21MIS1152 Rajeev Sekar

```
import numpy as np
import pandas as pd

dataset = pd.read_csv('auto_prize.csv')
dataset.head()
```

```
   symboling  normalized-losses  wheel-base      length      width
height  \
0          5                164   99.800003  176.600006  66.199997
54.299999
1          5                164   99.400002  176.600006  66.400002
54.299999
2          4                158  105.800003  192.699997  71.400002
55.700001
3          4                158  105.800003  192.699997  71.400002
55.900002
4          5                192  101.199997  176.800003  64.800003
54.299999

   curb-weight  engine-size  bore  stroke  compression-ratio
horsepower  \
0         2337          109  3.19     3.4               10.0
102
1         2824          136  3.19     3.4                8.0
115
2         2844          136  3.19     3.4                8.5
110
3         3086          131  3.13     3.4                8.3
140
4         2395          108  3.50     2.8                8.8
101

   peak-rpm  city-mpg  highway-mpg  target
0      5500        24           30   13950
1      5500        18           22   17450
2      5500        19           25   17710
3      5500        17           20   23875
4      5800        23           29   16430
```

```
dataset.describe()
```

```
        symboling  normalized-losses  wheel-base      length
width  \
count  159.000000         159.000000  159.000000  159.000000
159.000000
mean     3.735849         121.132075   98.264151  172.413837
65.607547
std      1.193086          35.651285    5.167417   11.523177
```

```
1.947883
min      1.000000          65.000000   86.599998  141.100006
60.299999
25%      3.000000          94.000000   94.500000  165.650002
64.000000
50%      4.000000         113.000000   96.900002  172.399994
65.400002
75%      5.000000         148.000000  100.799999  177.800003
66.500000
max      6.000000         256.000000  115.599998  202.600006
71.699997

           height  curb-weight  engine-size        bore      stroke  \
count  159.000000   159.000000   159.000000  159.000000  159.000000
mean    53.899371  2461.138365   119.226415    3.300126    3.236352
std      2.268761   481.941321    30.460791    0.267336    0.294888
min     49.400002  1488.000000    61.000000    2.540000    2.070000
25%     52.250000  2065.500000    97.000000    3.050000    3.105000
50%     54.099998  2340.000000   110.000000    3.270000    3.270000
75%     55.500000  2809.500000   135.000000    3.560000    3.410000
max     59.799999  4066.000000   258.000000    3.940000    4.170000

       compression-ratio  horsepower     peak-rpm    city-mpg
highway-mpg  \
count         159.000000  159.000000   159.000000  159.000000
159.000000
mean           10.161132   95.836478  5113.836478   26.522013
32.081761
std             3.889475   30.718583   465.754864    6.097142
6.459189
min             7.000000   48.000000  4150.000000   15.000000
18.000000
25%             8.700000   69.000000  4800.000000   23.000000
28.000000
50%             9.000000   88.000000  5200.000000   26.000000
32.000000
75%             9.400000  114.000000  5500.000000   31.000000
37.000000
max            23.000000  200.000000  6600.000000   49.000000
54.000000

             target
count    159.000000
mean   11445.729560
std     5877.856195
min     5118.000000
25%     7372.000000
50%     9233.000000
75%    14719.500000
max    35056.000000
```

```
#splitting feature and target cols
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:,-1].values
```

Finding correlation between the cols of the dataset

```
dataset.corr()
```

```
                   symboling  normalized-losses  wheel-base    length
\
symboling           1.000000           0.518344   -0.520591 -0.336257

normalized-losses   0.518344           1.000000   -0.060086  0.035541

wheel-base         -0.520591          -0.060086    1.000000  0.871535

length             -0.336257           0.035541    0.871535  1.000000

width              -0.219186           0.109726    0.814991  0.838338

height             -0.475185          -0.413702    0.555767  0.499251

curb-weight        -0.251880           0.125858    0.810182  0.871291

engine-size        -0.109453           0.207820    0.649206  0.725953

bore               -0.256469          -0.031558    0.578159  0.646318

stroke             -0.021285           0.063330    0.167449  0.121073

compression-ratio  -0.138316          -0.127259    0.291431  0.184814

horsepower         -0.003949           0.290511    0.516948  0.672063

peak-rpm            0.199106           0.237697   -0.289234 -0.234074

city-mpg            0.089550          -0.235523   -0.580657 -0.724544

highway-mpg         0.149830          -0.188564   -0.611750 -0.724599

target             -0.162794           0.202761    0.734419  0.760952


                      width    height  curb-weight  engine-size
bore  \
symboling         -0.219186 -0.475185    -0.251880    -0.109453 -
0.256469
normalized-losses  0.109726 -0.413702     0.125858     0.207820 -
0.031558
wheel-base         0.814991  0.555767     0.810182     0.649206
0.578159
```

```
length              0.838338  0.499251     0.871291     0.725953
0.646318
width               1.000000  0.292706     0.870595     0.779253
0.572554
height              0.292706  1.000000     0.367052     0.111083
0.254836
curb-weight         0.870595  0.367052     1.000000     0.888626
0.645792
engine-size         0.779253  0.111083     0.888626     1.000000
0.595737
bore                0.572554  0.254836     0.645792     0.595737
1.000000
stroke              0.196619 -0.091313     0.173844     0.299683 -
0.102581
compression-ratio   0.258752  0.233308     0.224724     0.141097
0.015119
horsepower          0.681872  0.034317     0.790095     0.812073
0.560239
peak-rpm           -0.232216 -0.245864    -0.259988    -0.284686 -
0.312269
city-mpg           -0.666684 -0.199738    -0.762155    -0.699139 -
0.590440
highway-mpg        -0.693338 -0.226136    -0.789338    -0.714095 -
0.590850
target              0.843371  0.244836     0.893639     0.841496
0.533890

                     stroke  compression-ratio  horsepower  peak-
rpm  \
symboling          -0.021285          -0.138316   -0.003949  0.199106

normalized-losses   0.063330          -0.127259    0.290511  0.237697

wheel-base          0.167449           0.291431    0.516948 -0.289234

length              0.121073           0.184814    0.672063 -0.234074

width               0.196619           0.258752    0.681872 -0.232216

height             -0.091313           0.233308    0.034317 -0.245864

curb-weight         0.173844           0.224724    0.790095 -0.259988

engine-size         0.299683           0.141097    0.812073 -0.284686

bore               -0.102581           0.015119    0.560239 -0.312269

stroke              1.000000           0.243587    0.148804 -0.011312

compression-ratio   0.243587           1.000000   -0.162305 -0.416769
```

```
horsepower          0.148804          -0.162305    1.000000  0.074057

peak-rpm           -0.011312          -0.416769    0.074057  1.000000

city-mpg           -0.020055           0.278332   -0.837214 -0.052929

highway-mpg        -0.012934           0.221483   -0.827941 -0.032777

target              0.160664           0.209361    0.759874 -0.171916


                    city-mpg  highway-mpg    target
symboling           0.089550     0.149830 -0.162794
normalized-losses  -0.235523    -0.188564  0.202761
wheel-base         -0.580657    -0.611750  0.734419
length             -0.724544    -0.724599  0.760952
width              -0.666684    -0.693338  0.843371
height             -0.199738    -0.226136  0.244836
curb-weight        -0.762155    -0.789338  0.893639
engine-size        -0.699139    -0.714095  0.841496
bore               -0.590440    -0.590850  0.533890
stroke             -0.020055    -0.012934  0.160664
compression-ratio   0.278332     0.221483  0.209361
horsepower         -0.837214    -0.827941  0.759874
peak-rpm           -0.052929    -0.032777 -0.171916
city-mpg            1.000000     0.971999 -0.692273
highway-mpg         0.971999     1.000000 -0.720090
target             -0.692273    -0.720090  1.000000
```

