SKILL ACTIVITY NO: 2

Name:Patil Shubham Ravasaheb Date:25/07/2021
PRN:
School: School of Data Science
Program: Machine Learning
Batch: ML 12
Module Name: Python Programming
Module Code: ML101

Title:Perform Regression on the Toyota Corolla Data Set.

Skills/Competencies to be acquired:

1. To gain an understanding of data and find clues from the data.
2. Assess assumptions on which statistical inference will be based.
3. To check the quality of data for further processing and cleaning if necessary.
4. To check for anomalies or outliers that may impact model.
5. Data Visualization.

Duration of activity: 1 Hour

1.What is the purpose of this activity?

Preview data.
Check total number of entries and column types.
Check any null values.
Apply multiple regression models on the data
Evaluate all the models performance

2.Steps performed in this activity.

1)EDA(Data Cleaning)
2)fitting regression models on the cleaned data
3)Evaluating the performance of all models

3.What resources / materials / equipment / tools did you use for this activity?

1)Jupyter notebook
2)python libraries
3)Google colab

4.What skills did you acquire?

1)Exploretory data analysis
2)Prediction using regression models
3)Finding best performing model

5.Time taken to complete the activity?

1) 1 Day

In [370]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [371]:
```
df =pd.read_csv('ToyotaCorolla.csv')
```

In [372]:
```
cols = ["Price","Age_08_04","KM","HP","cc","Doors","Gears","Quarterly_Tax","Weigh
```

In [373]:
```
df = df[cols]
```

In [374]:
```
df.head()
```

Out[374]:

	Price	Age_08_04	KM	HP	cc	Doors	Gears	Quarterly_Tax	Weight
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170

EDA

In [375]:
```
df.isna().sum()
```

Out[375]:
```
Price            0
Age_08_04        0
KM               0
HP               0
cc               0
Doors            0
Gears            0
Quarterly_Tax    0
Weight           0
dtype: int64
```

