회귀분석(Regression Analysis) - 수치예측

```
import warnings
warnings.filterwarnings('ignore')
```

실습용 데이터 설정

- seaborn 'mpg' Data Set

```
import seaborn as sns

DF = sns.load_dataset('mpg')
```

- pandas DataFrame

```
DF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64
 7   origin        398 non-null    object
 8   name          398 non-null    object
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB
```

```
DF.head(3)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	ori
0	18.0	8	307.0	130.0	3504	12.0	70	
1	15.0	8	350.0	165.0	3693	11.5	70	

I. Simple Linear Regression

- First-Order Function

1) 분석 변수 선택

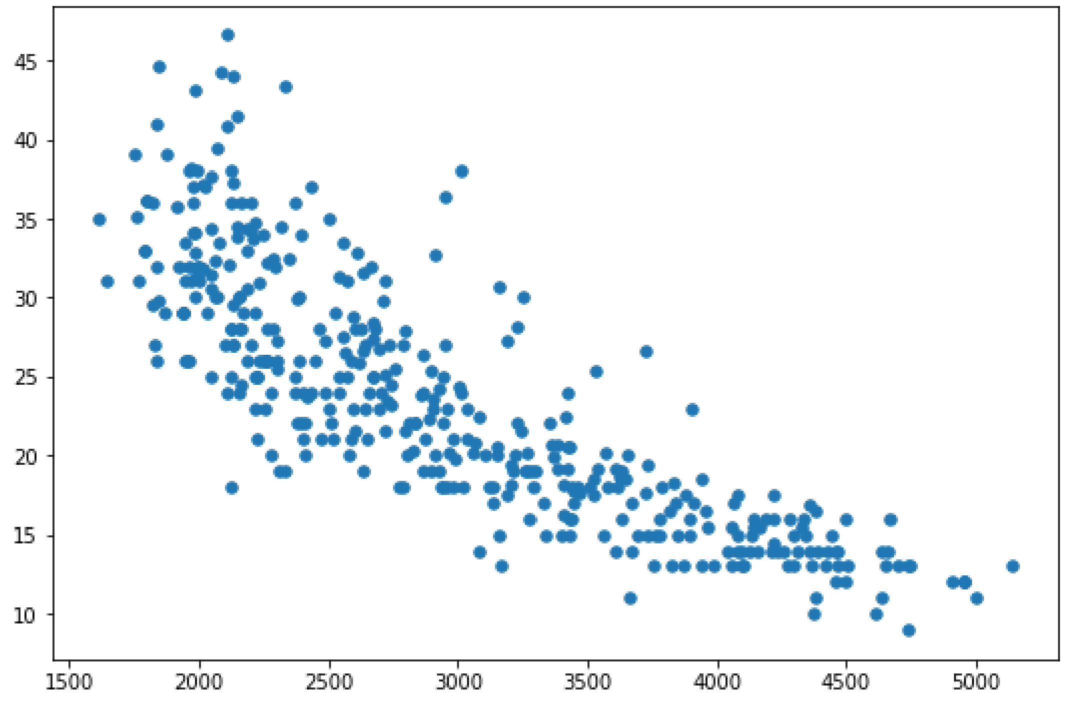
```
DF1 = DF[['mpg', 'cylinders', 'displacement', 'weight']]