the loosely correlated pair of (feature, output) should be dropped in order to apply linear regression model

```
dataset = dataset.drop(["symboling", "normalized-
losses","height","stroke","compression-ratio","peak-rpm"], axis=1)

dataset.corr()
```

```
             wheel-base    length     width  curb-weight  engine-size
\
wheel-base     1.000000  0.871535  0.814991     0.810182     0.649206

length         0.871535  1.000000  0.838338     0.871291     0.725953

width          0.814991  0.838338  1.000000     0.870595     0.779253

curb-weight    0.810182  0.871291  0.870595     1.000000     0.888626

engine-size    0.649206  0.725953  0.779253     0.888626     1.000000
```

```
bore            0.578159  0.646318  0.572554     0.645792     0.595737

horsepower      0.516948  0.672063  0.681872     0.790095     0.812073

city-mpg       -0.580657 -0.724544 -0.666684    -0.762155    -0.699139

highway-mpg    -0.611750 -0.724599 -0.693338    -0.789338    -0.714095

target          0.734419  0.760952  0.843371     0.893639     0.841496
```

```
                   bore  horsepower  city-mpg  highway-mpg    target
wheel-base     0.578159    0.516948 -0.580657    -0.611750  0.734419
length         0.646318    0.672063 -0.724544    -0.724599  0.760952
width          0.572554    0.681872 -0.666684    -0.693338  0.843371
curb-weight    0.645792    0.790095 -0.762155    -0.789338  0.893639
engine-size    0.595737    0.812073 -0.699139    -0.714095  0.841496
bore           1.000000    0.560239 -0.590440    -0.590850  0.533890
horsepower     0.560239    1.000000 -0.837214    -0.827941  0.759874
city-mpg      -0.590440   -0.837214  1.000000     0.971999 -0.692273
highway-mpg   -0.590850   -0.827941  0.971999     1.000000 -0.720090
target         0.533890    0.759874 -0.692273    -0.720090  1.000000
```

scaling the features (because the ranges are varying) and splitting training and testing sets

```
from sklearn.preprocessing import MinMaxScaler
# define min max scaler
scaler = MinMaxScaler()
# transform data
X = scaler.fit_transform(X)

#splitting train and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.30,random_state=125)
```

MODEL TRAINING: linear regression model is used because the target variable is a continous value => regression and the features are highly correlated with the target (linear relation)

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

LinearRegression()

y_pred = regressor.predict(X_test)
```

MODEL EVALUATION: r-squared is used as a metric to evaluate the model

```
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print('r2 score: ', r2)
```

```
r2 score:  0.8804433019303708
```

R-2 of 88% is a good model (>60%)

MODEL INTERPRETATION: the coefficients(b1...bn) of all the features and the intercept(b0) from the equation : y=b0+b1x1+b2x2+...+bnxn

```
regressor.coef_
```

```
array([  -76.37101268,   874.3034376 ,  6030.13766262, -4584.71242111,
        7207.75333962,  -837.62703782,  9671.68680982, 12731.16586569,
       -2244.62151685, -4894.5030305 ,  2617.00906314,  1607.93402346,
        2242.05407738, -5985.52571376,  1394.35434156])
```

```
regressor.intercept_
```

```
4251.92368904624
```

=> from the coef array we can say that: increase in 1 unit in the 1st feature (wheel_base) will decrease the target(auto_prize) by 76.37 times, similarly the coefs of other features carry the same meaning

=> the predicted price of the automobile is 4251.92(intercept) units when hypothetically all the features are 0

```

```