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

import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
```

```python
df = pd.read_csv(r'C:\Users\deepa\Desktop\cars.csv')
df.head()
```

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	13495
1	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	27	16500
2	1	?	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154	19	26	16500
3	2	164	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102	24	30	13950
4	2	164	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115	18	22	17450

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          205 non-null    int64  
 1   normalized-losses  205 non-null    object 
 2   make               205 non-null    object 
 3   fuel-type          205 non-null    object 
 4   body-style         205 non-null    object 
 5   drive-wheels       205 non-null    object 
 6   engine-location    205 non-null    object 
 7   width              205 non-null    float64
 8   height             205 non-null    float64
 9   engine-type        205 non-null    object 
 10  engine-size        205 non-null    int64  
 11  horsepower         205 non-null    object 
 12  city-mpg           205 non-null    int64  
 13  highway-mpg        205 non-null    int64  
 14  price              205 non-null    int64  
dtypes: float64(2), int64(5), object(8)
memory usage: 24.1+ KB
```

```python
df.isna().sum()
```

```
symboling            0
normalized-losses    0
make                 0
fuel-type            0
body-style           0
drive-wheels         0
engine-location      0
width                0
height               0
engine-type          0
engine-size          0
horsepower           0
city-mpg             0
highway-mpg          0
price                0
dtype: int64
```

```python
df['normalized-losses'].value_counts()
```

```
?      41
161    11
91      8
150     7
134     6
104     6
128     6
85      5
168     5
74      5
95      5
94      5
103     5
65      5
102     5
106     4
118     4
148     4
93      4
122     4
137     3
83      3
154     3
125     3
101     3
115     3
119     2
108     2
129     2
164     2
188     2
113     2
192     2
153     2
194     2
81      2
89      2
158     2
197     2
145     2
87      2
110     2
231     1
98      1
107     1
90      1
142     1
77      1
256     1
78      1
121     1
186     1
Name: normalized-losses, dtype: int64
```

```python
df['horsepower'].value_counts()
```

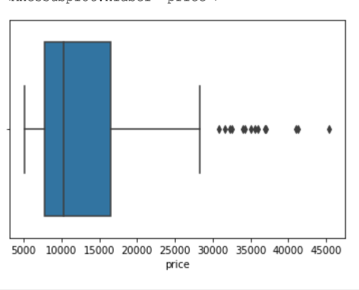
```
68     19
70     11
69     10
116     9
110     8
95      7
62      6
101     6
114     6
160     6
88      6
76      5
97      5
145     5
84      5
82      5
102     5
123     4
86      4
92      4
111     4
90      3
85      3
182     3
121     3
207     3
152     3
73      3
?       2
100     2
155     2
161     2
156     2
112     2
94      2
176     2
184     2
52      2
56      2
162     2
72      1
154     1
120     1
288     1
58      1
200     1
142     1
115     1
64      1
262     1
140     1
134     1
135     1
106     1
55      1
175     1
78      1
143     1
60      1
48      1
Name: horsepower, dtype: int64
```

```python
df['normalized-losses'].replace('?', np.nan, inplace=True)
df['horsepower'].replace('?', np.nan, inplace=True)

df['normalized-losses'] = df['normalized-losses'].astype(float)
df['horsepower'] = df['horsepower'].astype(float)


nlmean = df['normalized-losses'].mean()
hpmean =df['horsepower'].mean()

df['normalized-losses'].fillna(nlmean, inplace=True)
df['horsepower'].fillna(hpmean, inplace=True)
```

Outliers

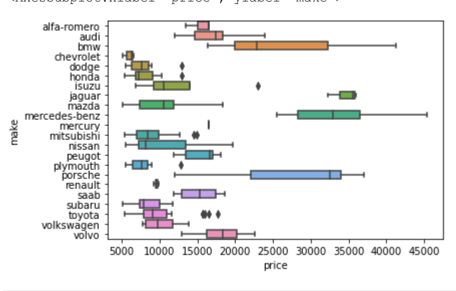
```python
sns.boxplot(df['price'])
```

<AxesSubplot:xlabel='price'>




```python
sns.boxplot(data=df,x='price',y='make')
```

