ridge-regression

December 23, 2023

[20]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

from sklearn.model_selection import GridSearchCV
```

[2]:
```
cars = pd.read_csv('/kaggle/input/car-price/CarPrice_Assignment.csv')
cars.head()
```

[2]:

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

	carbody	drivewheel	enginelocation	wheelbase	…	enginesize	\
0	convertible	rwd	front	88.6	…	130	
1	convertible	rwd	front	88.6	…	130	
2	hatchback	rwd	front	94.5	…	152	
3	sedan	fwd	front	99.8	…	109	
4	sedan	4wd	front	99.4	…	136	

	fuelsystem	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	\
0	mpfi	3.47	2.68	9.0	111	5000	21	
1	mpfi	3.47	2.68	9.0	111	5000	21	
2	mpfi	2.68	3.47	9.0	154	5000	19	

```
3        mpfi       3.19    3.40             10.0        102     5500      24
4        mpfi       3.19    3.40              8.0        115     5500      18

   highwaympg    price
0          27  13495.0
1          27  16500.0
2          26  16500.0
3          30  13950.0
4          22  17450.0

[5 rows x 26 columns]
```

[3]:
```
cars.info()
```

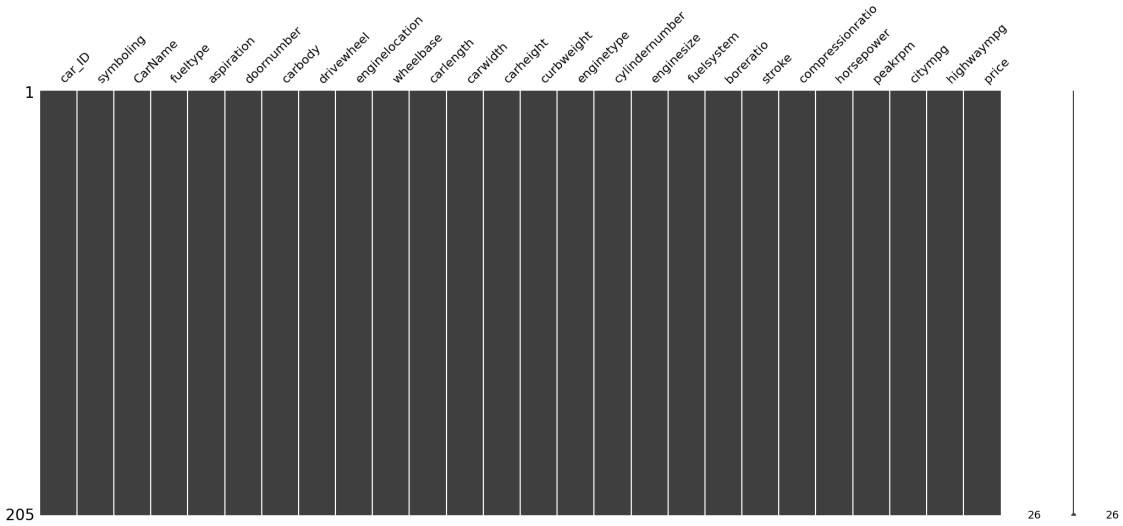
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           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
```

