**import** pandas **as** pd

2

**import** seaborn **as** sns

3

**import** matplotlib.pyplot **as** plt

In [2]:

1

file **=** pd.read\_excel("D:/New Microsoft Excel Worksheet.xlsx",parse\_dates**=**[['Mfg\_Month','Mfg\_Year']])

2

​

In [3]:

1

file.head(2)

. . .

**Copy the main data**

In [4]:

1

df1 **=** file.copy()

**Columns**

In [5]:

1

df1.columns

. . .

**Data info**

In [6]:

1

df1.info()

. . .

**Find out Null values**

In [7]:

1

df1.isnull().sum()

. . .

No null value present

**Stats of data**

In [8]:

1

df1.describe(include**=**'all')

. . .

**Highest price orineted model**

In [9]:

1

df1.groupby('Model')['Price'].max().sort\_values(ascending**=False**).head(1)

Out[9]:

Model

 TOYOTA Corolla VERSO 2.0 D4D SOL (7) BNS MPV 32500

Name: Price, dtype: int64

**Lowest price oriented model**

In [10]:

1

df1.groupby('Model')['Price'].min().sort\_values(ascending**=True**).head(1)

Out[10]:

Model

TOYOTA Corolla 1.8D Stationwagen 4350

Name: Price, dtype: int64

**Average price of models**

In [442]:

1

avr**=**df1.groupby('Model')['Price'].mean().head(3)

2

avr

Out[442]:

Model

TOYOTA Corolla 9390.0

TOYOTA Corolla ! 1.6-16v vvt-i sol airco sedan 4/5-Doors 9940.0

TOYOTA Corolla 1 6-16v VVT-i Linea Terra Comfort Airco 5drs 4/5-Doors 15950.0

Name: Price, dtype: float64

**Number of models**

In [12]:

1

df1['Model'].nunique()

Out[12]:

372

**Top 5 maximum KM run by model**

In [13]:

1

df1.groupby('Model')['KM'].mean().sort\_values(ascending**=False**).head(5)

Out[13]:

Model

TOYOTA Corolla 2.0 DSL WAGON LINEA TERRA COMM Anders 232940.0

 TOYOTA Corolla 1.9 D HATCHB SOL 2/3-Doors 216000.0

TOYOTA Corolla 1.6 LB \*G3\* AIRCO 4/5-Doors 207114.0

TOYOTA Corolla 2.0D LINEA TERRA+airco 2/3-Doors 203254.0

TOYOTA Corolla 1.9 D SEDAN TERRA 4/5-Doors 198167.0

Name: KM, dtype: float64

**Least 5 minimum KM run by model**

In [14]:

1

df1.groupby('Model')['KM'].min().sort\_values(ascending**=True**).head(5)

Out[14]:

Model

 TOYOTA Corolla VERSO 2.0 D4D SOL (7) BNS MPV 1

TOYOTA Corolla 1.6-16v VVT-i Linea Terra Comfort AIRCO NIEUW 5DRS 4/5-Doors 1

TOYOTA Corolla 1.6-16v VVT-i Linea Terra Comfort NIEUW AIRCO 5drs 4/5-Doors 1

TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-Doors 1

TOYOTA Corolla 2.0 d HB Diesel 2/3-Doors 1

Name: KM, dtype: int64

**Number of sports car**

In [15]:

1

spr**=**len(df1[df1['Sport\_Model']**==**1])

2

print('Number of Sports car=',spr)

Number of Sports car= 431

**Number of car has Automatic feture**

In [16]:

1

auto**=**len(df1[df1['Automatic']**==**1])

2

print('Number of car has Automatic feture =', auto)

Number of car has Automatic feture = 80

**Highest 'CC' based model**

In [17]:

1

df1.sort\_values(by**=**'cc', ascending**=False**).head(1)[['Model','cc']]

Out[17]:

|  | **Model** | **cc** |
| --- | --- | --- |
| **80** | TOYOTA Corolla 1.6 5drs 1 4/5-Doors | 16000 |

**Top 5 highest weight oriented model**

In [18]:

