multilinearregression

August 24, 2023

1 Multiple Linear Regression

- Linear Regression is used for predicting the relationship between the independent and dependent variable. It aims to predict the good fit line that describes the relationship by minimizing the mean_squared_error of the predicted value and the actual value.
- In multiple linear regression, we use multiple/more than one independent variable/feature to predict the target variable.
- Slope-intercept formulae, Y=B0+B1X1=B2X2+……+BiXi

  where, y= target variable(Dependent varible) B0= intercept B1=slope/coefficient X1,X2,X3….Xi=Predictive variables(Independent varible)
- Linear regression aims to find the slope value and intercepts value(in case of multiple linear model-B1,B2,B3….) to predict the target feature.
- Have used the Car Price Estimation data set from kaggle.com; where, Independent variable(X)- ‘car_name’,‘fuel_type’,‘no_cylinder’,‘seating_capacity’,‘transmission_type’,‘body_type’,‘start
  and
  target variable(Y)- “ending_price”

[1]:
```python
# Importing necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```python
# Onboarding data onto colab

from google.colab import files
rawdata=files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving cars.csv to cars.csv
```

[3]:
```python
# Converting to dataframe

df=pd.read_csv('cars.csv')
```

```
df
```

[3]:

```
                      car_name  reviews_count fuel_type  \
0               Maruti Alto K10             51    Petrol
1                Maruti Brezza              86    Petrol
2                Mahindra Thar             242    Diesel
3              Mahindra XUV700             313    Diesel
4           Mahindra Scorpio-N             107    Diesel
..                         …              …       …
198     Mercedes-Benz AMG A 45 S             35    Petrol
199  BMW 3 Series Gran Limousine              3    Petrol
200               MG Hector Plus              2    Diesel
201                  Audi RS Q8              9    Petrol
202           Maruti Alto 800 tour              4    Petrol

     engine_displacement  no_cylinder  seating_capacity transmission_type  \
0                    998            3               5.0         Automatic
1                   1462            4               5.0         Automatic
2                   2184            4               4.0         Automatic
3                   2198            4               7.0         Automatic
4                   2198            4               7.0         Automatic
..                   …          …               …               …
198                 1991            4               5.0         Automatic
199                 1998            4               5.0         Automatic
200                 1956            4               7.0            Manual
201                 3998            8               5.0         Automatic
202                  796            3               5.0            Manual

     fuel_tank_capacity  body_type  rating  starting_price  ending_price  \
0                  27.0  Hatchback     4.5          399000        583000
1                  48.0        SUV     4.5          799000       1396000
2                  57.0        SUV     4.5         1353000       1603000
3                  60.0        SUV     4.5         1318000       2458000
4                  57.0        SUV     4.5         1199000       2390000
..                  …        …     …             …           …
198                 0.0  Hatchback     4.5          659000        999000
199                59.0      Sedan     4.5         1041000       1041000
200                60.0        SUV     4.5         1615000       2075000
201                85.0        SUV     3.5        21700000      21700000
202                35.0  Hatchback     4.5          391000        397000

     max_torque_nm  max_torque_rpm  max_power_bhp  max_power_rp
0             89.0            3500          65.71          5500
1            136.8            4400         101.65          6000
2            300.0            2800         130.00          3750
3            450.0            2800         182.38          3500
4            400.0            2750         172.45          3500
```

```
..           …           …          …         …
198        500.0        5250      415.71       6750
199        400.0        4400      254.79       5000
200        350.0        2500      167.67       3750
201        800.0        4500      591.39       6000
202         69.0        3500       47.33       6000

[203 rows x 16 columns]
```