We can see that there are no null values in our dataset

In [376]:
```
df.shape
```

Out[376]:
```
(1436, 9)
```

In [377]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Price          1436 non-null   int64
 1   Age_08_04      1436 non-null   int64
 2   KM             1436 non-null   int64
 3   HP             1436 non-null   int64
 4   cc             1436 non-null   int64
 5   Doors          1436 non-null   int64
 6   Gears          1436 non-null   int64
 7   Quarterly_Tax  1436 non-null   int64
 8   Weight         1436 non-null   int64
dtypes: int64(9)
memory usage: 101.0 KB
```

All of our variables are integers

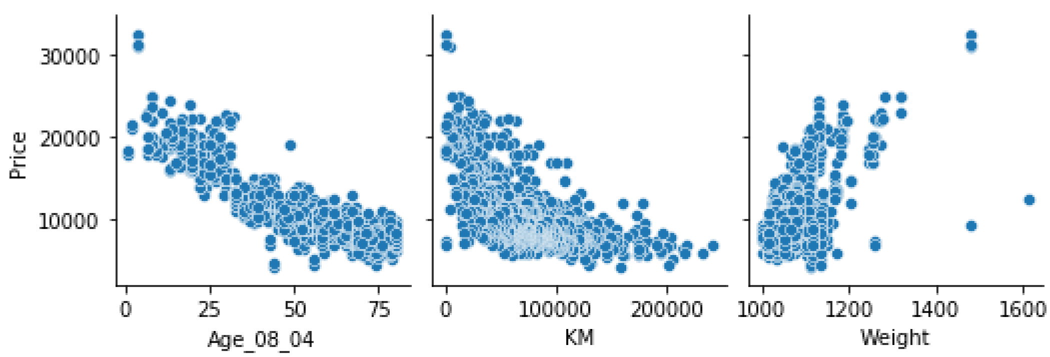
In [378]:
```
df.columns
```

Out[378]:
```
Index(['Price', 'Age_08_04', 'KM', 'HP', 'cc', 'Doors', 'Gears',
       'Quarterly_Tax', 'Weight'],
      dtype='object')
```

In [379]:
```
plt.figure(figsize=(30,8))
sns.pairplot(x_vars=['Age_08_04', 'KM','Weight'],y_vars=['Price'],data=df)
```

Out[379]:
```
<seaborn.axisgrid.PairGrid at 0x1318c590>
```

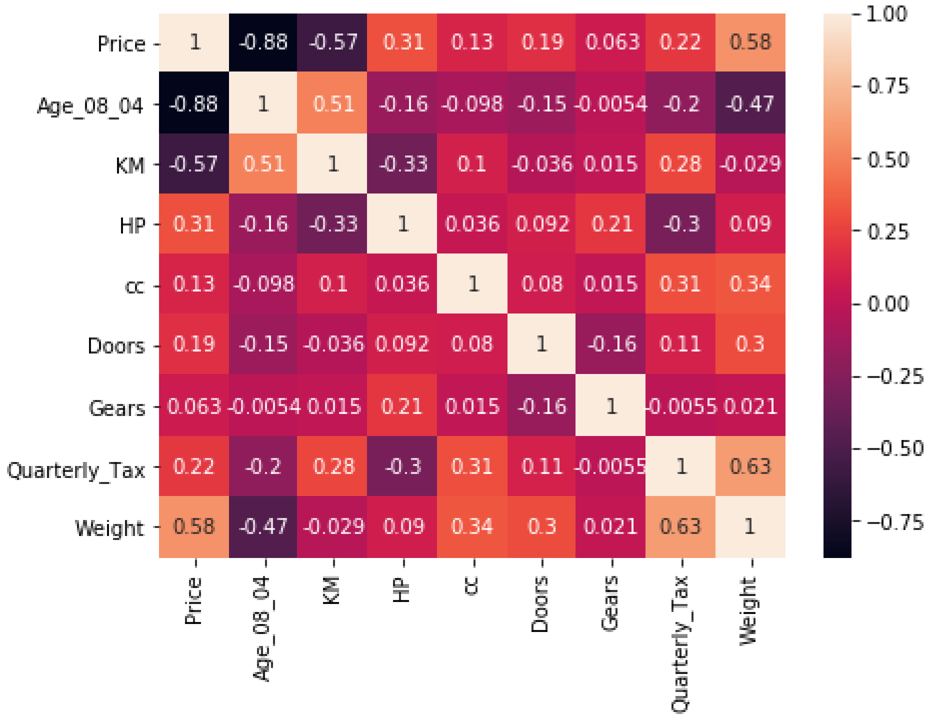
```
<Figure size 2160x576 with 0 Axes>
```



Here we can see that there is linear relationship between non categorical variables and the dependent variable

```
In [380]: plt.figure(figsize=(7,5))
          sns.heatmap(df.corr(),annot=True)
```

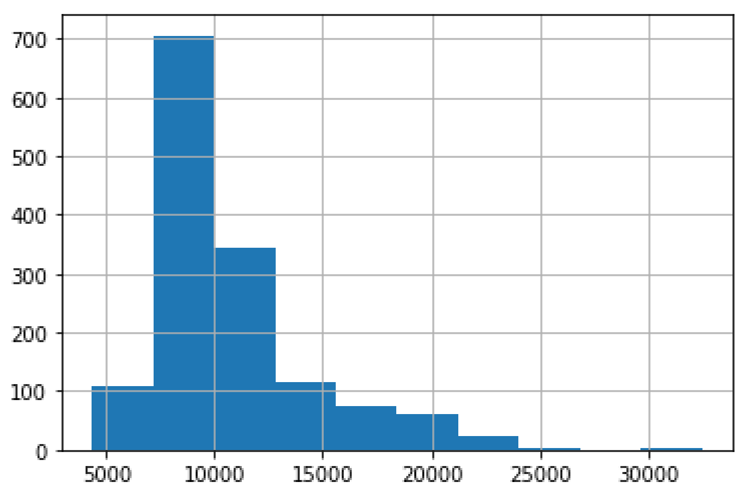
Out[380]: <matplotlib.axes._subplots.AxesSubplot at 0x12c836f0>



Here we can see that variable weight is correlated with other variables