DF1.head(3)
```

	mpg	cylinders	displacement	weight
0	18.0	8	307.0	3504
1	15.0	8	350.0	3693
2	18.0	8	318.0	3436

2) 상관관계 그래프

- matplotlib

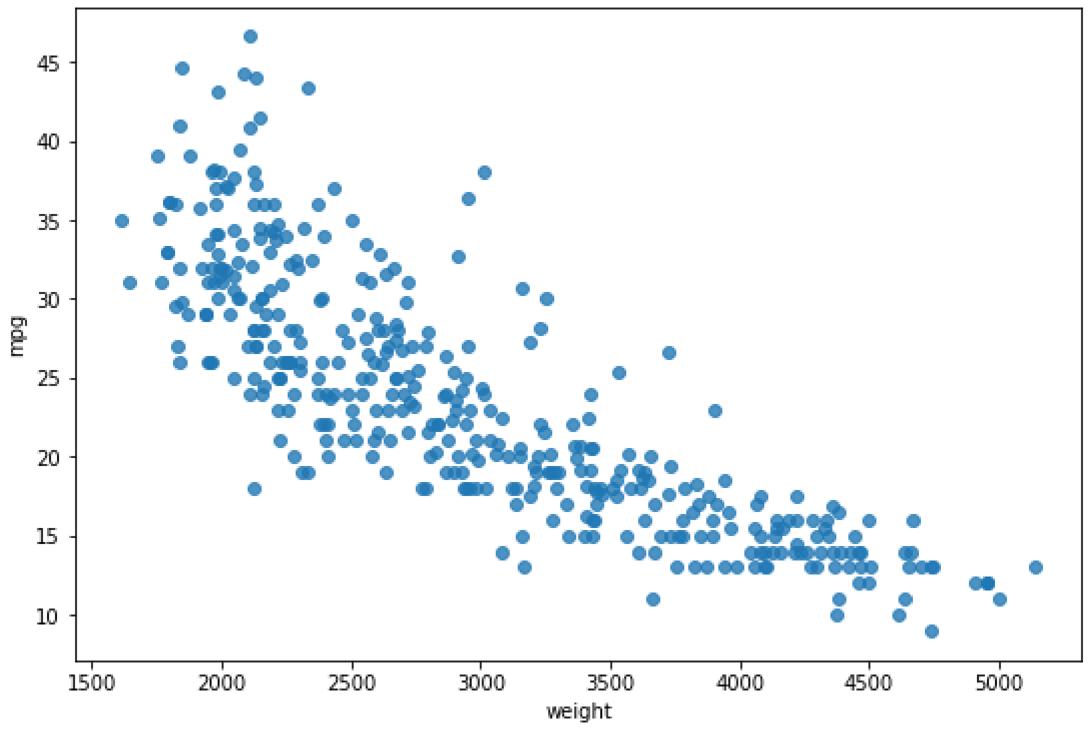
```
import matplotlib.pyplot as plt

plt.figure(figsize = (9, 6))
plt.scatter(x = DF1.weight, y = DF1.mpg, s = 30)
plt.show()
```



- seaborn

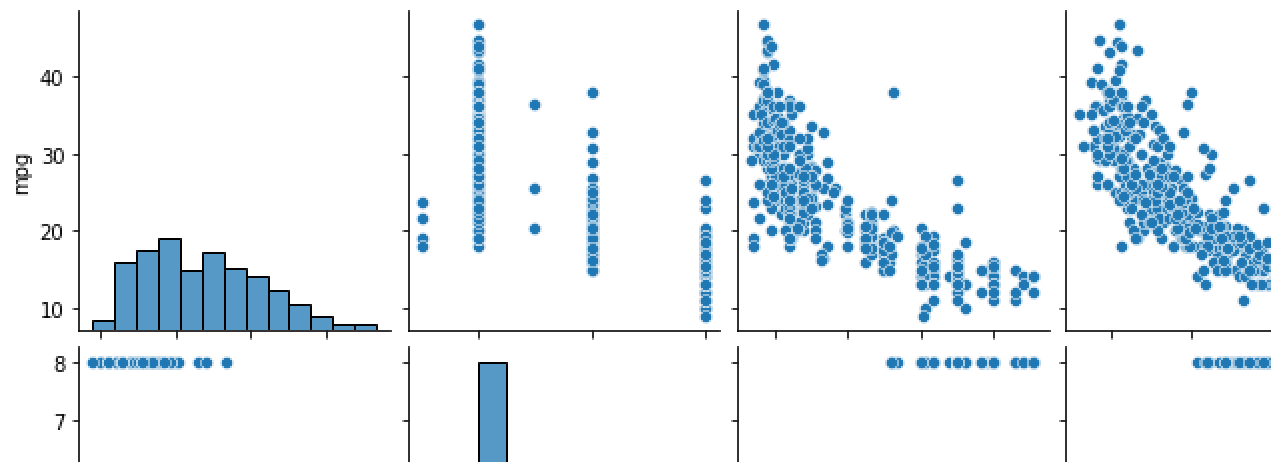
```
fig = plt.figure(figsize = (9, 6))
sns.regplot(x = 'weight', y = 'mpg', data = DF1, fit_reg = False)
```