[4]:

```
# Shallow copy

df_copy=df.copy()
```

2 Exploratory Data Analysis

[5]:

```
df.head()
```

[5]:

```
             car_name  reviews_count fuel_type  engine_displacement  \
0     Maruti Alto K10             51    Petrol                  998
1       Maruti Brezza             86    Petrol                 1462
2       Mahindra Thar            242    Diesel                 2184
3     Mahindra XUV700            313    Diesel                 2198
4  Mahindra Scorpio-N            107    Diesel                 2198

   no_cylinder  seating_capacity transmission_type  fuel_tank_capacity  \
0            3               5.0         Automatic                27.0
1            4               5.0         Automatic                48.0
2            4               4.0         Automatic                57.0
3            4               7.0         Automatic                60.0
4            4               7.0         Automatic                57.0

   body_type  rating  starting_price  ending_price  max_torque_nm  \
0  Hatchback     4.5          399000        583000           89.0
1        SUV     4.5          799000       1396000          136.8
2        SUV     4.5         1353000       1603000          300.0
3        SUV     4.5         1318000       2458000          450.0
4        SUV     4.5         1199000       2390000          400.0

   max_torque_rpm  max_power_bhp  max_power_rp
0            3500          65.71          5500
1            4400         101.65          6000
2            2800         130.00          3750
3            2800         182.38          3500
4            2750         172.45          3500
```

[6]:

```
df.columns
```

[6]:
```
Index(['car_name', 'reviews_count', 'fuel_type', 'engine_displacement',
       'no_cylinder', 'seating_capacity', 'transmission_type',
       'fuel_tank_capacity', 'body_type', 'rating', 'starting_price',
       'ending_price', 'max_torque_nm', 'max_torque_rpm', 'max_power_bhp',
       'max_power_rp'],
      dtype='object')
```

[7]:
```
# Feature Engineering

df=df.
↪drop(['engine_displacement','fuel_tank_capacity','rating','max_torque_nm',␣
↪'max_torque_rpm',␣
↪'max_power_bhp','reviews_count','max_power_rp'],axis='columns')
```

[8]:
```
df
```

[8]:
```
                       car_name fuel_type  no_cylinder  seating_capacity  \
0                Maruti Alto K10    Petrol            3               5.0
1                  Maruti Brezza    Petrol            4               5.0
2                  Mahindra Thar    Diesel            4               4.0
3                Mahindra XUV700    Diesel            4               7.0
4             Mahindra Scorpio-N    Diesel            4               7.0
..                           …         …            …                 …
198      Mercedes-Benz AMG A 45 S    Petrol            4               5.0
199   BMW 3 Series Gran Limousine    Petrol            4               5.0
200                MG Hector Plus    Diesel            4               7.0
201                    Audi RS Q8    Petrol            8               5.0
202           Maruti Alto 800 tour    Petrol            3               5.0

    transmission_type  body_type  starting_price  ending_price
0           Automatic  Hatchback          399000        583000
1           Automatic        SUV          799000       1396000
2           Automatic        SUV         1353000       1603000
3           Automatic        SUV         1318000       2458000
4           Automatic        SUV         1199000       2390000
..                …          …               …             …
198         Automatic  Hatchback          659000        999000
199         Automatic      Sedan         1041000       1041000
200            Manual        SUV         1615000       2075000
201         Automatic        SUV        21700000      21700000
202            Manual  Hatchback          391000        397000

[203 rows x 8 columns]
```

[9]:
```
# Technical enquiry

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 203 entries, 0 to 202
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   car_name           203 non-null    object
 1   fuel_type          203 non-null    object
 2   no_cylinder        203 non-null    int64
 3   seating_capacity   202 non-null    float64
 4   transmission_type  203 non-null    object
 5   body_type          203 non-null    object
 6   starting_price     203 non-null    int64
 7   ending_price       203 non-null    int64
dtypes: float64(1), int64(3), object(4)
memory usage: 12.8+ KB
```

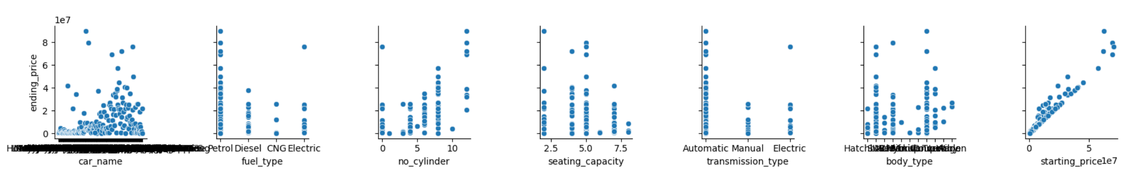
[10]:
```python
# Statistical enquiry

df.describe()
```

