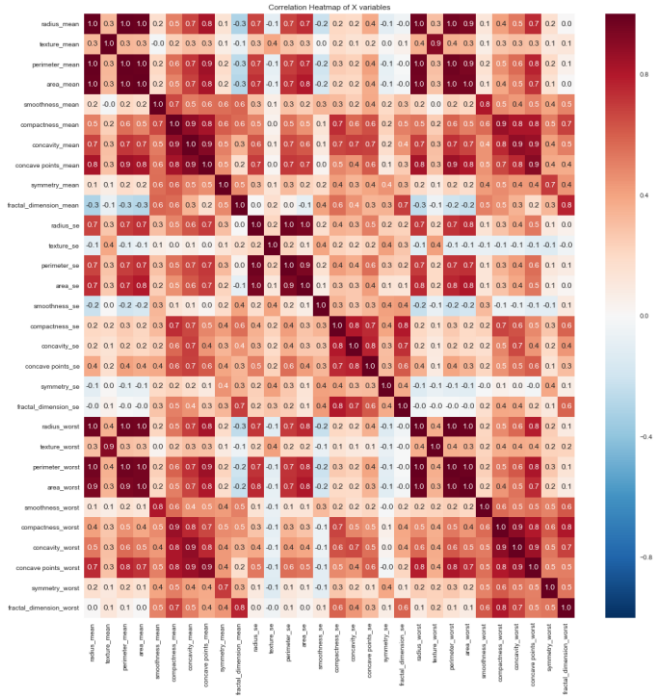
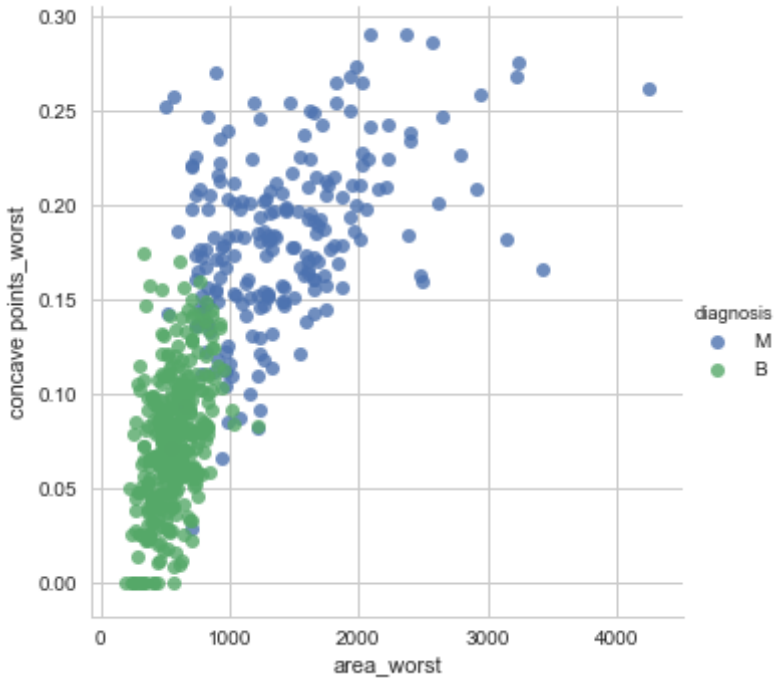
PCA / Clustering

PCA 예제

데이터 설명

- **'Breast Cancer Wisconsin'** data
  - 569개의 관측치가 2개의 범주 중 하나에 속함 (B:M = 357:212)
  - 30개의 실수형 변수 존재





https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

PCA 실습

- 모듈 불러오기

```
# Import modules
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

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

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
```

PCA 실습

- 데이터 불러오기

```
# Import data
data = pd.read_csv("BreastCancerWisconsin.csv")
print("- Data has {} rows and {} columns.".format(*data.shape))
print("- Column names: ", list(data.columns))
```

- X와 y로 나누기 및 Train / Test set 분류

```
# Split dataset into X and y
X = data.drop(['diagnosis'], axis=1)
X = X.iloc[:, :10]
y = data['diagnosis']

# Split trian / test data set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

PCA 실습

- Standard Scaling 수행

  → Test data는 미래의 데이터로 간주함

  → Train data로만 mean / stddev를 계산하고 이를 test data 에 적용

```
# Standardize data onto unit scale (mean=0 and variance=1)
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

- PCA 모델 fitting

```
# Perform PCA
pca = PCA(n_components=None)
pca.fit(X_train)

Z_train = pca.transform(X_train)
print("- Shape of transformed data: ", Z_train.shape)

Z_test = pca.transform(X_test)
print("- Shape of transformed data: ", Z_test.shape)
```

