5. Choose an appropriate application, find a solution using linear regression and express its performance measures.

Car Price Prediction

```
# importing necessary packages

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


%matplotlib inline
```

```
cars_info = pd.read_csv("/content/drive/MyDrive/AI-ML/DM/CarPrice_Assignment.csv")
cars_info.head()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	d
0	1	3	alfa-romero giulia	gas	std	two	convertible	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	
3	4	2	audi 100 ls	gas	std	four	sedan	
4	5	2	audi 100ls	gas	std	four	sedan	

```
# basic infos

cars_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
```

```
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

```
cars_info.describe()
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbwe
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.0(
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.5(
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.6{
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.0(
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.0(
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.0(
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.0(
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.0(

```
# checking for null values

cars_info.isnull().sum()
```

```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
```

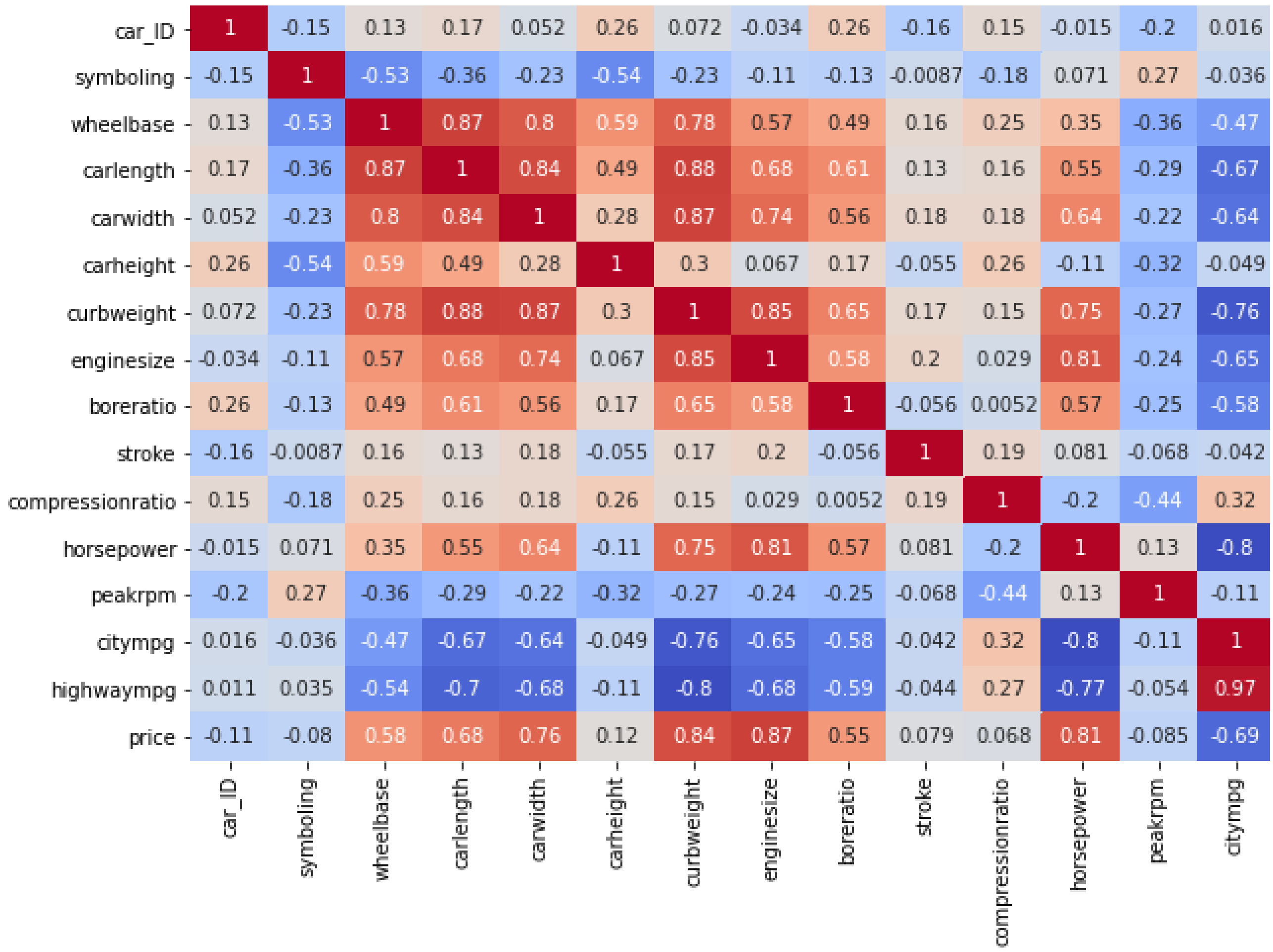
```
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

```
#Check the correlation between each of the columns
cars_info.corr()

# There is good correlation between
# 1. engine size and horse power
# 2. curb weight and car length
# 3. car width and car length
# 4. Engine size and price
# 5. curbweight and price
```