1

df1.sort\_values(by**=**'Weight', ascending**=False**).head(5)[['Model','Weight']]

Out[18]:

|  | **Model** | **Weight** |
| --- | --- | --- |
| **221** | TOYOTA Corolla 1.6 HB LINEA SOL 4/5-Doors | 1615 |
| **110** | TOYOTA Corolla VERSO 2.0 D4D SOL (7) MPV | 1480 |
| **960** | TOYOTA Corolla | 1480 |
| **109** | TOYOTA Corolla VERSO 2.0 D4D SOL (7) BNS MPV | 1480 |
| **111** | TOYOTA Corolla VERSO 2.0 D4D SOL (7) MPV | 1480 |

**Relation between Weight and Quarterly\_Tax**

In [19]:

1

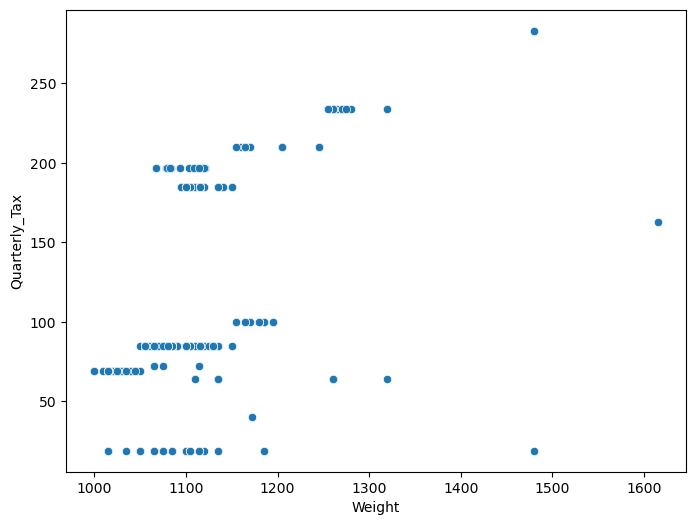
plt.figure(figsize**=**(8,6))

2

sns.scatterplot(x**=**'Weight', y**=**'Quarterly\_Tax', data**=**df1)

3

plt.show()



In [20]:

1

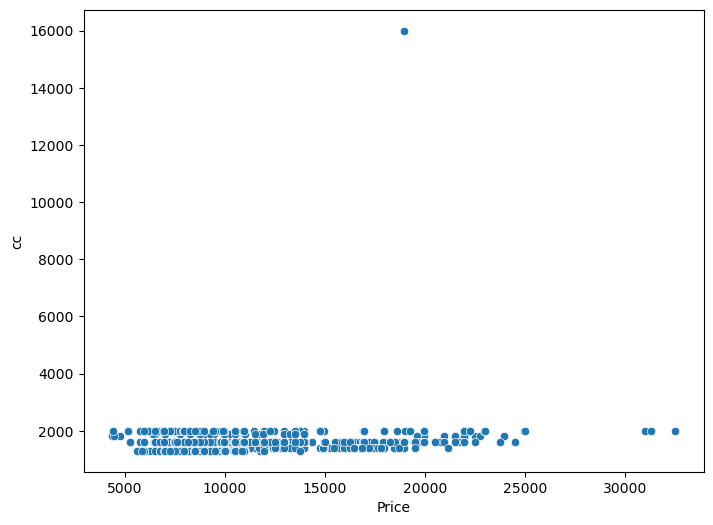
plt.figure(figsize**=**(8,6))

2

sns.scatterplot(x**=**'Price', y**=**'cc', data**=**df1)

3

plt.show()



In [25]:

1

df1.columns

Out[25]:

Index(['Mfg\_Month\_Mfg\_Year', 'Id', 'Model', 'Price', 'Age\_08\_04', 'KM',

'Fuel\_Type', 'HP', 'Met\_Color', 'Color', 'Automatic', 'cc', 'Doors',

'Cylinders', 'Gears', 'Quarterly\_Tax', 'Weight', 'Mfr\_Guarantee',

'BOVAG\_Guarantee', 'Guarantee\_Period', 'ABS', 'Airbag\_1', 'Airbag\_2',

'Airco', 'Automatic\_airco', 'Boardcomputer', 'CD\_Player',

'Central\_Lock', 'Powered\_Windows', 'Power\_Steering', 'Radio',

'Mistlamps', 'Sport\_Model', 'Backseat\_Divider', 'Metallic\_Rim',

'Radio\_cassette', 'Tow\_Bar'],

dtype='object')