```
plt.show()
```



- pairplot

```
sns.pairplot(DF1)
plt.show()
```



3) 상관계수(Correlation Coefficient)

- Pearson's r

- mpg vs. weight

```
from scipy import stats

stats.pearsonr(DF1.mpg, DF1.weight)[0]
```

```
-0.831740933244335
```

- mpg vs. displacement

```
from scipy import stats

stats.pearsonr(DF1.mpg, DF1.displacement)[0]
```

```
-0.8042028248058978
```

- mpg vs. cylinders

```
from scipy import stats

stats.pearsonr(DF1.mpg, DF1.cylinders)[0]
```

```
-0.7753962854205542
```

4) Train & Test Split

- 7:3

```
from sklearn.model_selection import train_test_split
```

```
X = DF1[['weight']]
y = DF1['mpg']

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.3,
                                                    random_state = 2045)

print('Train Data : ', X_train.shape, y_train.shape)
print('Test Data : ', X_test.shape, y_test.shape)
```

```
Train Data :  (278, 1) (278,)
Test Data :  (120, 1) (120,)
```

5) 선형회귀 Modeling

- 모델 생성

```
from sklearn.linear_model import LinearRegression

RA = LinearRegression()
RA.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

- Weight 및 Bias

```
print('weight(w) : ', RA.coef_)
print('bias(b) : ', RA.intercept_)
```

```
weight(w) :  [-0.00766168]
bias(b) :  46.28223639092363
```

- 결정계수(R-Sqaure)

```
RA.score(X_test, y_test)
```

```
0.7164499678296495
```

6) 모델 평가

- Mean Squared Error

```
from sklearn.metrics import mean_squared_error
```

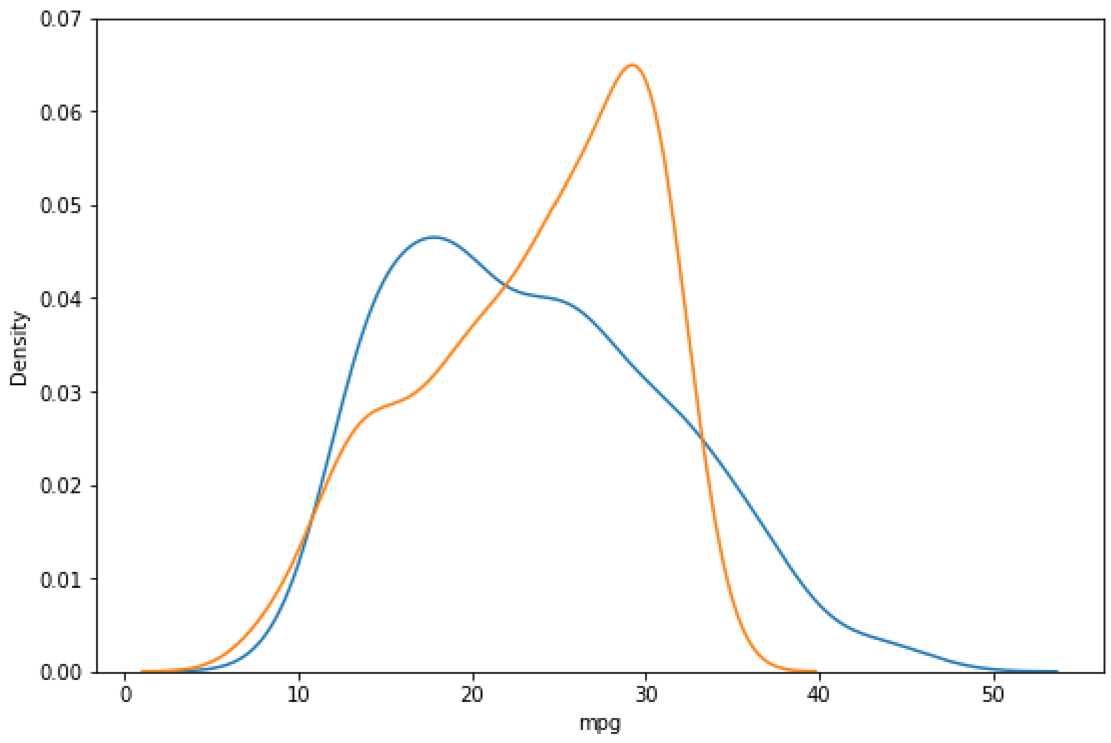
```
y_hat = RA.predict(X_test)

mean_squared_error(y_test, y_hat)
```

```
17.015184477782976
```

