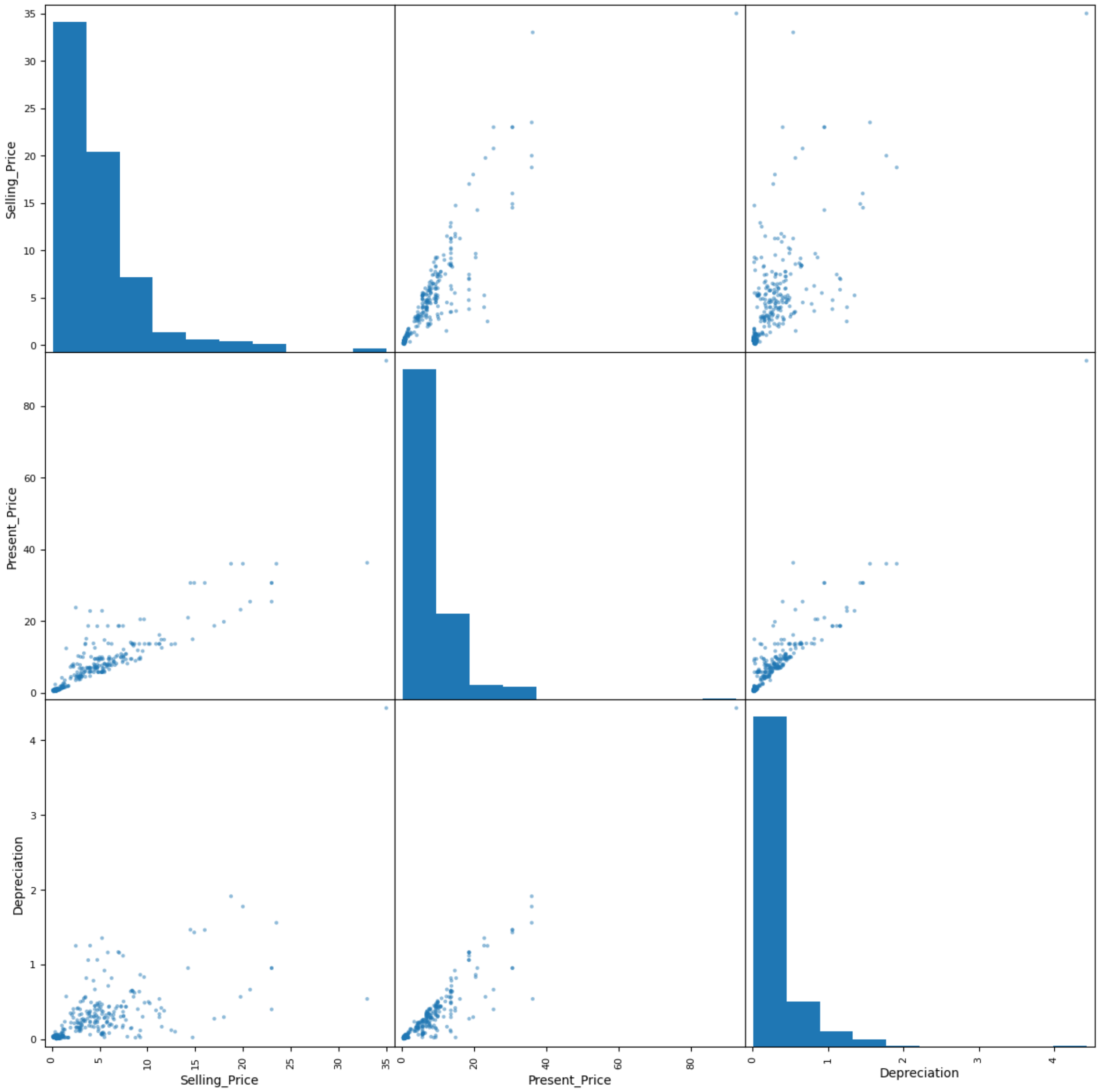
```
# transformer = ColumnTransformer(transformers=[('tnf1', OneHotEncoder(sparse=Fa
#                             remainder='passthrough')
# car_data = transformer.fit_transform(car_data).shape
```

In [ ]:

```
attributes = ['Selling_Price','Present_Price','Depreciation']
scatter_matrix(car_data[attributes],figsize=(15,15))
```