**Year wise model's fuel type orineted price analysis**

In [241]:

1

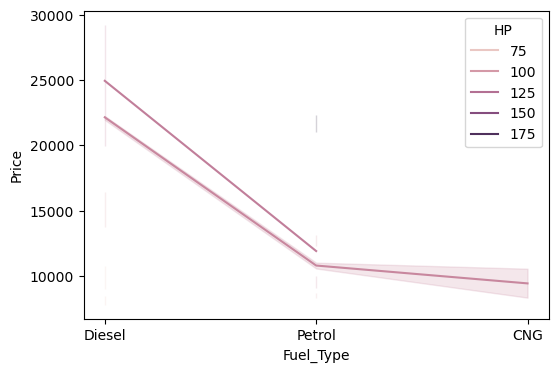
plt.figure(figsize**=**(6,4))

2

sns.lineplot(x**=**'Fuel\_Type', y**=**'Price', hue**=**'HP', data**=**df1)

3

plt.show()



**Constant growth in model's price and HP has been identified as one of element that impact on model price**

1. Diesel model price is constant higher than pertol and cng

**Year wise model's fuel type orineted price analysis**

In [242]:

1

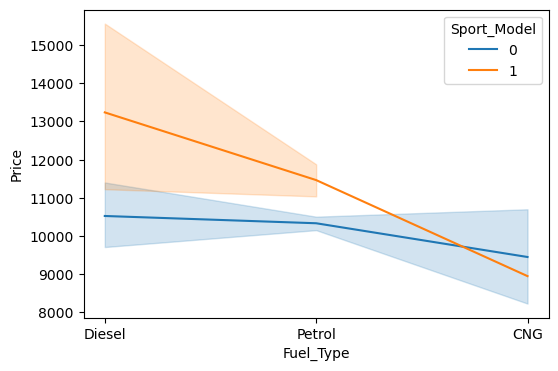
plt.figure(figsize**=**(6,4))

2

sns.lineplot(x**=**'Fuel\_Type', y**=**'Price', hue**=**'Sport\_Model', data**=**df1)

3

plt.show()



**Constant growth in sports model's price and fuel\_type has been identified as one of element that impact on model price**

1. Diesel model price of sports models is higher than pertol and cng

**Model- 1**

In [28]:

1

df1.columns

. . .

In [33]:

1

df1['Fuel\_Type'].unique()

Out[33]:

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

In [35]:

1

df1['Color'].unique()

Out[35]:

array(['Blue', 'Silver', 'Black', 'White', 'Grey', 'Red', 'Green',

'Yellow', 'Violet', 'Beige'], dtype=object)

**Encode Fuel Type**

In [36]:

1

dummy **=** pd.get\_dummies(df1['Fuel\_Type']).astype('int')

2

dummy.head(2)

Out[36]:

|  | **CNG** | **Diesel** | **Petrol** |
| --- | --- | --- | --- |
| **0** | 0 | 1 | 0 |
| **1** | 0 | 1 | 0 |

In [314]:

1

mod **=** pd.get\_dummies(df1['Model']).astype('int')

2

mod.head(2)

. . .

In [336]:

1

final\_data **=** pd.concat([df1,dummy], axis**=**1)

2

final\_data.head(2)

. . .

In [339]:

1

final\_data **=** pd.concat([final\_data,mod], axis**=**1)

2

final\_data.head(1)

. . .

In [334]:

1

final\_data.columns

. . .

In [347]:

1

X**=**final\_data[['HP','cc','Backseat\_Divider','Quarterly\_Tax','CNG', 'Diesel', 'Petrol', 'ABS', 'Airbag\_1', 'Airbag\_2', 'Airco', 'Automatic\_airco', 'Automatic', 'Central\_Lock']]

In [348]:

1

X.head(2)

. . .