7) Visualization

- y vs. y_hat

```
y_hat1 = RA.predict(X)

plt.figure(figsize = (9, 6))
ax1 = sns.distplot(y, hist = False, label = 'y')
ax2 = sns.distplot(y_hat1, hist = False, label='y_hat', ax = ax1)
plt.ylim(0, 0.07)
plt.show()
```



II. Lineare Regression

- High-Order Function

1) 분석 변수 선택

```
DF2 = DF[['mpg', 'cylinders', 'horsepower', 'weight']]
```

```
DF2.head(3)
```

	mpg	cylinders	horsepower	weight
0	18.0	8	130.0	3504
1	15.0	8	165.0	3693
2	18.0	8	150.0	3436

2) Train & Test Split

- 7:3

```
from sklearn.model_selection import train_test_split

X = DF2[['weight']]
y = DF2['mpg']

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.3,
                                                    random_state = 2045)

print('Train Data : ', X_train.shape, y_train.shape)
print('Test Data : ', X_test.shape, y_test.shape)
```

```
Train Data :  (278, 1) (278,)
Test Data :  (120, 1) (120,)
```

3) 선형회귀 Modeling

- 2차 다항식 변환

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree = 2, include_bias = False)
X_train_poly = poly.fit_transform(X_train)

print('변환 전 데이터: ', X_train.shape)
print('2차항 변환 데이터: ', X_train_poly.shape)
```

```
변환 전 데이터:  (278, 1)
2차항 변환 데이터:  (278, 2)
```

- High-Order 모델 생성

```
from sklearn.linear_model import LinearRegression
```

```
NL = LinearRegression()
NL.fit(X_train_poly, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

- Weight 및 Bias

```
# import numpy as np
# np.set_printoptions(suppress = True, precision = 10)

print('weight(w) : ', NL.coef_)
print('bias(b) : ', '%.8f' % NL.intercept_)
```

```
weight(w) :  [-1.75042457e-02  1.53383105e-06]
bias(b) :  60.88867527
```

- 결정계수(R-Sqaure)

```
X_test_poly = poly.fit_transform(X_test)

NL.score(X_test_poly, y_test)
```

```
0.7525521808321769
```