[10]:

	no_cylinder	seating_capacity	starting_price	ending_price
count	203.000000	202.000000	2.030000e+02	2.030000e+02
mean	4.709360	5.014851	9.443640e+06	1.112005e+07
std	2.538664	1.161050	1.357035e+07	1.551746e+07
min	0.000000	2.000000	3.390000e+05	3.610000e+05
25%	4.000000	5.000000	9.455000e+05	1.407500e+06
50%	4.000000	5.000000	4.312000e+06	4.600000e+06
75%	6.000000	5.000000	1.160000e+07	1.575000e+07
max	12.000000	8.000000	7.060000e+07	9.000000e+07

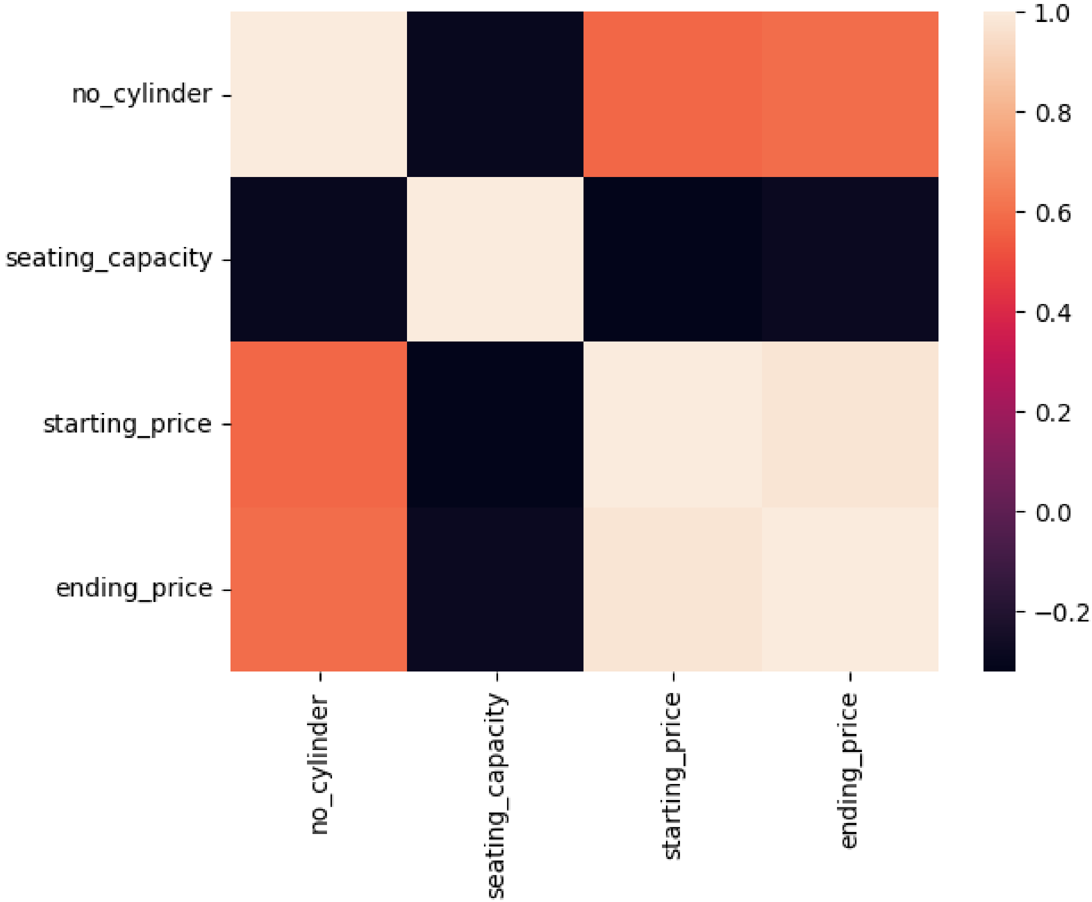
[11]:
```python
sns.
↪pairplot(df,x_vars=['car_name','fuel_type','no_cylinder','seating_capacity','transmission_t
plt.show()
```