<AxesSubplot:xlabel='price', ylabel='make'>

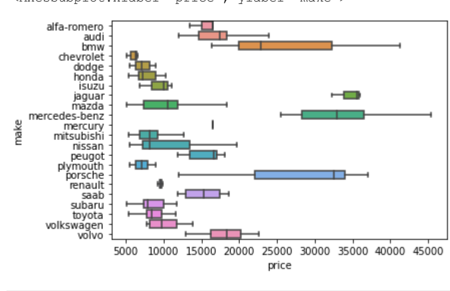



```python
df[(df['make']=='honda') & (df['price']>11000)]
df[(df['make']=='toyota') & (df['price']>14000)]
df[(df['make']=='dodge') & (df['price']>11000)]

df.drop(41,inplace=True)
df.drop([172,178,179,180,181],inplace=True)
df.drop(29,inplace=True)
```

```python
df[(df['make']=='isuzu') & (df['price']>20000)]
df[(df['make']=='mitsubishi') & (df['price']>13000)]
df[(df['make']=='plymouth') & (df['price']>10000)]

df.drop(45,inplace=True)
df.drop([83,84],inplace=True)
df.drop(124,inplace=True)
```

```python
sns.boxplot(data=df,x='price',y='make')
```

<AxesSubplot:xlabel='price', ylabel='make'>

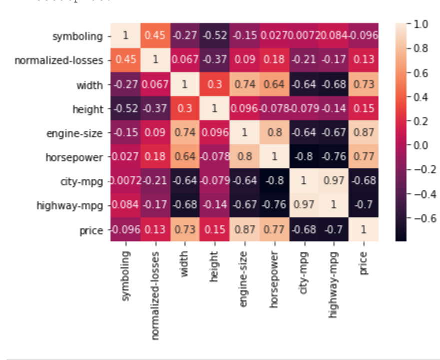



```python
df.describe()
```

	symboling	normalized-losses	width	height	engine-size	horsepower	city-mpg	highway-mpg	price
count	194.000000	194.000000	194.000000	194.000000	194.000000	194.000000	194.000000	194.000000	194.000000
mean	0.788660	121.402062	65.892268	53.822165	125.628866	102.430476	25.412371	30.984536	13087.304124
std	1.200896	31.406299	2.189327	2.448045	42.036826	39.307294	6.546371	6.882039	8075.197621
min	-2.000000	65.000000	60.300000	47.800000	61.000000	48.000000	13.000000	16.000000	5118.000000
25%	0.000000	101.000000	64.000000	52.000000	97.000000	70.000000	19.250000	25.000000	7747.250000
50%	1.000000	122.000000	65.400000	54.100000	110.000000	95.000000	25.000000	30.500000	9992.000000
75%	2.000000	137.000000	66.900000	55.675000	140.750000	116.000000	30.000000	36.000000	16500.000000
max	3.000000	256.000000	72.300000	59.800000	326.000000	288.000000	49.000000	54.000000	45400.000000