4) 모델 평가

- Mean Squared Error

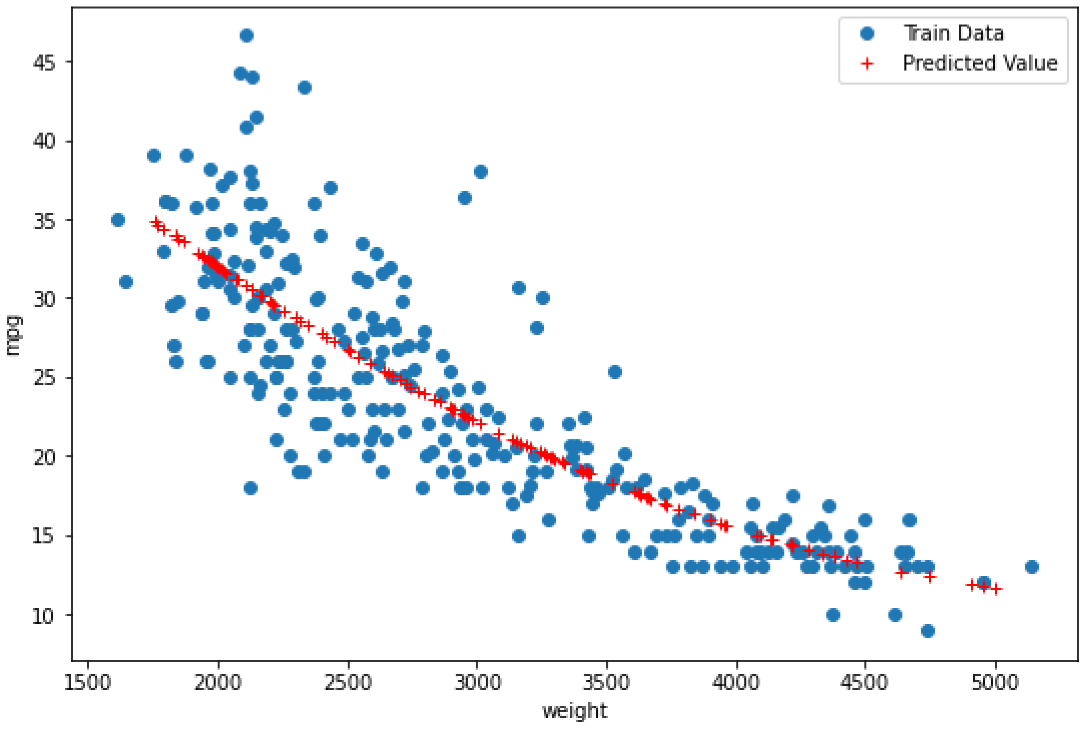
```
from sklearn.metrics import mean_squared_error

X_test_poly = poly.fit_transform(X_test)

mean_squared_error(y_test, NL.predict(X_test_poly))
```

```
14.848773810921921
```

5) Visualization

- High-Order Model

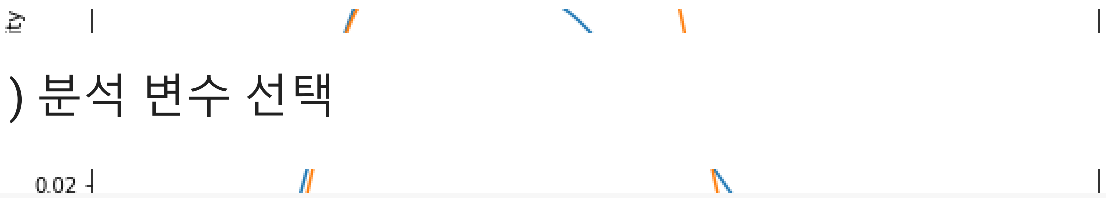
```
y_hat_test = NL.predict(X_test_poly)

plt.figure(figsize=(9, 6))
plt.plot(X_train, y_train, 'o', label = 'Train Data')
plt.plot(X_test, y_hat_test, 'r+', label = 'Predicted Value')
plt.legend(loc='best')
plt.xlabel('weight')
plt.ylabel('mpg')
plt.show()
```



- y vs. y_hat

```
X_ploy = poly.fit_transform(X)
y_hat2 = NL.predict(X_ploy)

plt.figure(figsize = (9, 6))
ax1 = sns.distplot(y, hist=False, label="y")
ax2 = sns.distplot(y_hat2, hist=False, label="y_hat", ax=ax1)
plt.ylim(0, 0.07)
plt.show()
```