[5]:
```
cars.isnull().sum()
```

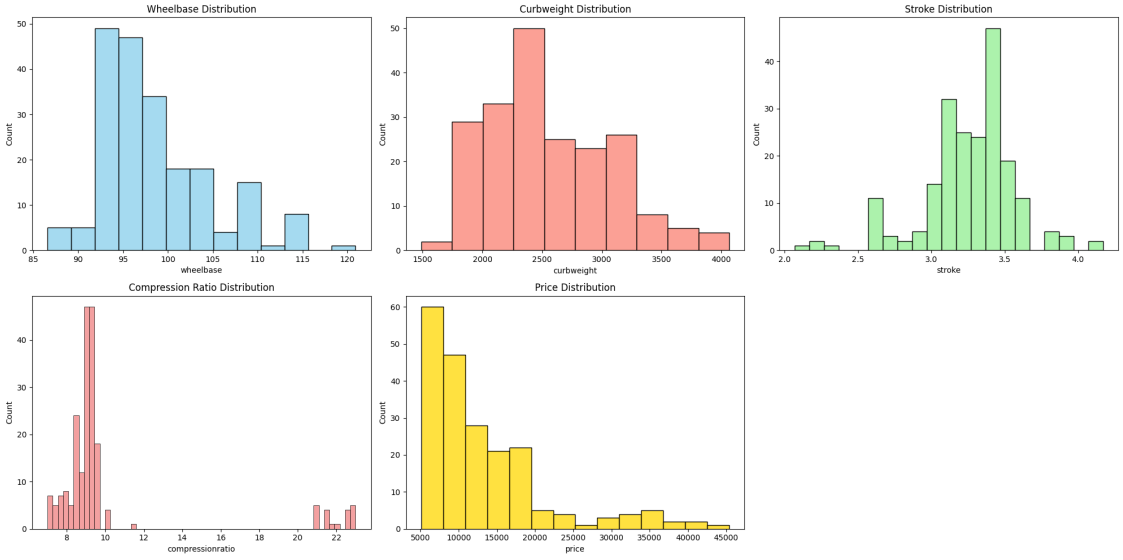
[5]:
```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          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
```

[16]:
```
import missingno
missingno.matrix(cars)
```

[16]:
```
<Axes: >
```



[9]:
```
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(20, 10))

# Plot 1: Wheelbase
sns.histplot(cars['wheelbase'], ax=axes[0, 0], color='skyblue')
axes[0, 0].set_title('Wheelbase Distribution')

# Plot 2: Curbweight
sns.histplot(cars['curbweight'], ax=axes[0, 1], color='salmon')
axes[0, 1].set_title('Curbweight Distribution')

# Plot 3: Stroke
sns.histplot(cars['stroke'], ax=axes[0, 2], color='lightgreen')
axes[0, 2].set_title('Stroke Distribution')

# Plot 4: Compression Ratio
sns.histplot(cars['compressionratio'], ax=axes[1, 0], color='lightcoral')
axes[1, 0].set_title('Compression Ratio Distribution')

# Plot 5: Price
sns.histplot(cars['price'], ax=axes[1, 1], color='gold')
axes[1, 1].set_title('Price Distribution')

# Hide the empty subplot
axes[1, 2].axis('off')

# Adjust layout
plt.tight_layout()
plt.show()
```



1 all numeric (float and int) variables in the dataset

[10]:
```
cars_numeric = cars.select_dtypes(include=['float', 'int'])
cars_numeric.head()
```

[10]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight
0	1	3	88.6	168.8	64.1	48.8	2548
1	2	3	88.6	168.8	64.1	48.8	2548
2	3	1	94.5	171.2	65.5	52.4	2823
3	4	2	99.8	176.6	66.2	54.3	2337
4	5	2	99.4	176.6	66.4	54.3	2824

	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm
0	130	3.47	2.68	9.0	111	5000
1	130	3.47	2.68	9.0	111	5000
2	152	2.68	3.47	9.0	154	5000
3	109	3.19	3.40	10.0	102	5500
4	136	3.19	3.40	8.0	115	5500

	citympg	highwaympg	price
0	21	27	13495.0
1	21	27	16500.0
2	19	26	16500.0
3	24	30	13950.0
4	18	22	17450.0

2 dropping symboling and car_ID

[11]:
```
cars_numeric = cars_numeric.drop(['symboling','car_ID'], axis=1)
cars_numeric.head()
```