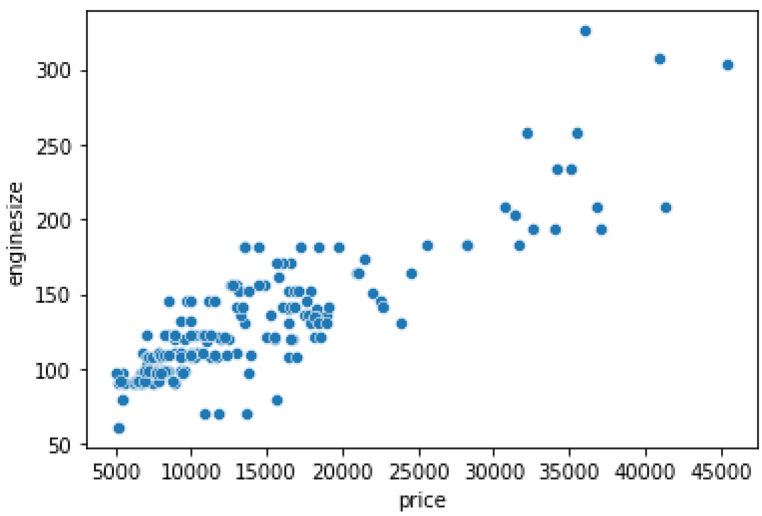
	car_ID	symboling	wheelbase	carlength	carwidth	carheight
car_ID	1.000000	-0.151621	0.129729	0.170636	0.052387	0.255960
symboling	-0.151621	1.000000	-0.531954	-0.357612	-0.232919	-0.541038
wheelbase	0.129729	-0.531954	1.000000	0.874587	0.795144	0.589435
carlength	0.170636	-0.357612	0.874587	1.000000	0.841118	0.491029
carwidth	0.052387	-0.232919	0.795144	0.841118	1.000000	0.279210
carheight	0.255960	-0.541038	0.589435	0.491029	0.279210	1.000000
curbweight	0.071962	-0.227691	0.776386	0.877728	0.867032	0.295572
enginesize	-0.033930	-0.105790	0.569329	0.683360	0.735433	0.067149
boreratio	0.260064	-0.130051	0.488750	0.606454	0.559150	0.171071
stroke	-0.160824	-0.008735	0.160959	0.129533	0.182942	-0.055307
compressionratio	0.150276	-0.178515	0.249786	0.158414	0.181129	0.261214
horsepower	-0.015006	0.070873	0.353294	0.552623	0.640732	-0.108802
peakrpm	-0.203789	0.273606	-0.360469	-0.287242	-0.220012	-0.320411
citympg	0.015940	-0.035823	-0.470414	-0.670909	-0.642704	-0.048640
highwaympg	0.011255	0.034606	-0.544082	-0.704662	-0.677218	-0.107358
price	-0.109093	-0.079978	0.577816	0.682920	0.759325	0.119336

```
# visulaizing correlation using heat map

plt.figure(figsize=(14,7))
sns.heatmap(cars_info.corr(),annot=True,cmap='coolwarm')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f90eacbd350>
```




```
# highest correlation is engine size vs price

# scatter plot

sns.scatterplot(x='price',y='enginesize',data=cars_info)
plt.show()
```



```
# dropping these columns

cars_info.drop(['car_ID','CarName','fuelsystem','enginetype'],axis=1,inplace=True)
```

```
cars_info.head()
```

	symboling	fueltype	aspiration	doornumber	carbody	drivewheel	enginelc
0	3	gas	std	two	convertible	rwd	
1	3	gas	std	two	convertible	rwd	
2	1	gas	std	two	hatchback	rwd	
3	2	gas	std	four	sedan	fwd	
4	2	gas	std	four	sedan	4wd	

```
# encoding strings to number to use it in the model

# mapping

cars_info['fueltype'] = cars_info['fueltype'].map({'gas':'0','diesel':'1'})
cars_info['aspiration'] = cars_info['aspiration'].map({'std':'0','turbo':'1'})
cars_info['doornumber'] = cars_info['doornumber'].map({'two':'2','four':'4'})
cars_info['carbody'] = cars_info['carbody'].map({'convertible':'0','hatchback':'1'
cars_info['drivewheel'] = cars_info['drivewheel'].map({'rwd':'0','fwd':'1','4wd':'
cars_info['enginelocation'] = cars_info['enginelocation'].map({'front':'0','rear':
cars_info['cylindernumber'] = cars_info['cylindernumber'].map({'four':'4','six':'6
# checking
cars_info.head()
```

	symboling	fueltype	aspiration	doornumber	carbody	drivewheel	engineloc
0	3	0	0	2	0	0	
1	3	0	0	2	0	0	
2	1	0	0	2	1	0	
3	2	0	0	4	2	1	
4	2	0	0	4	2	2	

Model building part

```
# taking price as the label or target


from sklearn.model_selection import train_test_split

X = cars_info.drop('price',axis=1)
```

```
X = cars_info.drop('price',axis=1)
y= cars_info['price']

X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.3,random_state=55)
```

```
# training or fitting the curve

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train, )
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
```

```
# prediction part

prediction = lr.predict(X_test)
```

Perfomance measures

```
# r2 score

from sklearn.metrics import r2_score
print(r2_score(y_test,prediction))

# got 78% accuracy
```

```
0.7784967344806435
```

```
# The explained_variance_score computes the explained variance regression score.

from sklearn.metrics import explained_variance_score

explained_variance_score(y_test, prediction, multioutput='raw_values')
```

```
array([0.77858187])
```

```
# The max_error function computes the maximum residual error ,
# a metric that captures the worst case error between the predicted
# value and the true value.

from sklearn.metrics import max_error
max_error(y_test, prediction)
```

```
16867.0309776942
```

```
# Mean Abosulute Error

from sklearn.metrics import mean_absolute_error
```

```
mean_absolute_error(y_test, prediction, multioutput='raw_values')
```

```
array([2343.26176044])
```

```
# Mean squared error

from sklearn.metrics import mean_squared_error

mean_squared_error(y_test, prediction)
```

```
13119405.902743611
```

```
# test data vs prediction

plt.scatter(y_test, prediction)
```

```
<matplotlib.collections.PathCollection at 0x7f90e734e610>
```