III. Multivariate Regression



1) 분석 변수 선택

```
DF3 = DF[['mpg', 'cylinders', 'displacement', 'weight']]

DF3.head(3)
```

	mpg	cylinders	displacement	weight
0	18.0	8	307.0	3504
1	15.0	8	350.0	3693
2	18.0	8	318.0	3436

2) Train &Test Split

- 7:3

```
from sklearn.model_selection import train_test_split

X = DF3[['displacement', 'weight']]
y = DF3['mpg']

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.3,
                                                    random_state = 2045)

print('Train Data : ', X_train.shape, y_train.shape)
print('Test Data : ', X_test.shape, y_test.shape)
```

```
Train Data :  (278, 2) (278,)
Test Data :  (120, 2) (120,)
```

3) 다중회귀 Modeling

- 모델 생성

```
from sklearn.linear_model import LinearRegression
```

```
MR = LinearRegression()
MR.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

- Weight 및 Bias

```
print('weight(w) : ', MR.coef_)
print('bias(b) : ', '%.8f' % MR.intercept_)
```

```
weight(w) :  [-0.01766533 -0.00567273]
bias(b) :  43.74652237
```

- 결정계수(R-Sqaure)

```
MR.score(X_test, y_test)
```

```
0.720971246285159
```

4) 모델 평가

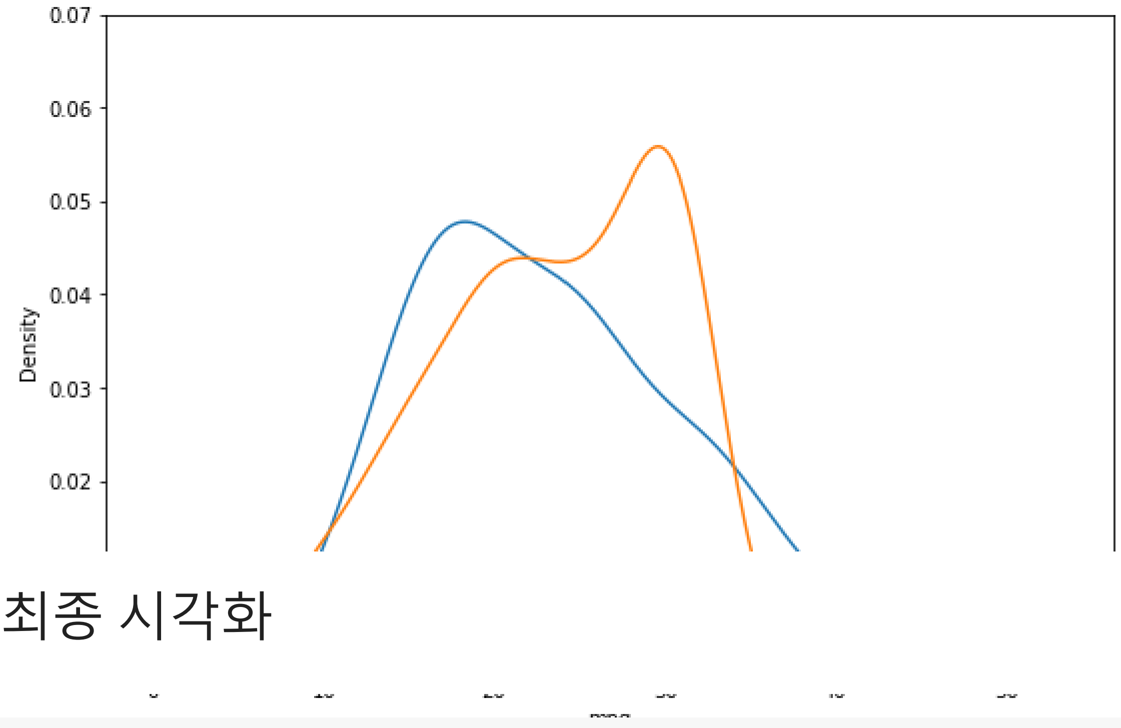
- Mean Squared Error