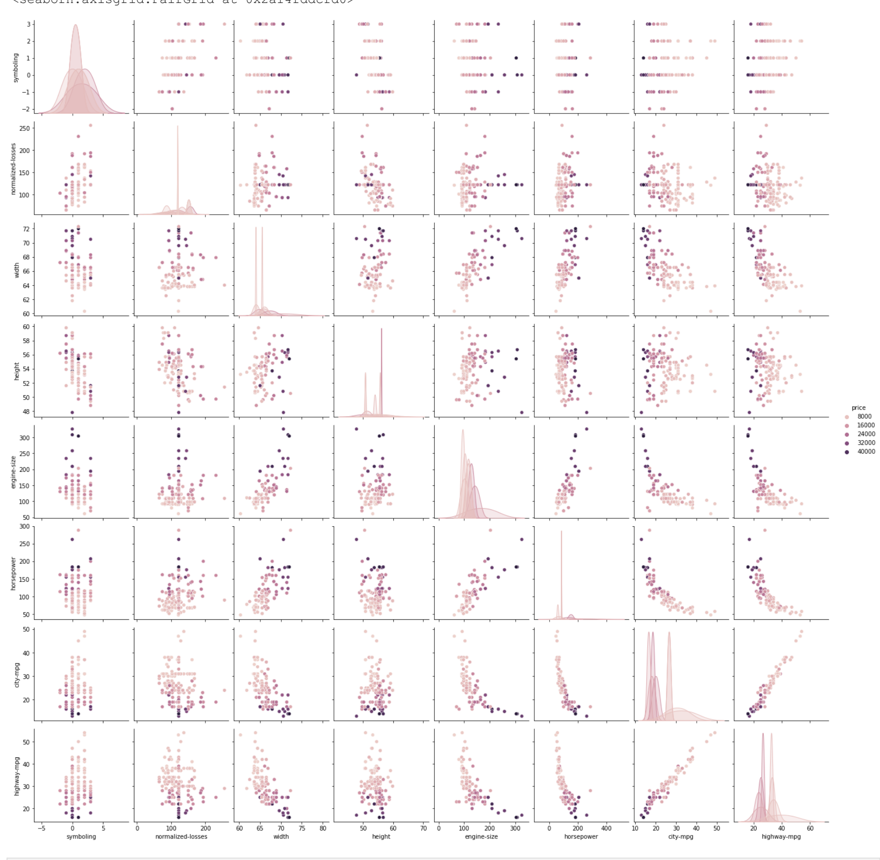
```python
sns.heatmap(df.corr() , annot=True)
```

<AxesSubplot:>




```python
sns.pairplot(df, hue='price')
```

<seaborn.axisgrid.PairGrid at 0x2a14fddcfd0>




```python
df_num = df.select_dtypes(['int64','float64'])
df_cat = df.select_dtypes('object')
```

```python
df_num
```

	symboling	normalized-losses	width	height	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	122.0	64.1	48.8	130	111.0	21	27	13495
1	3	122.0	64.1	48.8	130	111.0	21	27	16500
2	1	122.0	65.5	52.4	152	154.0	19	26	16500
3	2	164.0	66.2	54.3	109	102.0	24	30	13950
4	2	164.0	66.4	54.3	136	115.0	18	22	17450
...	...	...	...	...	...	...	...	...	...
200	-1	95.0	68.9	55.5	141	114.0	23	28	16845
201	-1	95.0	68.8	55.5	141	160.0	19	25	19045
202	-1	95.0	68.9	55.5	173	134.0	18	23	21485
203	-1	95.0	68.9	55.5	145	106.0	26	27	22470
204	-1	95.0	68.9	55.5	141	114.0	19	25	22625

194 rows × 9 columns

```python
df_cat
```

	make	fuel-type	body-style	drive-wheels	engine-location	engine-type
0	alfa-romero	gas	convertible	rwd	front	dohc
1	alfa-romero	gas	convertible	rwd	front	dohc
2	alfa-romero	gas	hatchback	rwd	front	ohcv
3	audi	gas	sedan	fwd	front	ohc
4	audi	gas	sedan	4wd	front	ohc
...	...	...	...	...	...	...
200	volvo	gas	sedan	rwd	front	ohc
201	volvo	gas	sedan	rwd	front	ohc
202	volvo	gas	sedan	rwd	front	ohcv
203	volvo	diesel	sedan	rwd	front	ohc
204	volvo	gas	sedan	rwd	front	ohc