[12]:
```python
sns.heatmap(df.corr())
plt.show()
```

```
<ipython-input-12-88edb43bf50b>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
```

```
to silence this warning.
  sns.heatmap(df.corr())
```



[13]:
```python
# Finding Nan values

df.isnull().sum()
```

[13]:
```
car_name             0
fuel_type            0
no_cylinder          0
seating_capacity     1
transmission_type    0
body_type            0
starting_price       0
ending_price         0
dtype: int64
```

[14]:
```python
df.dropna(inplace=True)
```

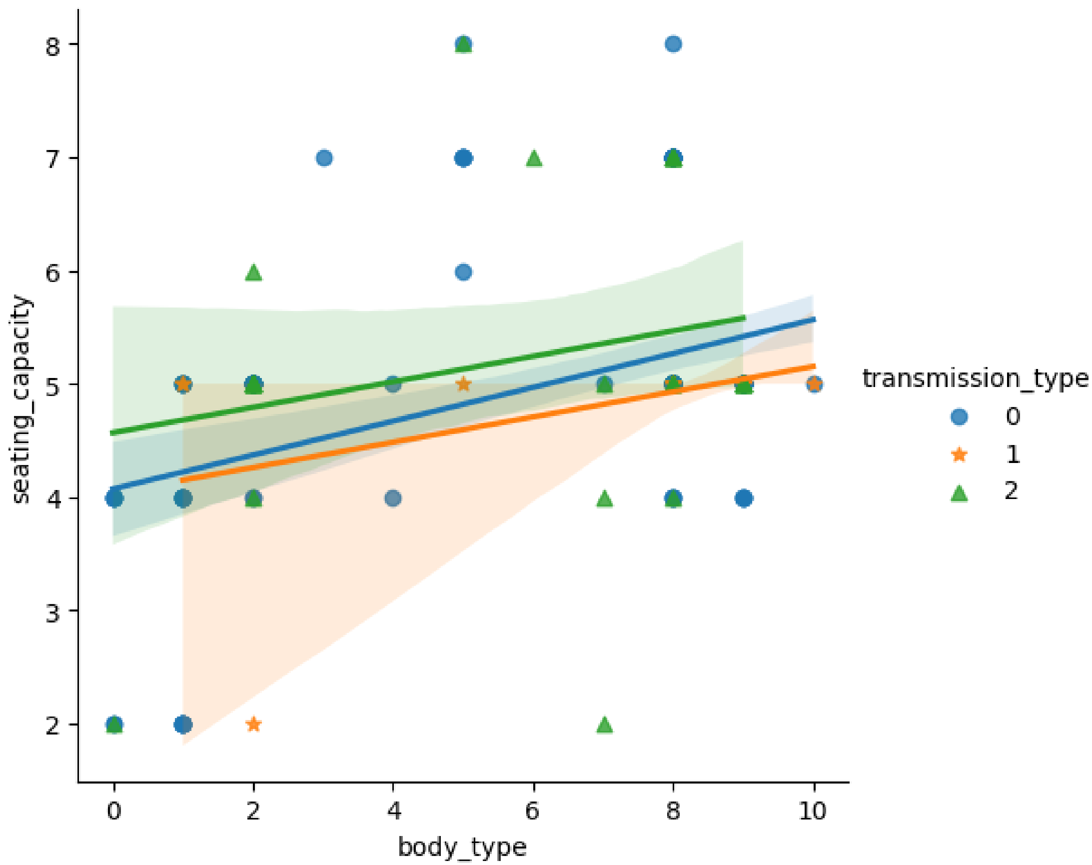
```
[16]: # Finding duplicates if any,

      df.duplicated().sum()
```

```
[16]: 0
```

3 Multivariate Analysis

```
[45]: sns.
       ↪lmplot(x='body_type',y='seating_capacity',data=df,hue='transmission_type',markers=['o','*',
      plt.show()
```