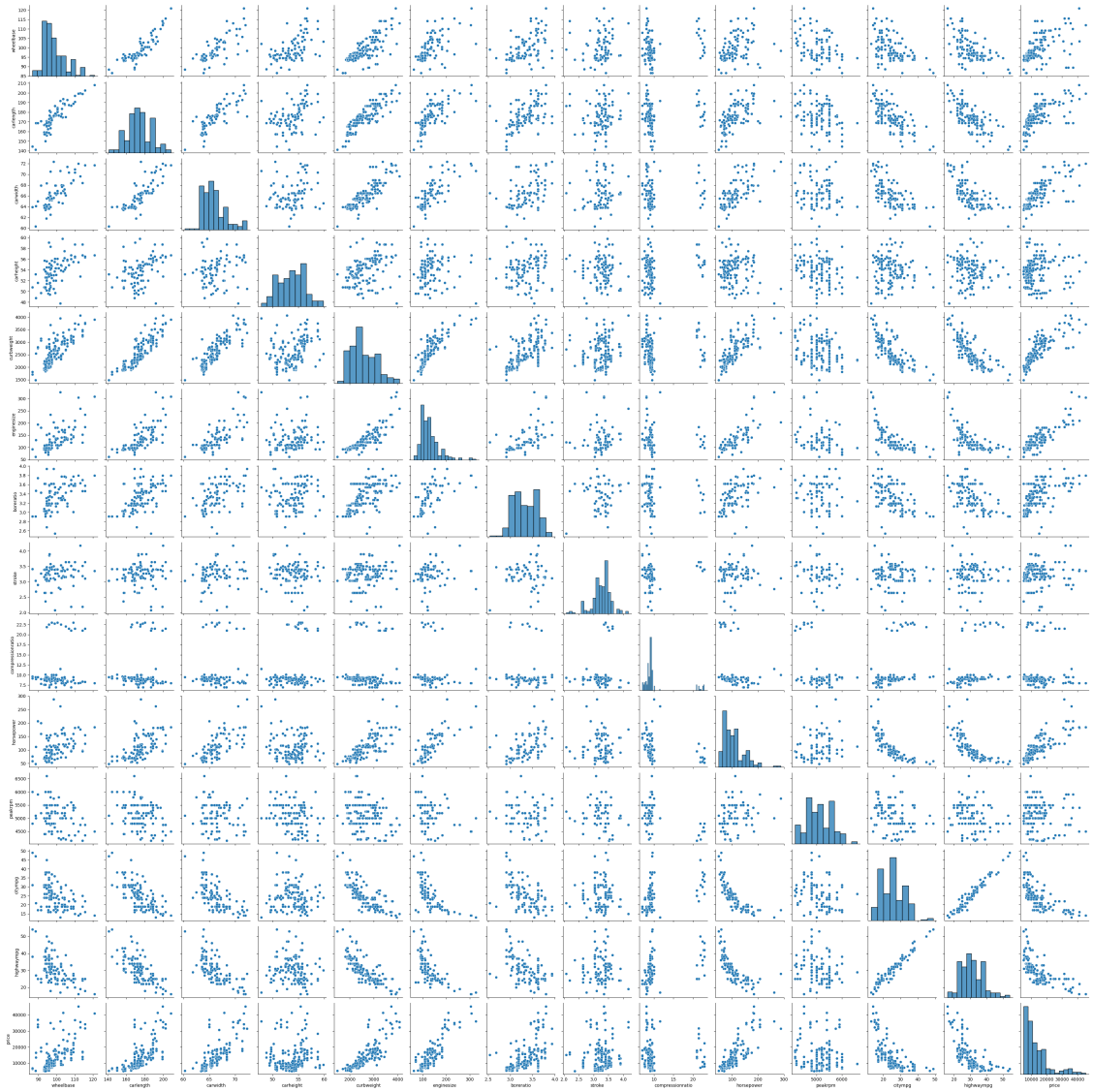
[11]:

	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	\
0	88.6	168.8	64.1	48.8	2548	130	
1	88.6	168.8	64.1	48.8	2548	130	
2	94.5	171.2	65.5	52.4	2823	152	
3	99.8	176.6	66.2	54.3	2337	109	
4	99.4	176.6	66.4	54.3	2824	136	

	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	\
0	3.47	2.68	9.0	111	5000	21	
1	3.47	2.68	9.0	111	5000	21	
2	2.68	3.47	9.0	154	5000	19	
3	3.19	3.40	10.0	102	5500	24	
4	3.19	3.40	8.0	115	5500	18	