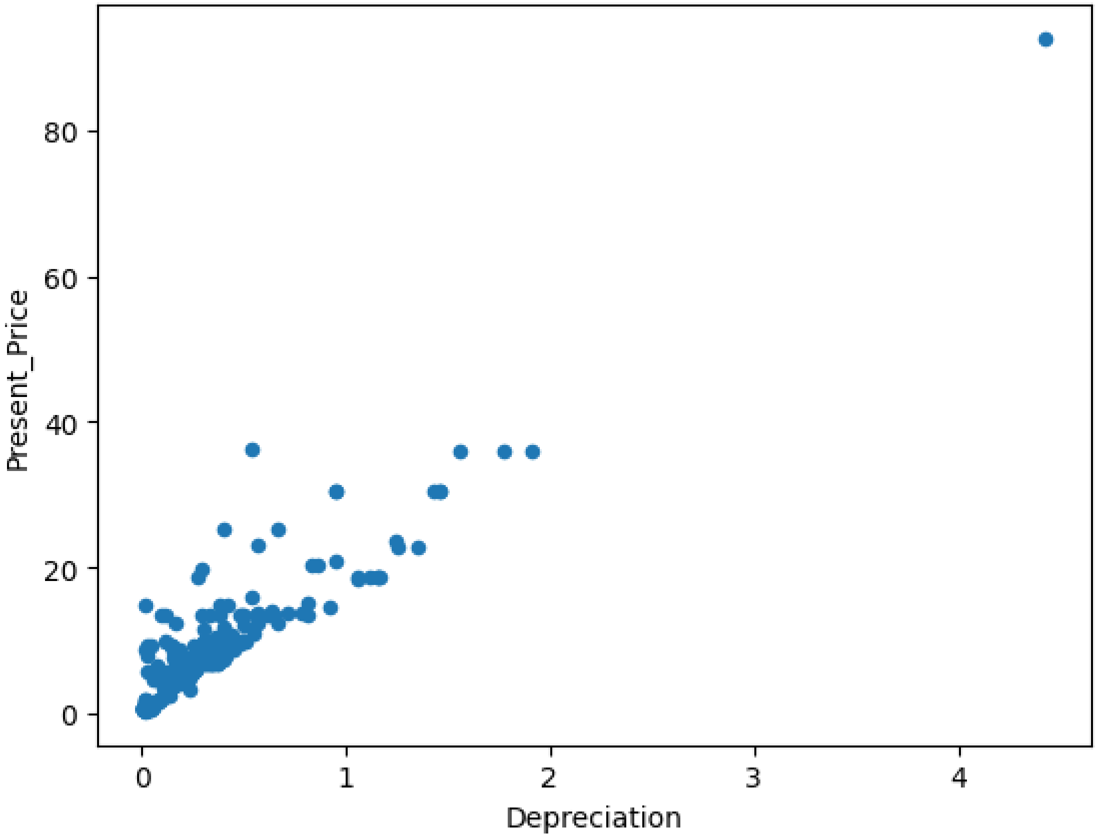
Out[ ]:

```
array([[<AxesSubplot:xlabel='Selling_Price', ylabel='Selling_Price'>,
        <AxesSubplot:xlabel='Present_Price', ylabel='Selling_Price'>,
        <AxesSubplot:xlabel='Depreciation', ylabel='Selling_Price'>],
       [<AxesSubplot:xlabel='Selling_Price', ylabel='Present_Price'>,
        <AxesSubplot:xlabel='Present_Price', ylabel='Present_Price'>,
        <AxesSubplot:xlabel='Depreciation', ylabel='Present_Price'>],
       [<AxesSubplot:xlabel='Selling_Price', ylabel='Depreciation'>,
        <AxesSubplot:xlabel='Present_Price', ylabel='Depreciation'>,
        <AxesSubplot:xlabel='Depreciation', ylabel='Depreciation'>]],
      dtype=object)
```



```
In [ ]: car_data.plot(kind='scatter',x='Depreciation',y = 'Present_Price')
```

```
Out[ ]: <AxesSubplot:xlabel='Depreciation', ylabel='Present_Price'>
```