4 Data encoding

```
[17]: from sklearn.preprocessing import LabelEncoder

      le=LabelEncoder()
```

```
df['car_name']=le.fit_transform(df['car_name'])
df['fuel_type']=le.fit_transform(df['fuel_type'])
df['transmission_type']=le.fit_transform(df['transmission_type'])
df['body_type']=le.fit_transform(df['body_type'])
```

[18]:
```
df
```

[18]:
```
     car_name  fuel_type  no_cylinder  seating_capacity  transmission_type  \
0         106          3            3               5.0                  0
1         108          3            4               5.0                  0
2         101          1            4               4.0                  0
3         103          1            4               7.0                  0
4         100          1            4               7.0                  0
..        ...        ...          ...               ...                ...
198       127          3            4               5.0                  0
199        12          3            4               5.0                  0
200        89          1            4               7.0                  2
201         7          3            8               5.0                  0
202       105          3            3               5.0                  2

     body_type  starting_price  ending_price
0            2          399000        583000
1            8          799000       1396000
2            8         1353000       1603000
3            8         1318000       2458000
4            8         1199000       2390000
..         ...             ...           ...
198          2          659000        999000
199          9         1041000       1041000
200          8         1615000       2075000
201          8        21700000      21700000
202          2          391000        397000

[202 rows x 8 columns]
```

[19]:
```
X=df.iloc[:,:-1]
```

[20]:
```
Y=df.iloc[:,-1]
```

5 Normalization

[21]:
```
from sklearn.preprocessing import StandardScaler

sc=StandardScaler()
X_sc=sc.fit_transform(X)
```

6 Train test split

[24]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(X_sc,Y,test_size=0.
 ↪3,random_state=51)
```

7 Model Training

[25]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

[25]:
```
LinearRegression()
```

[26]:
```
# Predicting the output

y_predict=lr.predict(x_test)
```

[27]:
```
y_predict
```

[27]:
```
array([ 5450734.87876823, 28883042.20779844,  4830941.34009862,
        1087634.03203458, 28701197.29354954,  1499695.85680375,
        2310424.10412433, 14674775.70844833, 17934581.96638384,
       27757791.07494777, 10128661.25953963, 38009168.96838657,
         831654.32959579,   685638.64178005,   486334.52338482,
        2713701.43982593,  1324169.33806021,   588217.97080897,
        3214435.02888541,  -279229.20308003,  4786805.61487156,
         914268.63947213, 18452982.23247068, 24314601.25868671,
       21267974.07534245, 75207031.85186312,  9396687.67137627,
       15048608.02795183, 10202771.12999566,  1890689.88796972,
        2257427.57043463, 11301089.0161855 ,  9011895.71803283,
        5572125.85625006, 16294798.48283658, 15184742.40018022,
        1103229.21737047,  9478188.42698938,  6571878.88623431,
       19450456.39461272, 18541788.49635421,  1655876.11357459,
        7897797.73433908,  3540427.63245242,  5159408.95108046,
        4566098.34026237,  2279503.69999649,  8936771.034402  ,
        4006078.11742655,  1751211.43100507,  1673271.3206316 ,
       67488597.67475884, 15746785.31150446,  1774231.44549256,
        4283497.83869911,   973177.57010573,  1315179.70276314,
        1070705.19787242,  2778785.12087851,  1861267.76001905,
       21042372.77040582])
```

[28]:
```
y_test
```

[28]:
```
68       4435000
121     27100000
90       3188000
7         949000
120     25500000
          …
19        918000
32       1079000
27       1549000
51       1949000
124     22200000
Name: ending_price, Length: 61, dtype: int64
```

8 Model Evaluation

[29]:
```python
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score

MAE=mean_squared_error(y_test,y_predict)
MSE=mean_absolute_error(y_test,y_predict)
RMSE=np.sqrt(MSE)
print("MAE",MAE)
print("MSE",MSE)
print("RMSE",RMSE)
```

```
MAE 20002026248420.62
MSE 2237738.5409558667
RMSE 1495.907263487903
```

[30]:
```python
R2=r2_score(y_test,y_predict)
print("R2",R2)
```

```
R2 0.9276182330996255
```

R2 score reveals that the model accuracy is 0.9 (i.e) 90% of the data fits in the good fit line/regression line.