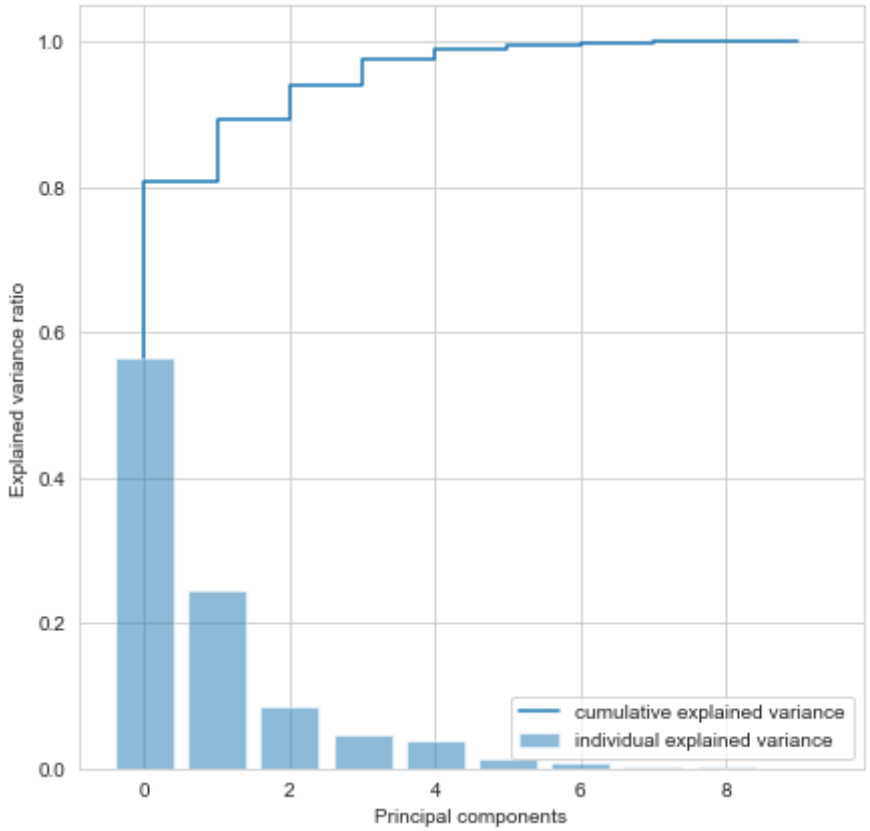
PCA 실습

- PC들의 explained variance ratio 계산 및 시각화

```
# Explained variance ratio of principal components
num_components = pca.n_components_
exp_var = pca.explained_variance_ratio_
cum_exp_var = np.cumsum(exp_var)


# Plot explained variance ratio and cumulative sums
plt.figure(num=1, figsize=(7, 7))
plt.bar(range(num_components), exp_var, alpha=0.5, label='individual explained variance')
plt.step(range(num_components), cum_exp_var, label='cumulative explained variance')
plt.xlabel('Principal components')
plt.ylabel('Explained variance ratio')
plt.legend(loc='best')
plt.show()
```

PCA 실습

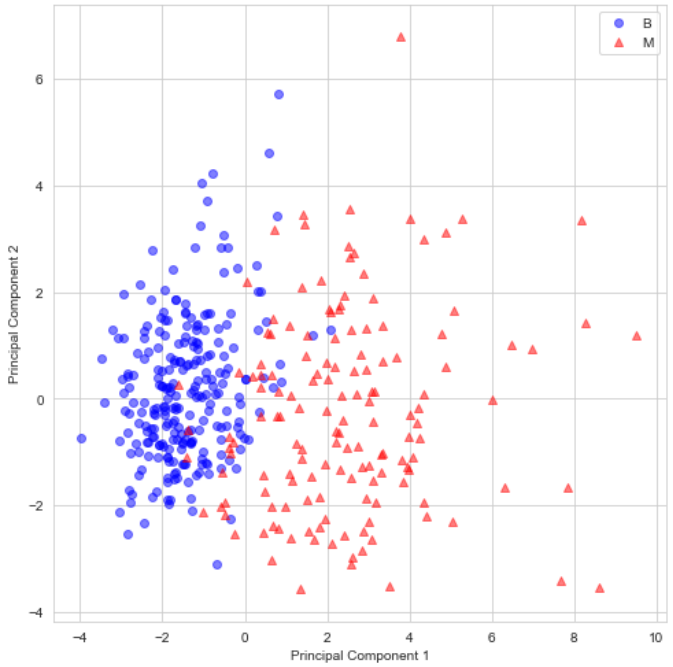
- PC들의 explained variance ratio 계산 및 시각화



PCA 실습

- PC1, PC2를 사용하여 2D scatter plot 시각화

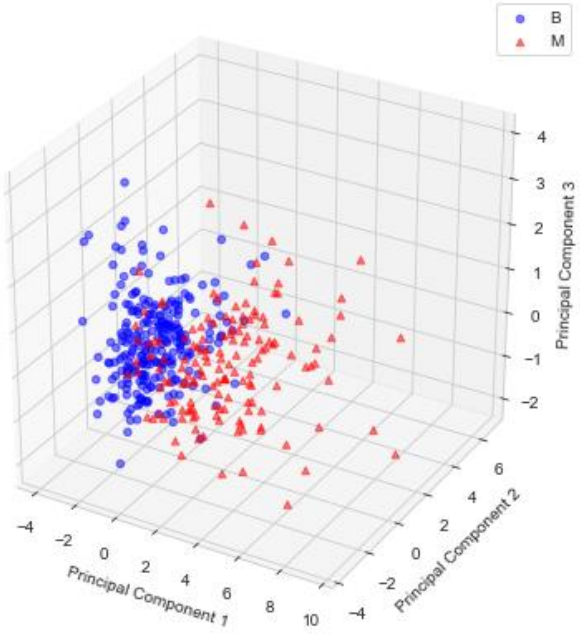
```
# Plot the transformed data (Z) with 2 PCs
plt.figure(num=2, figsize=(7, 7))
for label, color, marker in zip(('B', 'M'), ('blue', 'red'), ('o', '^')):
    plt.scatter(Z_train[y_train == label, 0], Z_train[y_train == label, 1],
                label=label, color=color, marker=marker, alpha=0.5)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```



PCA 실습

- PC1, PC2, PC3를 사용하여 3D scatter plot 시각화

```
# Plot the transformed data (Z) with 3 PCs
fig = plt.figure(num=3, figsize=(7, 7))
ax = fig.add_subplot(111, projection='3d')
for label, color, marker in zip(('B', 'M'), ('blue', 'red'), ('o', '^')):
    ax.scatter(Z_train[y_train == label, 0], Z_train[y_train == label, 1],
               Z_train[y_train == label, 2], label=label, color=color,
               marker=marker, alpha=0.5)
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')
ax.legend(loc='best')
plt.show(fig)
```



PCA 실습 + PCR

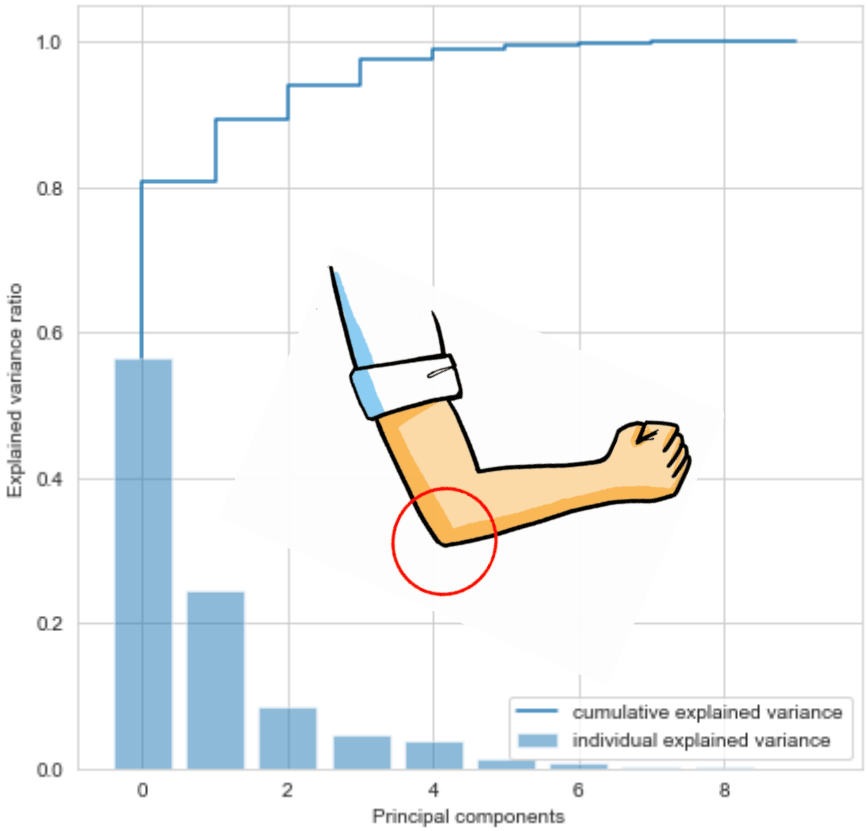
- Logistic Regression 적용을 위한 Label Encoding

  → [“B” ,”M”] >> [“0”, “1”]

