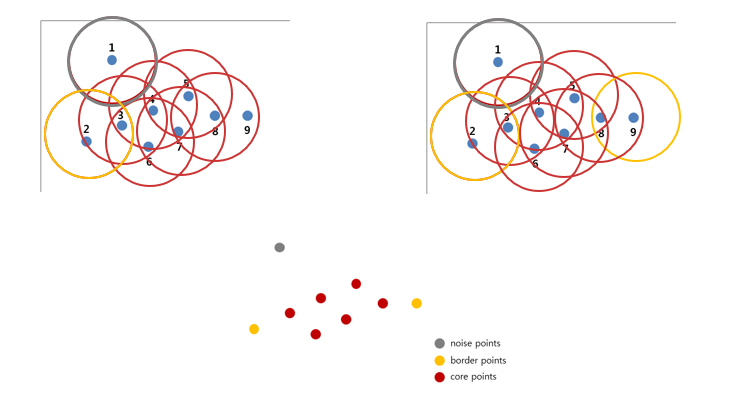
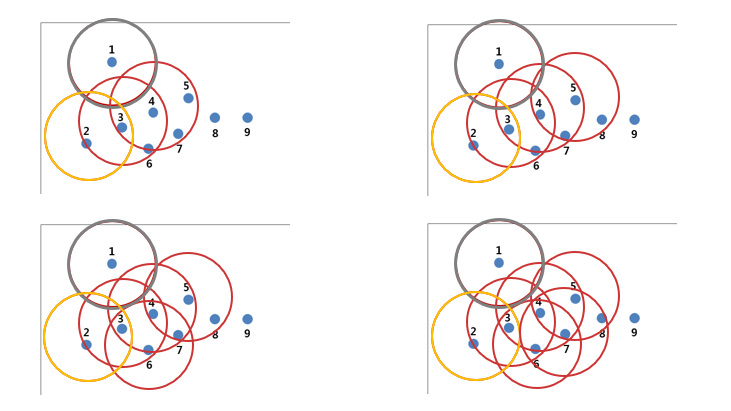
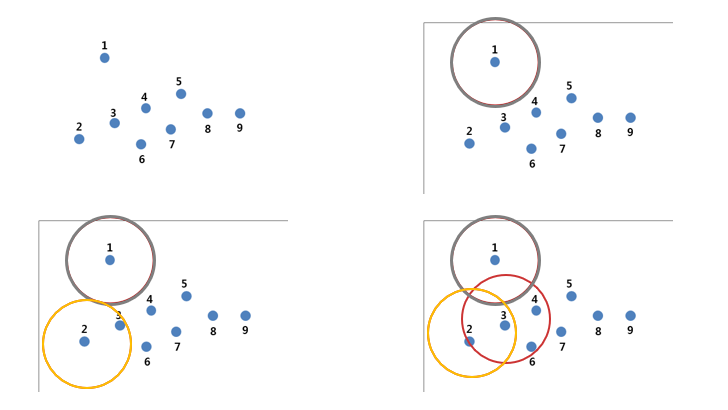
**HW4**

201635840 이정명

Gray -> noise, Yellow -> border, Red -> core

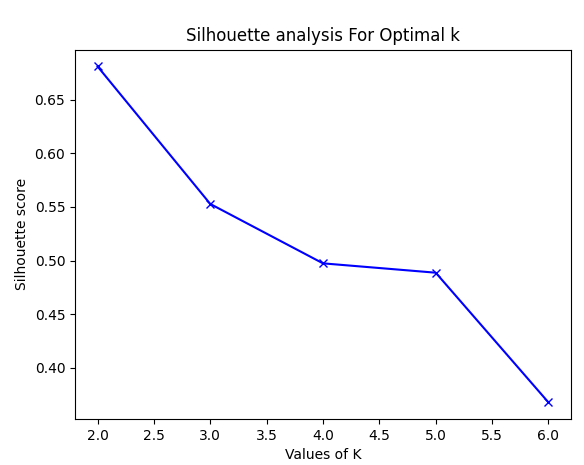


Code(Clustering-quality)

import pandas as pd  
from matplotlib import pyplot as plt  
from sklearn import datasets  
from sklearn.cluster import KMeans  
from sklearn.metrics import silhouette\_score  
  