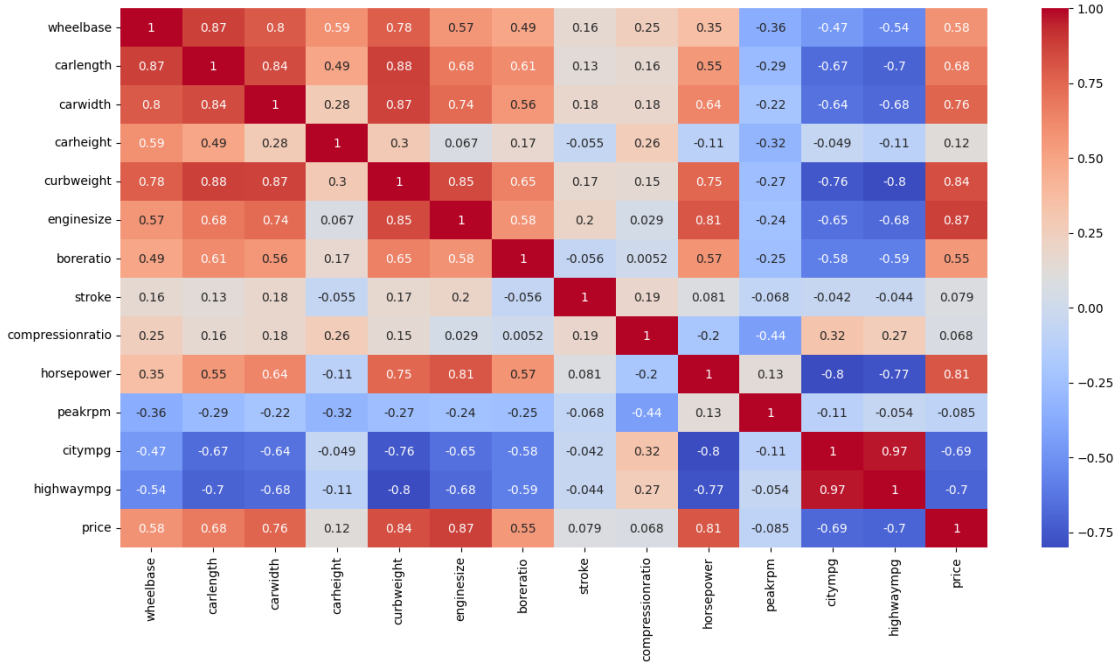
	highwaympg	price
0	27	13495.0
1	27	16500.0
2	26	16500.0
3	30	13950.0
4	22	17450.0

[12]:
```
#paiwise scatter plot
sns.pairplot(cars_numeric)
plt.show()
```



3 plotting correlations on a heatmap

[14]:
```
cor = cars_numeric.corr()
plt.figure(figsize=(16,8))
sns.heatmap(cor, cmap="coolwarm", annot=True)
plt.show()
```



4 Convert categorical variables to numerical using one-hot encoding

[17]:
```
cars = pd.get_dummies(cars, columns=['fueltype', 'aspiration', 'doornumber',␣
↪'carbody', 'drivewheel', 'enginelocation'])
```

5 Feature Selection

[18]:
```
X = cars[['horsepower', 'enginesize', 'highwaympg']]
y = cars['price']
```

[21]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

6 Standardize the features

[22]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

7 Ridge Regression

[23]:
```
alpha = 1.0  # You can adjust the alpha parameter
ridge_model = Ridge(alpha=alpha)
ridge_model.fit(X_train_scaled, y_train)
```