In [ ]:

```
car_data
```

Out[ ]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner	Depreciation	Uncommmon	brio
0	9	3.35	5.59	27000	0	0.248889	1	0
1	10	4.75	9.54	43000	0	0.479000	1	0
2	6	7.25	9.85	6900	0	0.433333	0	0
3	12	2.85	4.15	5200	0	0.108333	1	0
4	9	4.60	6.87	42450	0	0.252222	1	0
...	...	...	...	...	...	...	...	...
296	7	9.50	11.60	33988	0	0.300000	0	0
297	8	4.00	5.90	60000	0	0.237500	0	1
298	14	3.35	11.00	87934	0	0.546429	0	0
299	6	11.50	12.50	9000	0	0.166667	0	0
300	7	5.30	5.90	5464	0	0.085714	0	1

301 rows × 20 columns

In [ ]:

```
x = car_data.drop(columns=['Selling_Price'])
x
```

Out[ ]:

	Year	Present_Price	Kms_Driven	Owner	Depreciation	Uncommmon	brio	ciaz	city	co
0	9	5.59	27000	0	0.248889	1	0	0	0	
1	10	9.54	43000	0	0.479000	1	0	0	0	
2	6	9.85	6900	0	0.433333	0	0	1	0	
3	12	4.15	5200	0	0.108333	1	0	0	0	
4	9	6.87	42450	0	0.252222	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	
296	7	11.60	33988	0	0.300000	0	0	0	1	
297	8	5.90	60000	0	0.237500	0	1	0	0	
298	14	11.00	87934	0	0.546429	0	0	0	1	
299	6	12.50	9000	0	0.166667	0	0	0	1	
300	7	5.90	5464	0	0.085714	0	1	0	0	

301 rows × 19 columns

In [ ]:

```
y = car_data.loc[:,['Selling_Price']]
y
```

Out[ ]:

	Selling_Price
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

301 rows × 1 columns

In [ ]:

```
# car_name = car_data['Car_Name']
# car_name.shape
```

In [ ]:

```
# vectorizer = TfidfVectorizer()
# vectorizer.fit(x['Car_Name'])
# car_name = vectorizer.transform(x['Car_Name'].values)
# print(car_name)
```

In [ ]:

```
x_train, x_test, y_train , y_test = train_test_split(x,y,test_size=0.3,random_st
x_train.reset_index(drop = True, inplace = True)
x_test.reset_index(drop = True, inplace = True)
y_train.reset_index(drop = True, inplace = True)
y_test.reset_index(drop = True, inplace = True)
```

In [ ]:

```
x_train
```

Out[ ]:

	Year	Present_Price	Kms_Driven	Owner	Depreciation	Uncommmon	brio	ciaz	city	co
0	15	0.58	1900	0	0.022000	1	0	0	0	
1	10	18.61	56001	0	1.116000	0	0	0	0	
2	7	10.79	43000	0	0.434286	1	0	0	0	
3	6	3.60	2135	0	0.125000	1	0	0	0	
4	15	0.52	500000	0	0.023333	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	
205	10	0.57	18000	0	0.032000	1	0	0	0	
206	12	12.48	45000	0	0.665000	0	0	0	0	
207	9	3.45	16500	1	0.233333	1	0	0	0	
208	12	10.00	69341	0	0.491667	0	0	0	1	
209	6	1.78	4000	0	0.021667	1	0	0	0	

210 rows × 19 columns

In [ ]:

```
model = LinearRegression()
model.fit(x_train,y_train)
y_predicted = model.predict(x_test)
```

In [ ]:

```
mse_train = mean_squared_error(y_train, model.predict(x_train))
mse_train
```

Out[ ]:

```
0.6585354385292549
```

In [ ]:

```
mse = mean_squared_error(y_test,y_predicted)
mse
```

Out[ ]:

```
0.9485598085321431
```

In [ ]:

```
rmse = math.sqrt(mse)
rmse
```

Out[ ]:

```
0.9739403516294738
```

In [ ]:

```
mae = mean_absolute_error(y_test, y_predicted)
mae
```

Out[ ]:

```
0.6637647509784605
```

In [ ]:

```
mae_train = mean_absolute_error(y_train, model.predict(x_train))
mae_train
```

Out[ ]:

```
0.5587951609975123
```

In [ ]:

```
score = model.score(x_test,y_test)
score
```

Out[ ]:

```
0.9666870433858504
```

In [ ]:

```
# calculating score manually
u = ((y_test - y_predicted)**2).sum()
v = ((y-y_test.mean())**2).sum()

score = 1-(u/v)
print(score)
```

```
Selling_Price    0.988881
dtype: float64
```

In [ ]:

```
score_train = model.score(x_train,y_train)
score_train
```

Out[ ]:

```
0.9731676974852368
```