# Load IRIS dataset  
iris = datasets.load\_iris()  
  
data = pd.DataFrame(data=iris.data, columns=iris.feature\_names)  
df = pd.DataFrame(data)  
  
# Instantiate the KMeans models  
km = KMeans(n\_clusters=4, random\_state=42)  
  
# Fit the KMeans model  
km.fit\_predict(data)  
  
# Calculate Silhoutte Score  
score = silhouette\_score(data, km.labels\_, metric='euclidean')  
  
# Print the score  
print('Silhouette Score(4-Means Clustering): %.3f' % score)  
  
  
range\_n\_clusters = [2, 3, 4, 5, 6]  
silhouette\_avg = []  
for num\_clusters in range\_n\_clusters:  
 # initialize kmeans  
 kmeans = KMeans(n\_clusters=num\_clusters)  
 kmeans.fit(df)  
 cluster\_labels = kmeans.labels\_  
 # silhouette score  
 silhouette\_avg.append(silhouette\_score(df, cluster\_labels))  
  
plt.plot(range\_n\_clusters, silhouette\_avg, "bx-")  
plt.xlabel("Values of K")  
plt.ylabel("Silhouette score")  
plt.title("Silhouette analysis For Optimal k")  
plt.show()

Result





Code(Regularization)

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
  
data = pd.read\_csv('C:/Users/ASUS/PycharmProjects/pythonProject2/sample.csv')  
# print(data.head())  
  
data = data.dropna(axis=0)  
data = data.drop(columns='MODEL')  
  
# print(data.head())  
  
# write a helper function ‘scatter\_plot’  
# def scatter\_plot(feature, target):  
# plt.figure(figsize=(16, 8))  
# plt.scatter(  
# data[feature],  
# data[target],  
# c='black'  
# )  
# plt.xlabel(feature)  
# plt.ylabel("Emissions")  
# plt.show()  
#  
# scatter\_plot('ENGINE', 'EMISSIONS');  
# scatter\_plot('CYLINDERS', 'EMISSIONS');  
# scatter\_plot('FUEL', 'EMISSIONS');  
  
  
# Linear Regression  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.linear\_model import LinearRegression  
x5 = data.drop(['EMISSIONS'], axis=1)  
y = data['EMISSIONS'].values.reshape(-1, 1)  
lin\_reg = LinearRegression()  
MSE5 = cross\_val\_score(lin\_reg, x5, y,  
 scoring='neg\_mean\_squared\_error', cv=5)  
mean\_MSE = np.mean(MSE5)  
print(mean\_MSE)  
  
  
# Ridge Regression  
from sklearn.model\_selection import GridSearchCV  
from sklearn.linear\_model import Ridge  
ridge =Ridge()  
parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,  
1e-2, 1, 5, 10, 20]}  
ridge\_regressor = GridSearchCV(ridge, parameters,  
 scoring='neg\_mean\_squared\_error', cv=5)  
ridge\_regressor.fit(x5, y)  
  
print(ridge\_regressor.best\_params\_)  
print(ridge\_regressor.best\_score\_)  
  
  
# LASSO Regression  
from sklearn.linear\_model import Lasso  
lasso = Lasso()  
parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,  
1e-2, 1, 5, 10, 20]}  
lasso\_regressor = GridSearchCV(lasso, parameters,  
 scoring='neg\_mean\_squared\_error', cv=5)  
lasso\_regressor.fit(x5, y)  
  
print(lasso\_regressor.best\_params\_)  
print(lasso\_regressor.best\_score\_)

# Elastic Net  
from sklearn.linear\_model import ElasticNet  
elastic = ElasticNet()  
parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,  
1e-2, 1, 5, 10, 20]}  
elastic\_regressor = GridSearchCV(elastic, parameters,  
 scoring='neg\_mean\_squared\_error', cv=5)  
elastic\_regressor.fit(x5, y)  
  
print(elastic\_regressor.best\_params\_)  
print(elastic\_regressor.best\_score\_)

Result