In [349]:

1

y**=**final\_data['Price']

In [350]:

1

**from** sklearn.model\_selection **import** train\_test\_split

In [351]:

1

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y, train\_size**=**0.85,random\_state**=**5612)

In [352]:

1

final\_data.shape

Out[352]:

(1436, 1156)

In [353]:

1

X\_train.shape

Out[353]:

(1220, 14)

In [354]:

1

y\_train.shape

Out[354]:

(1220,)

**Model fitting**

In [355]:

1

**from** sklearn.linear\_model **import** LinearRegression

In [356]:

1

model**=** LinearRegression()

2

model\_train**=**model.fit(X\_train, y\_train)

In [357]:

1

pred1**=**model\_train.predict(X\_test)

In [358]:

1

pred1

. . .

In [359]:

1

p**=**pd.DataFrame(pred1, columns**=**['Pred\_price'])

In [360]:

1

p['Actual']**=**y\_test.values

**Score and error¶**

In [361]:

1

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

2

**from** math **import** sqrt

In [362]:

1

round(sqrt(mean\_squared\_error(p['Actual'], p['Pred\_price'])))

Out[362]:

2344

In [331]:

1

str**=**round(r2\_score(p['Actual'], p['Pred\_price'])**\***100)

2

print('Strangth of the model =',str )

Strangth of the model = 52

In [332]:

1

mean\_absolute\_error(p['Actual'], p['Pred\_price'])

Out[332]:

1764.4836907555664

**Save the model1**

In [433]:

1

**import** joblib

In [434]:

1

pred\_price **=** 'pred\_price.sav'

2

joblib.dump(model\_train,pred\_price)

Out[434]:

['pred\_price.sav']

In [439]:

1

loaded\_model\_1 **=** joblib.load(pred\_price)

2

print(loaded\_model)

LinearRegression()

**Model - 2**

In [363]:

1

df2**=**final\_data.copy()

In [364]:

1

df2.columns

. . .

In [418]:

1

X **=** df2[['Sport\_Model','Age\_08\_04','CNG', 'HP','Diesel', 'Petrol', 'Met\_Color','ABS', 'Airbag\_1', 'Airbag\_2','KM', 'Cylinders']]

In [419]:

1

X.head(2)

. . .

In [420]:

1

y**=**df2['Price']

In [421]:

1

**from** sklearn.model\_selection **import** train\_test\_split

In [422]:

1

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y, train\_size**=**0.8,random\_state**=**5612)

**Model Fitting**

In [423]:

1

**from** sklearn.linear\_model **import** LinearRegression

In [424]:

1

model**=** LinearRegression()

2

model\_train**=**model.fit(X\_train, y\_train)

In [425]:

1

pred**=**model\_train.predict(X\_test)

In [426]:

1

pred

. . .

In [427]:

1

sp **=** pd.DataFrame(pred, columns**=**['Pred\_pro\_sp'])

In [428]:

1

sp['Actual']**=**y\_test.values

**Score and error**

In [429]:

1

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

2

**from** math **import** sqrt

In [430]:

1

round(sqrt(mean\_squared\_error(sp['Actual'], sp['Pred\_pro\_sp'])))

Out[430]:

1413

In [431]:

1

str**=**round(r2\_score(sp['Actual'], sp['Pred\_pro\_sp'])**\***100)

2

print('Strangth of the model =',str )

Strangth of the model = 82

In [432]:

1

mean\_absolute\_error(sp['Actual'], sp['Pred\_pro\_sp'])

Out[432]:

1080.4605440738046

**Save model - 2**

In [ ]:

1

**import** joblib

In [437]:

1

pred\_price\_spr **=** 'joblib\_reg\_model\_pred\_price\_spr.sav'

2

joblib.dump(model\_train,pred\_price\_spr)

Out[437]:

['joblib\_reg\_model\_pred\_price\_spr.sav']

In [441]:

1

loaded\_model\_2 **=** joblib.load(pred\_price\_spr)

2

print(loaded\_model\_2)

LinearRegression()