[23]:
```
Ridge()
```

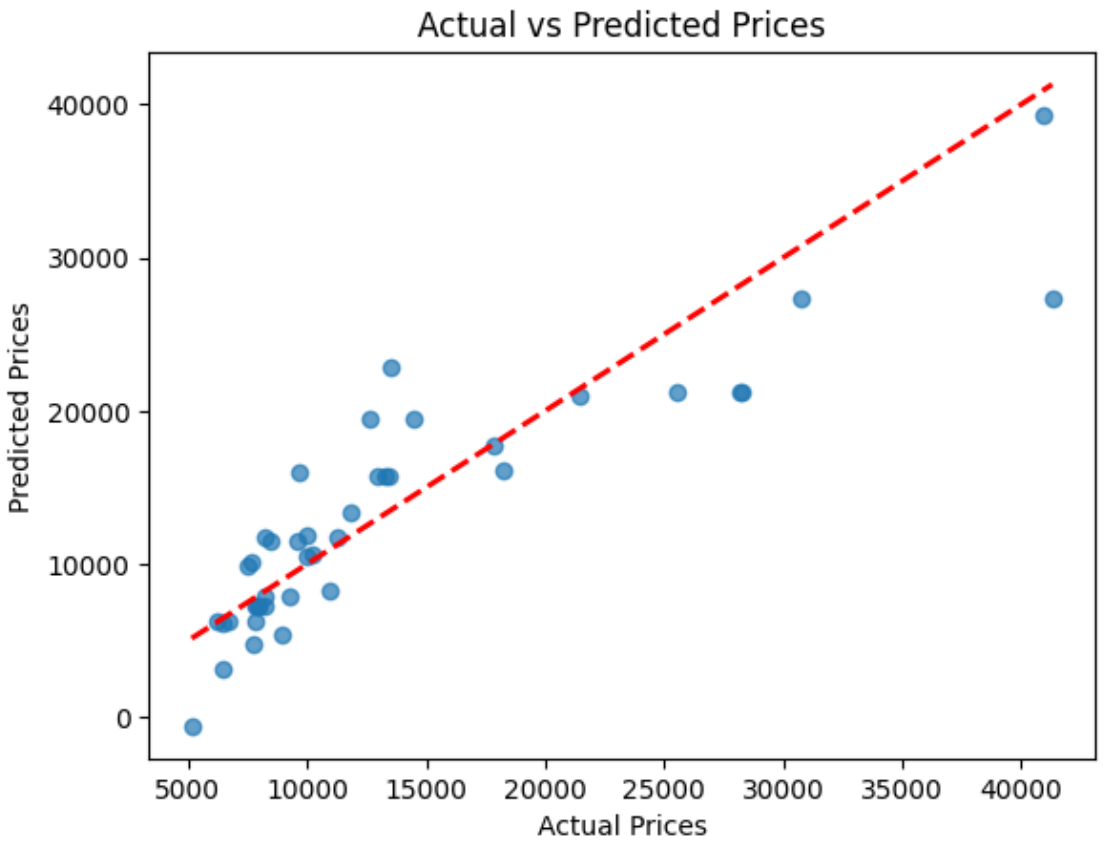
[24]:
```
# Make predictions
y_pred = ridge_model.predict(X_test_scaled)
```

8 Model Evaluation

[25]:
```
from sklearn.metrics import explained_variance_score, max_error,␣
 ↪median_absolute_error

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
explained_variance = explained_variance_score(y_test, y_pred)
max_err = max_error(y_test, y_pred)
median_absolute_err = median_absolute_error(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
print(f"Explained Variance: {explained_variance}")
print(f"Max Error: {max_err}")
print(f"Median Absolute Error: {median_absolute_err}")
```

```
Mean Squared Error: 16423639.119754428
R-squared: 0.7919584163974795
Explained Variance: 0.793025130445292
Max Error: 13984.712702321249
Median Absolute Error: 2295.186013470191
```

[26]:
```
# Plot actual vs predicted prices
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],␣
 ↪linestyle='--', color='red', linewidth=2)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted Prices")
plt.show()
```



[27]:
```
df= pd.DataFrame({'Actual':y_test,'Predictions':y_pred})
df['Predictions']= round(df['Predictions'],2)
df.head()
```

[27]:

	Actual	Predictions
15	30760.000	27330.29
9	17859.167	17784.28
100	9549.000	11549.64
132	11850.000	13336.73
68	28248.000	21266.27