```
print("- sample of 'y': ", y_train[:5])

le = LabelEncoder()
le.fit(y_train)

print(" - encoding label of 'Y': ", le.classes_)
print(" - encoding label of 'Y': [0, 1]")

Y_train = le.transform(y_train)
Y_test = le.transform(y_test)
```

PCA 실습 + PCR

- PC들의 explained variance ratio 계산 및 시각화



PCA 실습 + PCR

- Logistic Regression 모델 구축 및 예측 수행 (PC1, PC2만 선택)

```
## Select sub-features
Z_sub_train = pd.DataFrame(Z_train[:,:2])
Z_sub_test = pd.DataFrame(Z_test[:,:2])

## Build LR model
log_Z = LogisticRegression()
log_Z.fit(Z_sub_train, Y_train)

## Predict & calculate score
log_Z.score(Z_sub_test, Y_test)
pred_Z = log_Z.predict(Z_sub_test)
confusion_matrix(Y_test, pred_Z)
```

```
In [187]: log_Z.score(Z_sub_test, Y_test)
Out[187]: 0.9298245614035088

In [188]: confusion_matrix(Y_test, pred_Z)
Out[188]:
array([[101,   3],
       [  9,  58]], dtype=int64)
```

PCA 실습 + PCR

- Logistic Regression 모델 구축 및 예측 수행 (원본 데이터)

```
## Build model and prediction (Original Data)
log_ori = LogisticRegression()
log_ori.fit(X_train, Y_train)
log_ori.score(X_test, Y_test)
pred_ori = log_ori.predict(X_test)
confusion_matrix(Y_test, pred_ori)
```

```
In [189]: log_ori.score(X_test, Y_test)
Out[189]: 0.9415204678362573

In [190]: confusion_matrix(Y_test, pred_ori)
Out[190]:
array([[100,   4],
       [  6,  61]], dtype=int64)
```

PCA + PCR 실습

데이터 설명

- **주어진 데이터를 사용하여 PCA plot 및 Logistic Regression을 활용한 PCR 모델 구축**

- **실습 데이터 1: P2P 대출 상환 데이터**
  - P2P 대출 관련 데이터
  - 22개의 변수 / 28,784개 관측치로 구성

- **실습 데이터 2: 대출 연체 데이터**
  - 대출 연체 예측을 위한 데이터 (성별, 대출 금액 등)
  - 27개의 변수 / 43,386개의 관측치로 구성

- **각 폴더내의 Description.txt 파일 참조**

Hierarchical Clusterin 예제

Hierarchical Clustering 실습

- 모듈 불러오기

```
# Import modules
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.decomposition import PCA
```

Hierarchical Clustering 실습

- 데이터 불러오기

```
# Import data
data = pd.read_csv("./0_Data/BreastCancerWisconsin.csv")
print("- Data has {} rows and {} columns.".format(*data.shape))
print("- Column names: ", list(data.columns))
```

- X와 y로 나누기

```
# For clustering, we will only be using the X variables
X = data.drop(['diagnosis'], axis=1)
X = X.iloc[:, :10]
y = data['diagnosis']
```

- 데이터 정규화

```
# Standardize dataset columnwise, to have zero mean and unit variance
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