```
In [381]: df['Price'].hist()
```

Out[381]: <matplotlib.axes._subplots.AxesSubplot at 0x12cc38f0>



Here we can see that variable price is highly positively skewed

In [382]:
```
x = df.drop(columns = ['Price'])
y = df['Price']
```

In [383]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.2, random_state =
```

In [384]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(xtrain,ytrain)
ypred = model.predict(xtest)
```

In [385]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
mse=mean_squared_error(ytest,ypred)
print('mse',mse)
rmse=np.sqrt(mse)
print('rmse',rmse)
mae=mean_absolute_error(ytest,ypred)
print('mae',mae)
score=r2_score(ytest,ypred)
print('r2_score',score)
```

```
mse 5664560.078922728
rmse 2380.033629788186
mae 1064.523058493787
r2_score 0.5978784673802031
```

Looking at the r2_square we can say our model is performing poorly that is because skewness of the target variavle i.e. price

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

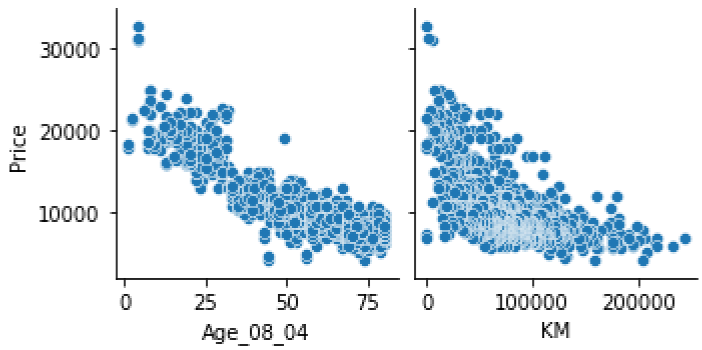
Verifying the assumptions of the linear regression

Linear relationship

In [386]:
```
plt.figure(figsize=(30,8))
sns.pairplot(x_vars=['Age_08_04', 'KM'],y_vars=['Price'],data=df)
```

Out[386]: <seaborn.axisgrid.PairGrid at 0x12d1d170>

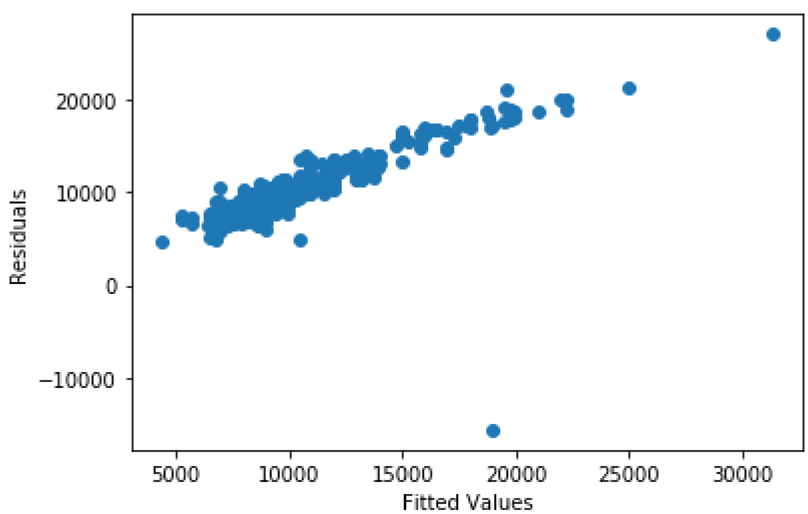
<Figure size 2160x576 with 0 Axes>



Homoscedisticity

In [387]:
```
plt.scatter(ytest.values,ypred)
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
```