```
from sklearn.metrics import mean_squared_error

mean_squared_error(y_test, MR.predict(X_test))
```

```
16.743872969214195
```

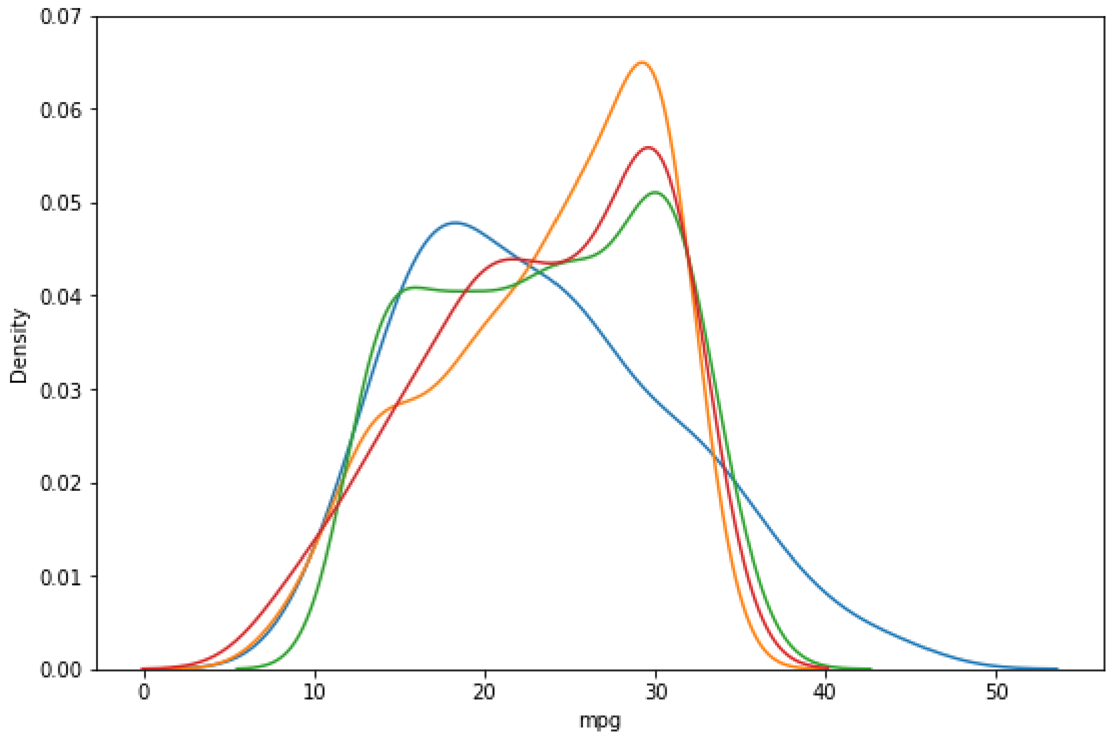
5) Visualization

```
y_hat3 = MR.predict(X_test)

plt.figure(figsize = (9, 6))
ax1 = sns.distplot(y_test, hist = False, label = 'y_test')
ax2 = sns.distplot(y_hat3, hist = False, label='y_hat', ax = ax1)
plt.ylim(0, 0.07)
plt.show()
```



IV. 최종 시각화

```
y_hat3 = MR.predict(X_test)

plt.figure(figsize = (9, 6))
ax1 = sns.distplot(y_test, hist = False, label = 'y_test')
ax2 = sns.distplot(y_hat1, hist = False, label='y_hat', ax = ax1)
ax3 = sns.distplot(y_hat2, hist = False, label='y_hat', ax = ax1)
ax4 = sns.distplot(y_hat3, hist = False, label='y_hat', ax = ax1)
plt.ylim(0, 0.07)
plt.show()
```



#

#

#

The End

#

#

#