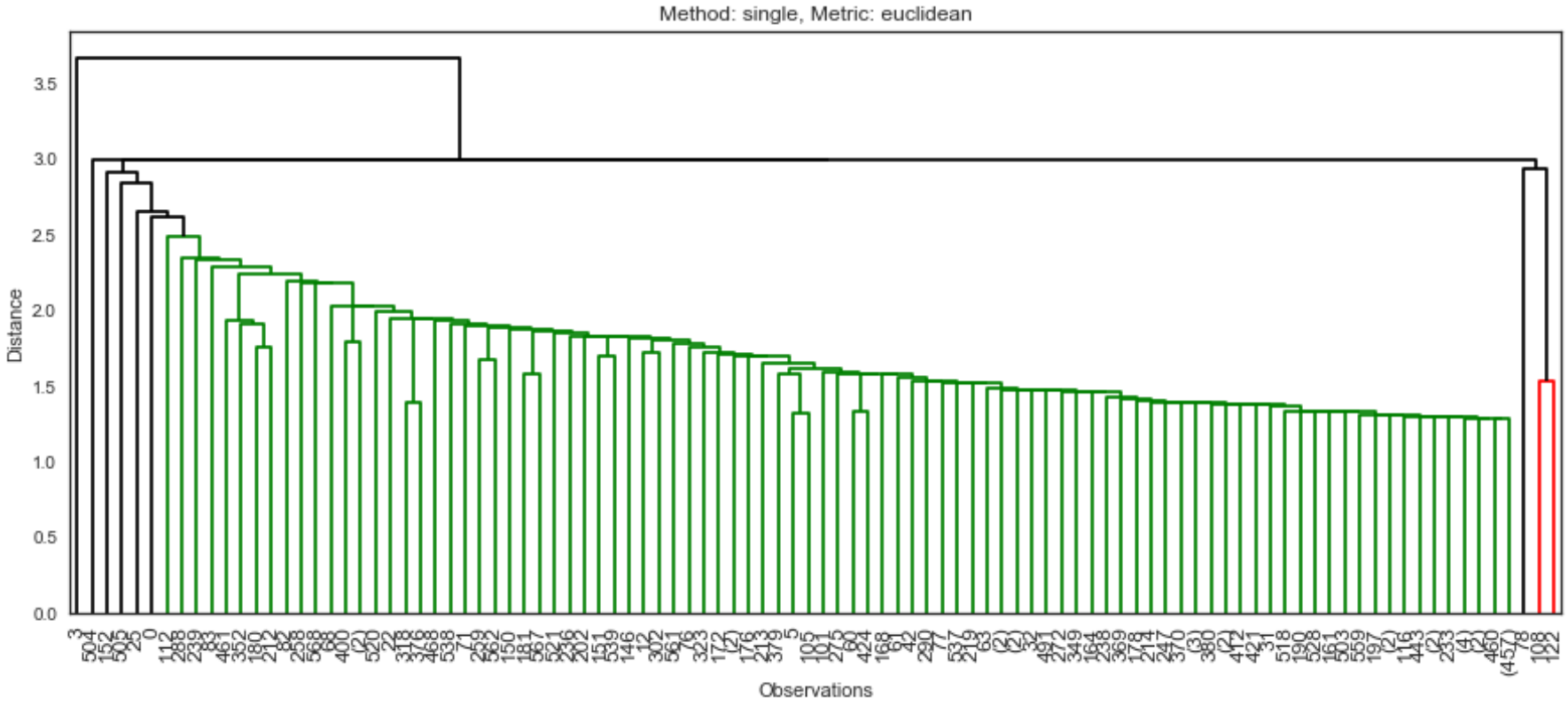
Hierarchical Clustering 실습

- Hierarchical Clustering 모델 학습 및 시각화 (다른 method와 metric 사용해보기)

```
# Perform hierarchical clustering
method = 'ward'
metric = 'euclidean'
D = linkage(X, method=method, metric=metric)


# Draw dendrogram
fig, ax = plt.subplots(1, 1, figsize=(15, 6))
dendrogram(Z=D,
           p=100,
           truncate_mode='lastp',
           orientation='top',
           show_leaf_counts=True,
           no_labels=False,
           leaf_font_size=12.,
           leaf_rotation=90.,
           ax=ax,
           above_threshold_color='k')
ax.set_xlabel('Observations')
ax.set_ylabel('Distance')
ax.set_title('Method: {}, Metric: {}'.format(method, metric))
plt.show(fig)
```

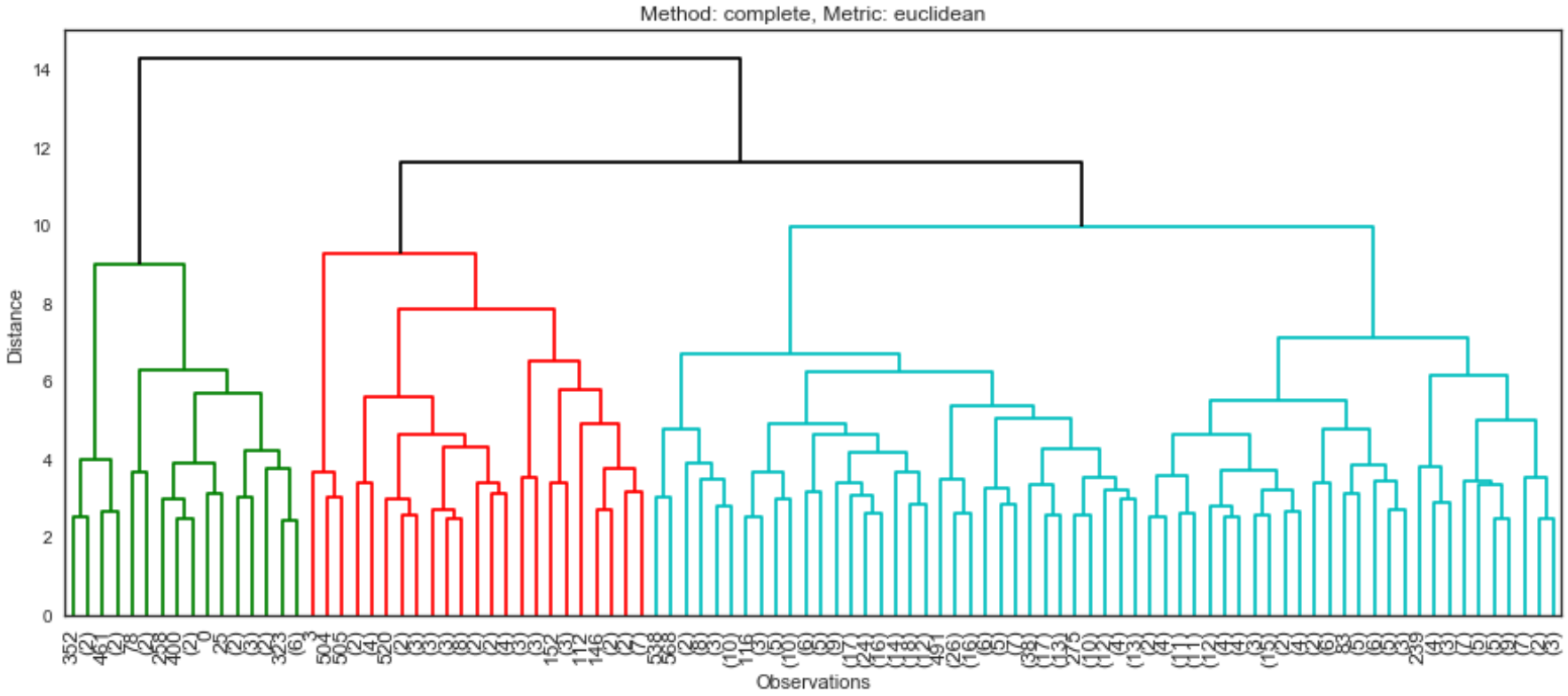
Hierarchical Clustering 실습

- Hierarchical Clustering 모델 학습 및 시각화 결과
  - method='single', metric='euclidean'