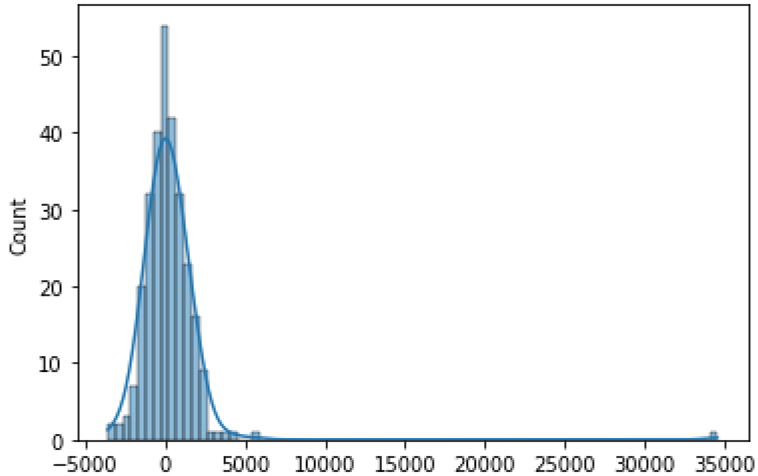
Out[387]: Text(0, 0.5, 'Residuals')



Residuals are normaly distributed

In [388]:
```
residuals = ytest.values - ypred
sns.histplot(residuals,kde=True)
```

Out[388]: <matplotlib.axes._subplots.AxesSubplot at 0x13150670>



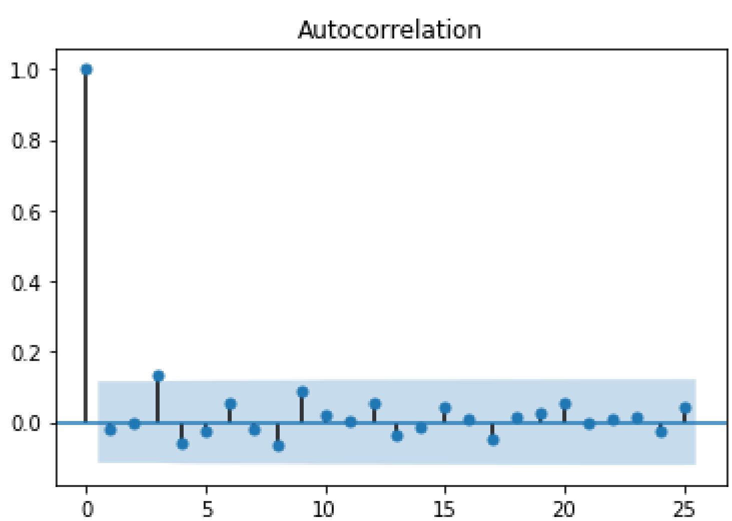
Mean of residual is zero

In [389]:
```
print("Mean of Residual is :", np.mean(residuals))
```

```
Mean of Residual is : 238.7292762796466
```

No Autocorrelation among residuals

In [390]:
```
import statsmodels.api as sm
sm.graphics.tsa.plot_acf(residuals)
plt.show()
```



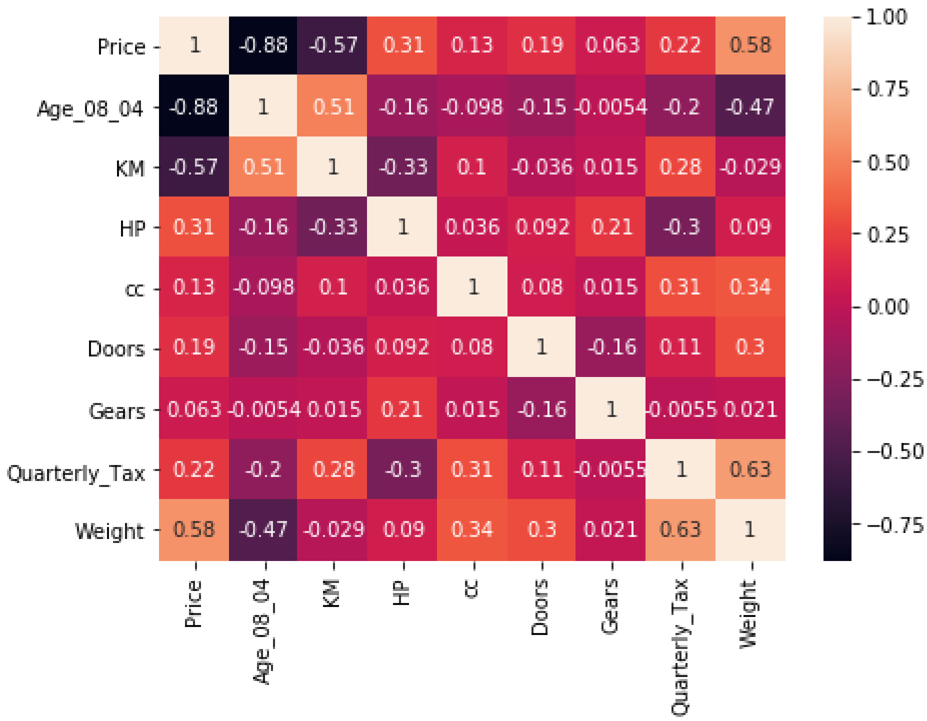
In [ ]:

In [ ]:

No multicolinearity

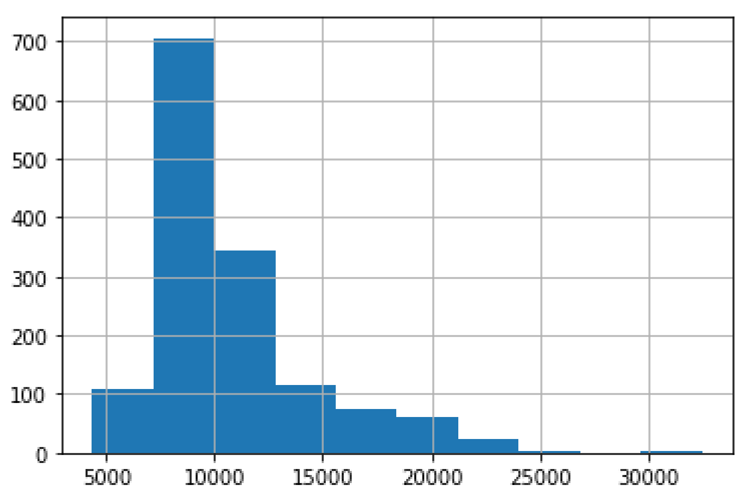
In [391]:
```
plt.figure(figsize=(7,5))
sns.heatmap(df.corr(),annot=True)
```

Out[391]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x134e4910>
```



In [392]:
```
df['Price'].hist()
```

Out[392]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x130a7df0>
```



Here we can see that variable price is skewed so we log transform the price variable also variable weight causing slight multicolinearity in the data so we have to remove it

In [393]:
```
df['target']=np.log(df.Price)
df = df.drop(columns = ['Price','Weight'] )
```

In [394]:
```
x = df.drop(columns = ['target'])
y = df['target']
```

In [395]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.2, random_state =
```

In [396]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(xtrain,ytrain)
ypred = model.predict(xtest)
```

In [397]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
linear_mse=mean_squared_error(ytest,ypred)
print("MSE:",linear_mse)
linear_rmse = np.sqrt(linear_mse)
print("RMSE:",linear_rmse)
linear_mae=mean_absolute_error(ytest,ypred)
print("MAE:",linear_mae)
linear_score=r2_score(ytest,ypred)
print("R-squared :",linear_score)
```

```
MSE: 0.015134471709828259
RMSE: 0.1230222407121097
MAE: 0.09320425145063185
R-squared : 0.8381581786099122
```

After transformation accuracy increased to 83% also the erros decreased significantly

Polynomial Regression

In [398]:
```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=3)
poly_reg.fit(xtrain)
x_train_poly=poly_reg.transform(xtrain)
x_test_poly=poly_reg.transform(xtest)
```

In [399]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train_poly,ytrain)
y_pred=lr.predict(x_test_poly)
```

In [400]:
```
poly_mse=mean_squared_error(ytest,y_pred)
print("MSE:",poly_mse)
poly_rmse = np.sqrt(poly_mse)
print("RMSE:",poly_rmse)
poly_mae=mean_absolute_error(ytest,y_pred)
print("MAE:",poly_mae)
poly_score=r2_score(ytest,y_pred)
print("R-squared :",poly_score)
```

```
MSE: 6370949.775295781
RMSE: 2524.0740431484533
MAE: 148.8313775385129
R-squared : -68128317.93888026
```