194 rows × 6 columns

```python
from sklearn.preprocessing import LabelEncoder
```

```python
for col in df_cat:
    le = LabelEncoder()
    df_cat[col] = le.fit_transform(df_cat[col])
```

```python
df_cat
```

	make	fuel-type	body-style	drive-wheels	engine-location	engine-type
0	0	1	0	2	0	0
1	0	1	0	2	0	0
2	0	1	2	2	0	5
3	1	1	3	1	0	3
4	1	1	3	0	0	3
...	...	...	...	...	...	...
200	21	1	3	2	0	3
201	21	1	3	2	0	3
202	21	1	3	2	0	5
203	21	0	3	2	0	3
204	21	1	3	2	0	3

194 rows × 6 columns

```python
df_new = pd.concat([df_num, df_cat], axis=1)
df_new
```

	symboling	normalized-losses	width	height	engine-size	horsepower	city-mpg	highway-mpg	price	make	fuel-type	body-style	drive-wheels	engine-location	engine-type
0	3	122.0	64.1	48.8	130	111.0	21	27	13495	0	1	0	2	0	0
1	3	122.0	64.1	48.8	130	111.0	21	27	16500	0	1	0	2	0	0
2	1	122.0	65.5	52.4	152	154.0	19	26	16500	0	1	2	2	0	5
3	2	164.0	66.2	54.3	109	102.0	24	30	13950	1	1	3	1	0	3
4	2	164.0	66.4	54.3	136	115.0	18	22	17450	1	1	3	0	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	68.9	55.5	141	114.0	23	28	16845	21	1	3	2	0	3
201	-1	95.0	68.8	55.5	141	160.0	19	25	19045	21	1	3	2	0	3
202	-1	95.0	68.9	55.5	173	134.0	18	23	21485	21	1	3	2	0	5
203	-1	95.0	68.9	55.5	145	106.0	26	27	22470	21	0	3	2	0	3
204	-1	95.0	68.9	55.5	141	114.0	19	25	22625	21	1	3	2	0	3

194 rows × 15 columns

```python
x = df_new.iloc[:, :-1].values
y = df_new.iloc[:,-1].values
```

```python
xtrain, xtest, ytrain, ytest =train_test_split(x,y, test_size=0.2, random_state=0)
```

```python
linreg = LinearRegression()

linreg.fit(xtrain, ytrain)

ypred = linreg.predict(xtest)
```

```python
print(f'MAError -: {mean_absolute_error(ytest, ypred)}')
print(f'MSError -: {mean_squared_error(ytest, ypred)}')
print(f'RMSError -: {np.sqrt(mean_absolute_error(ytest, ypred))}')
```

```
MAError -: 0.8184842045596241
MSError -: 1.8715722675939368
RMSError -: 0.9047011686516295
```

Hyperparameter Tunning

Ridge regression

```python
from sklearn.linear_model import Ridge
```

```python
ridge = Ridge()
```

```python
parameters = {'alpha' : [1, 5, 10, 20, 30, 40]}
```

```python
ridge_regressor = GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error',cv=10)
ridge_regressor.fit(x,y)
```

```
GridSearchCV(cv=10, estimator=Ridge(),
             param_grid={'alpha': [1, 5, 10, 20, 30, 40]},
             scoring='neg_mean_squared_error')
```

```python
print(ridge_regressor.best_params_)

print(ridge_regressor.best_score_)
```

```
{'alpha': 40}
-1.634240291703168
```

Lasso Regression

```python
from sklearn.linear_model import Lasso
```

```python
from sklearn.model_selection import GridSearchCV
```

```python
lasso=Lasso()
```

```python
parameters = {'alpha':[1, 5, 10, 20, 30, 35, 40, 45, 50, 55, 100]}
```

```python
lasso_regressor=GridSearchCV(lasso,parameters,scoring='neg_mean_squared_error',cv=5)
```

```python
lasso_regressor.fit(x,y)
```

```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [1, 5, 10, 20, 30, 35, 40, 45, 50, 55, 100]})
```

```python
print(lasso_regressor.best_params_)

print(lasso_regressor.best_score_)
```

```
{'alpha': 5}
-0.06758576251202633
```