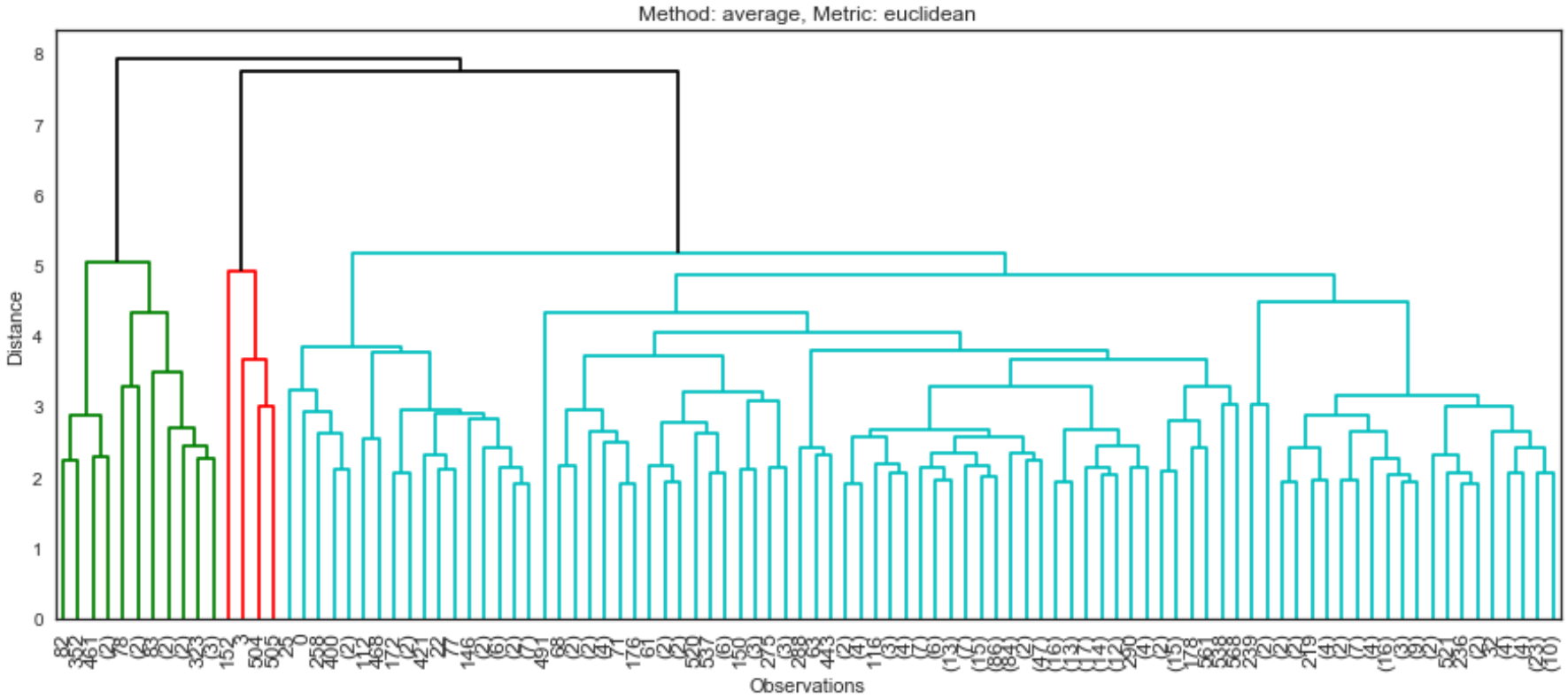
Hierarchical Clustering 실습

- Hierarchical Clustering 모델 학습 및 시각화 결과
  - method='complete', metric='euclidean'



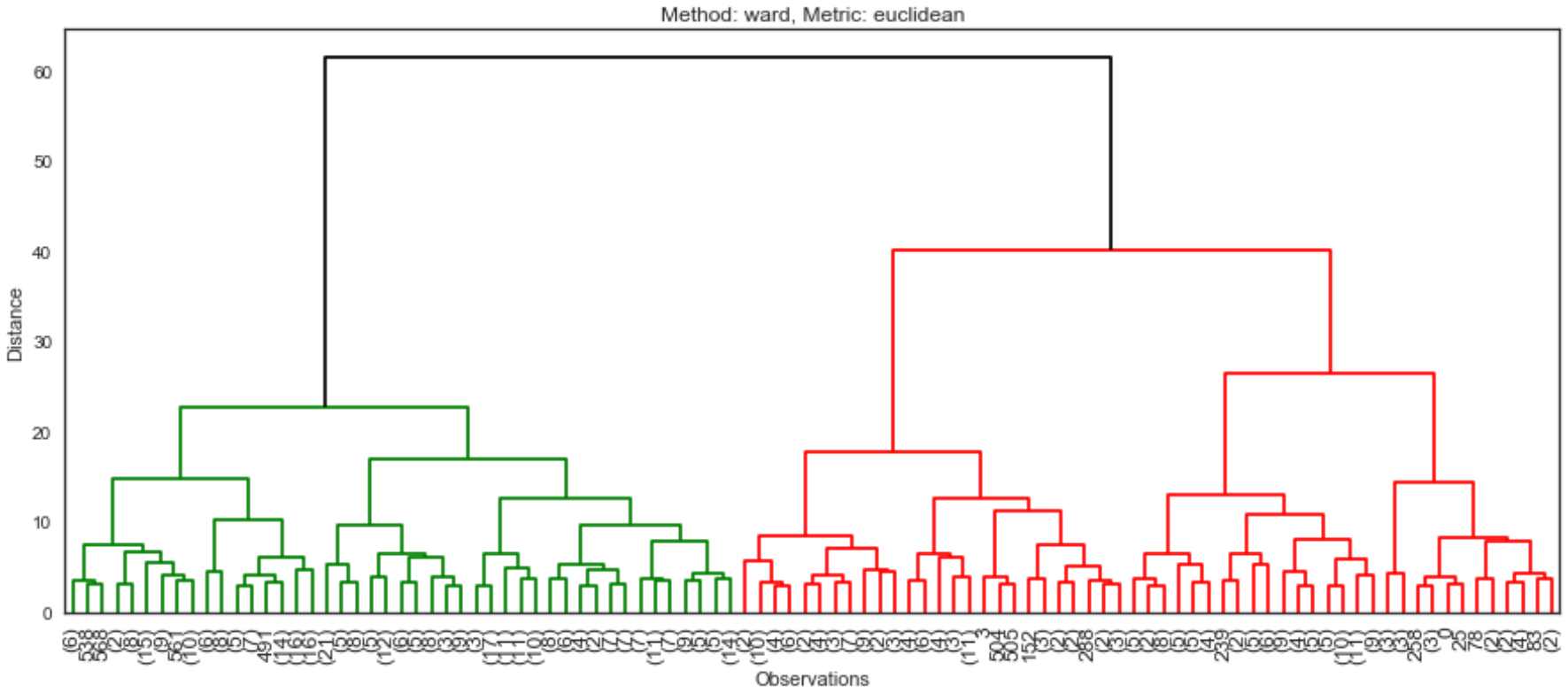
Hierarchical Clustering 실습

- Hierarchical Clustering 모델 학습 및 시각화 결과
  - method='average', metric='euclidean'



Hierarchical Clustering 실습

- Hierarchical Clustering 모델 학습 및 시각화 결과
  - method='ward', metric='euclidean'