Fit the Ridge Regression model

In [401]:
```
from sklearn.linear_model import Ridge
ridge=Ridge(alpha=0.3)
ridge.fit(xtrain,ytrain)
rypred=ridge.predict(xtest)
```

In [402]:
```
ridge_mse=mean_squared_error(ytest,rypred)
print('mse',ridge_mse)
ridge_rmse=np.sqrt(ridge_mse)
print('rmse',ridge_rmse)
ridge_mae=mean_absolute_error(ytest,rypred)
print('mae',ridge_mae)
ridge_score=r2_score(ytest,rypred)
print('r2_score',ridge_score)
```

```
mse 0.015137440040743 09
rmse 0.1230343043250259
mae 0.09321012888099021
r2_score 0.838126436498859
```

fit the Lasso Regression model

In [403]:
```
from sklearn.linear_model import Lasso
lasso=Lasso(alpha=0.0001)
lasso.fit(xtrain,ytrain)
lypred=lasso.predict(xtest)
```

In [404]:
```
lasso_mse=mean_squared_error(ytest,lypred)
print('mse',lasso_mse)
lasso_rmse=np.sqrt(lasso_mse)
print('rmse',lasso_rmse)
lasso_mae=mean_absolute_error(ytest,lypred)
print('mae',lasso_mae)
lasso_score=r2_score(ytest,lypred)
print('r2_score',lasso_score)
```

```
mse 0.015149762636119292
rmse 0.12308437202228109
mae 0.09324379694392951
r2_score 0.8379946637275211
```

Fit the ElasticNet Regression Model

In [405]:
```
from sklearn.linear_model import ElasticNet
```

In [406]:
```
alpha=[0.0001,0.001,0.01,0.1,0.3,0.5,1,10]
```

In [407]:
```
# function for getting best alpha value
scores={}
def get_best_alpha(alpha):
    for i in alpha:
        model=ElasticNet(alpha=i)
        model.fit(xtrain,ytrain)
        ypred=model.predict(xtest)
        elastic_mse=mean_squared_error(ytest,ypred)
        elastic_rmse=np.sqrt(elastic_mse)
        elastic_mae=mean_absolute_error(ytest,ypred)
        scores[i]=model.score(xtest,ytest)
        #print(" For Alpha = {} | R-square :{} MSE :{} RMSE :{} MAE:{} ".format(i
    return max(scores, key= lambda x: scores[x]),elastic_mse,elastic_rmse,elastic
```

In [408]:
```
best_score,elastic_mse,elastic_rmse,elastic_mae=get_best_alpha(alpha)
print('mse',elastic_mse)
print('rmse',elastic_rmse)
print('mae',elastic_mae)
```

```
mse 0.14928420022574718
rmse 0.3863731360042351
mae 0.1989431306618824
```

In [409]:
```
print("Best Alpha is :",best_score)
print('R_square of the model is :',scores[best_score])
```

```
Best Alpha is : 0.1
R_square of the model is : 0.8468754985505694
```

Evaluation

In [410]:
```
models = ['LinearRegression','Polynomial','Ridge','Lasso','ElasticNet']
rsquare = [linear_score,poly_score,ridge_score,lasso_score,scores[0.3]]
mse = [linear_mse,poly_mse,ridge_mse,lasso_mse,elastic_mse]
rmse = [linear_rmse,poly_rmse,ridge_rmse,lasso_rmse,elastic_rmse]
mae = [linear_mae,poly_mae,ridge_mae,lasso_mae,elastic_mae]
Evaluation = pd.DataFrame({'Model':models,'R-square':rsquare,'MSE':mse,'RMSE':rms
```

In [411]:
```
Evaluation
```

Out[411]:

	Model	R-square	MSE	RMSE	MAE
0	LinearRegression	8.381582e-01	1.513447e-02	0.123022	0.093204
1	Polynomial	-6.812832e+07	6.370950e+06	2524.074043	148.831378
2	Ridge	8.381264e-01	1.513744e-02	0.123034	0.093210
3	Lasso	8.379947e-01	1.514976e-02	0.123084	0.093244
4	ElasticNet	8.406291e-01	1.492842e-01	0.386373	0.198943

After observing the evaluation table we can conclude that ElasticNet is the best model for predicting price of the car