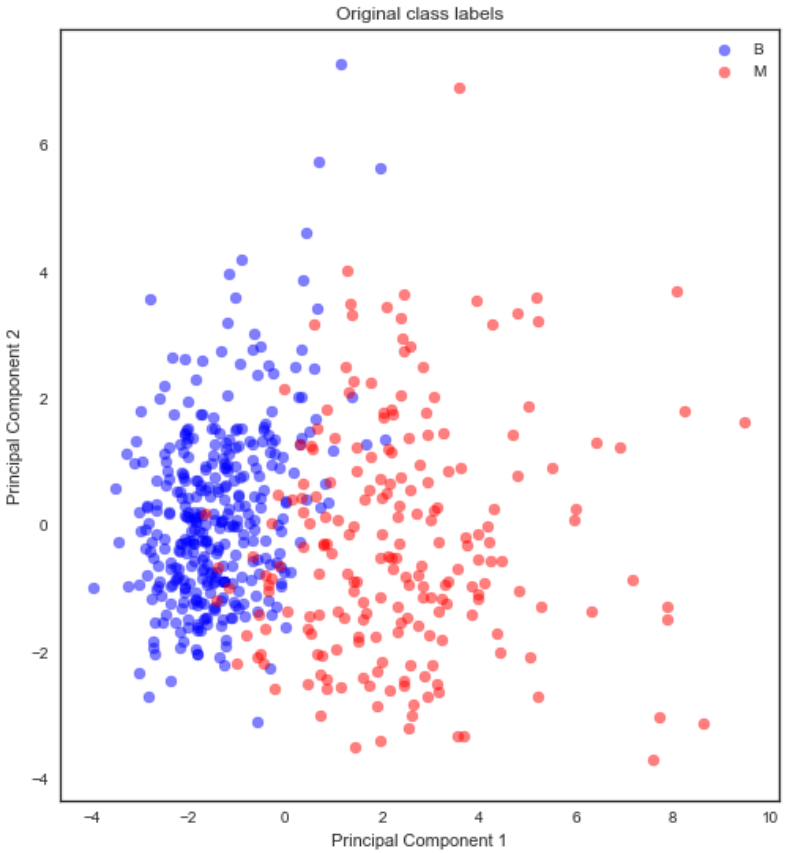
Hierarchical Clustering 실습

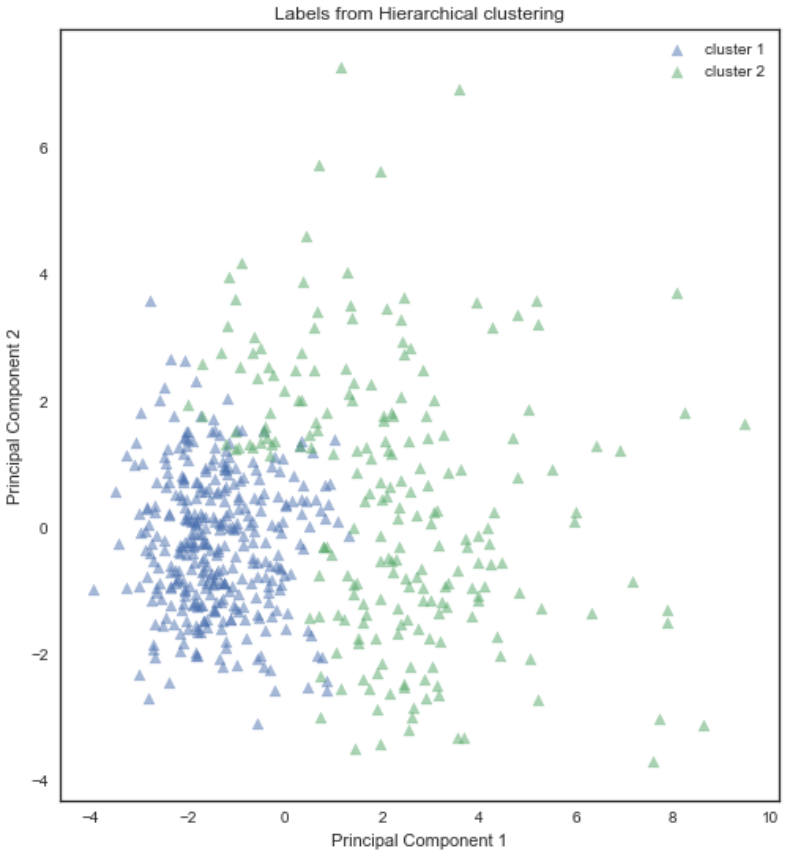
- Cluster 개수를 임의 지정 후, PCA로 학습한 PC1, PC2를 사용하여 결과 시각화

```
# Get cluster labels, by predefining the number of clusters
num_clusters = 2
label_hc = fcluster(D, t=num_clusters, criterion='maxclust')

# Plot on 2D space, using PCA components
# Perform PCA
pca = PCA(n_components=2)
Z = pca.fit_transform(X)

# Plot the transformed data (Z) with 2 PCs
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(18, 9))
axes = axes.ravel()
for label, color in zip(('B', 'M'), ('blue', 'red')):
    axes[0].scatter(Z[y == label, 0], Z[y == label, 1],
                    label=label, color=color, marker='o', alpha=0.5)
    axes[0].set_xlabel('Principal Component 1')
    axes[0].set_ylabel('Principal Component 2')
    axes[0].legend(loc='best')
    axes[0].set_title('Original class labels')
for i in range(num_clusters):
    axes[1].scatter(Z[label_hc == i + 1, 0], Z[label_hc == i + 1, 1],
                    label='cluster {}'.format(i + 1), marker='^', alpha=0.5)
    axes[1].set_xlabel('Principal Component 1')
    axes[1].set_ylabel('Principal Component 2')
    axes[1].legend(loc='best')
    axes[1].set_title('Labels from Hierarchical clustering')
plt.show(fig)
```

Hierarchical Clustering 실습

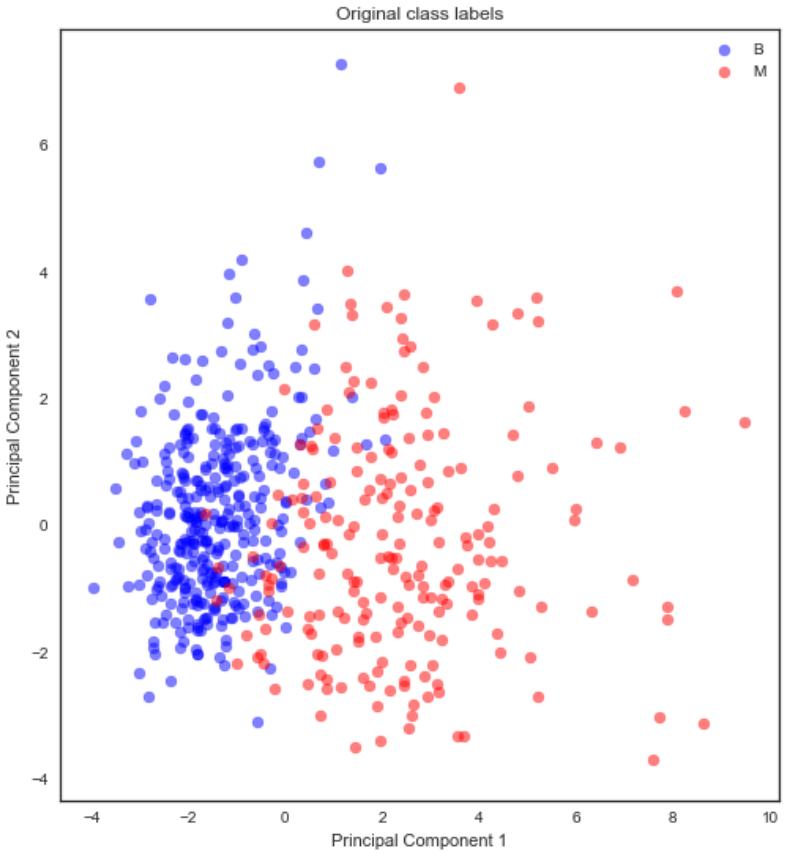
- Cluster 개수 = 2 (method='ward', metric='euclidean')

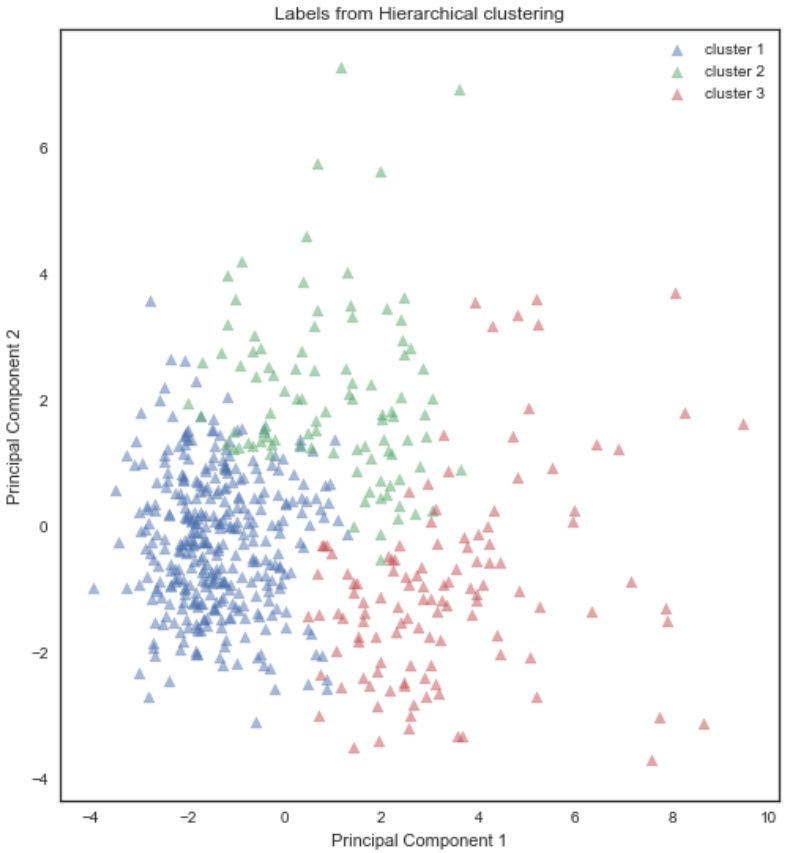




Hierarchical Clustering 실습

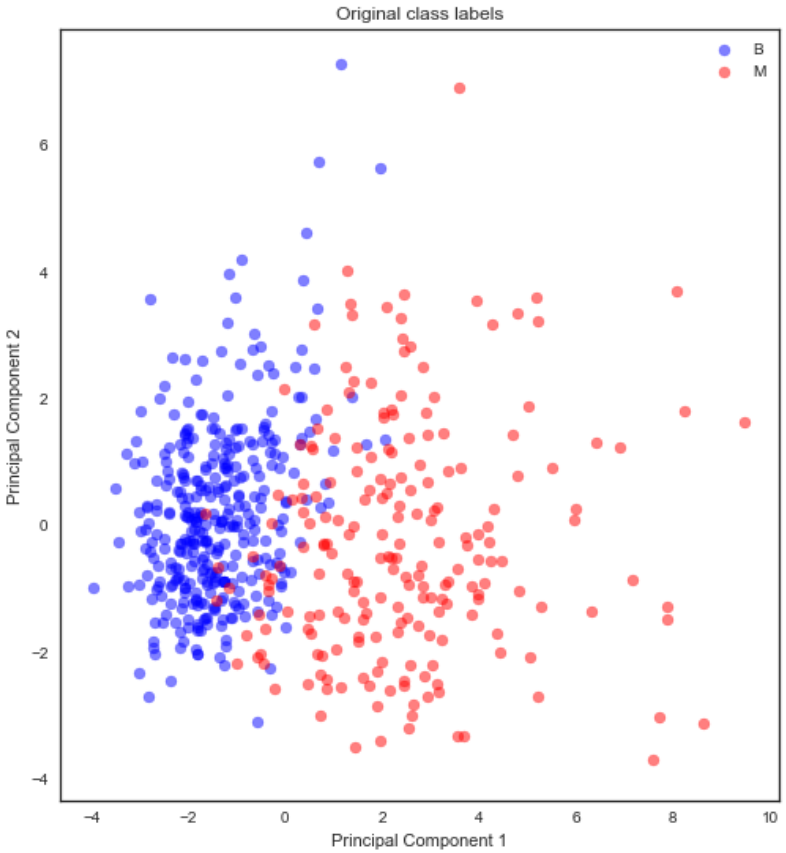
- Cluster 개수 = 3 (method='ward', metric='euclidean')

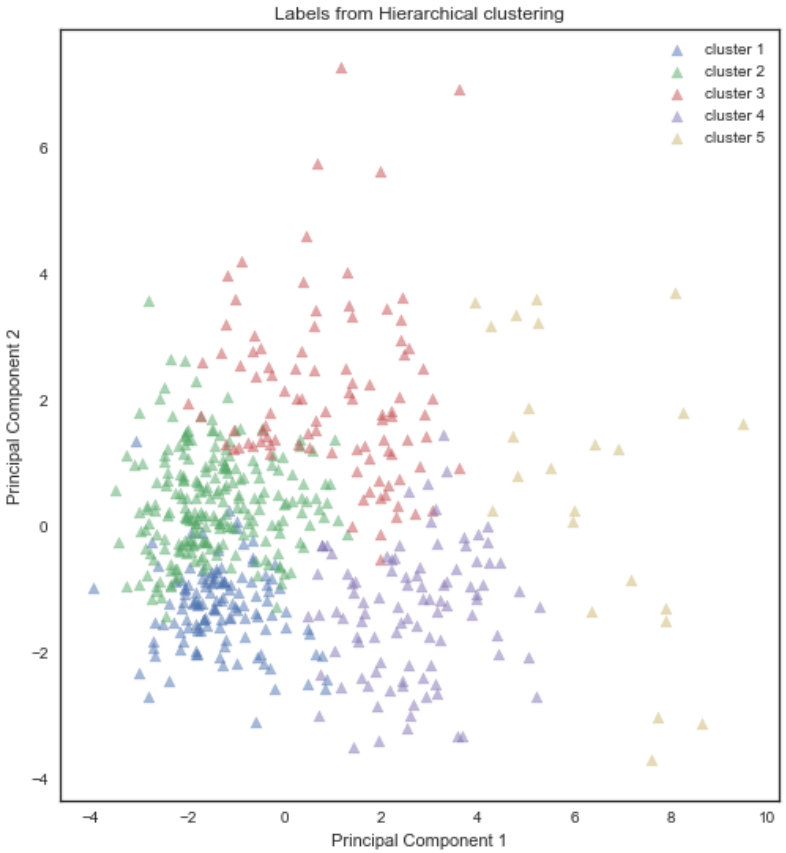




Hierarchical Clustering 실습

- Cluster 개수 = 5 (method='ward', metric='euclidean')





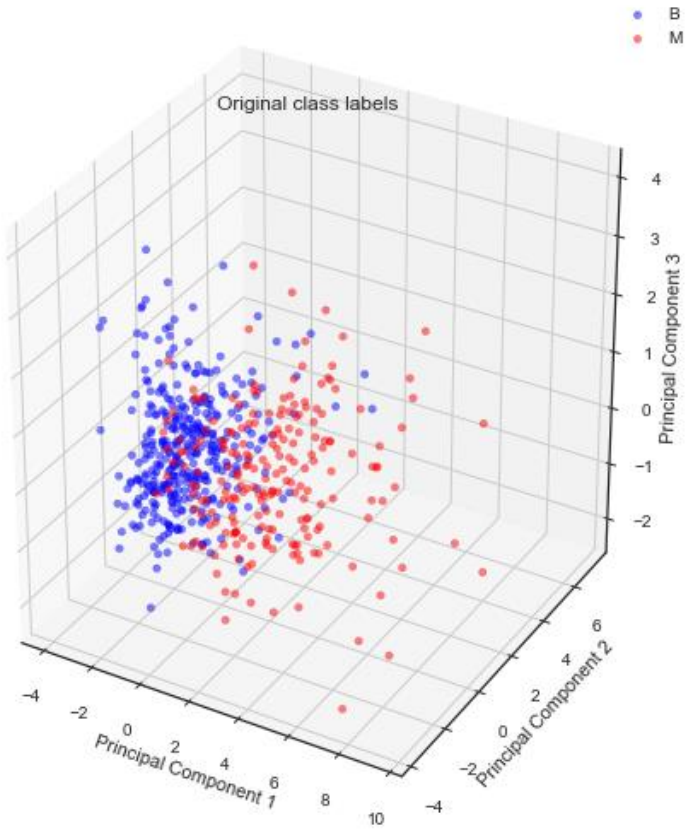
Hierarchical Clustering 실습

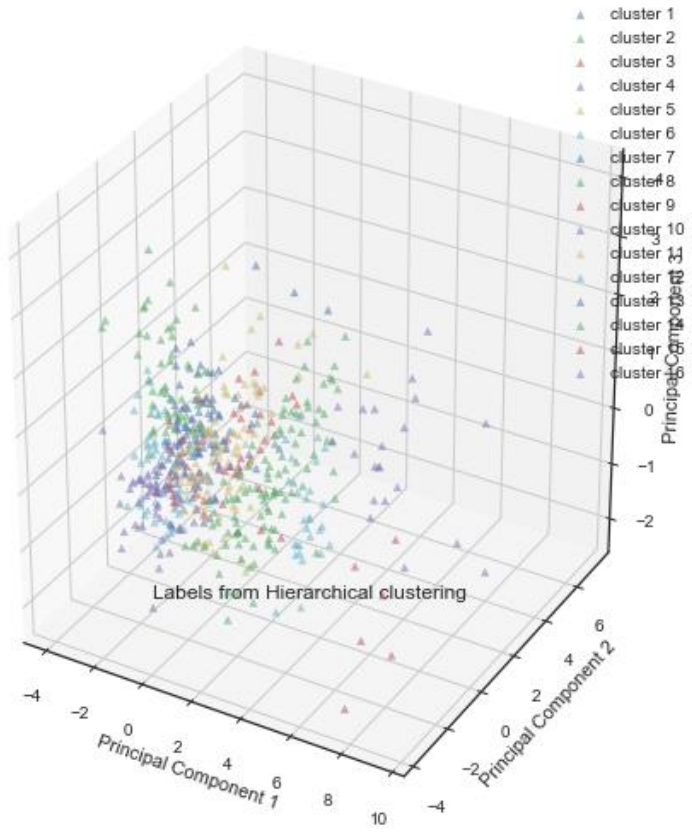
- Cluster 간 최소거리 지정 후, PCA로 학습한 PC1, PC2, PC3를 사용하여 결과 시각화

```
# Get cluster labels, by defining the upper threshold on distance
threshold = 20
label_hc = fcluster(D, t=threshold, criterion='distance')

# Plot on 3D space, using PCA components
# Perform PCA
pca = PCA(n_components=3)
Z = pca.fit_transform(X)

# Plot the transformed data (Z) with 3 PCs
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(18, 9), subplot_kw={'projection': '3d'})
axes = axes.ravel()
for label, color in zip(('B', 'M'), ('blue', 'red')):
    axes[0].scatter(Z[y == label, 0], Z[y == label, 1], Z[y == label, 2],
                    label=label, color=color, marker='o', alpha=0.5)
    axes[0].set_xlabel('Principal Component 1')
    axes[0].set_ylabel('Principal Component 2')
    axes[0].set_zlabel('Principal Component 3')
    axes[0].legend(loc='best')
    axes[0].set_title('Original class labels')
for i in range(max(label_hc) + 1):
    axes[1].scatter(Z[label_hc == i, 0], Z[label_hc == i, 1], Z[label_hc == i, 2],
                    label='cluster {}'.format(i + 1), marker='^', alpha=0.5)
    axes[1].set_xlabel('Principal Component 1')
    axes[1].set_ylabel('Principal Component 2')
    axes[1].set_zlabel('Principal Component 3')
    axes[1].legend(loc='best')
    axes[1].set_title('Labels from Hierarchical clustering')
plt.show(fig)
```

Hierarchical Clustering 실습

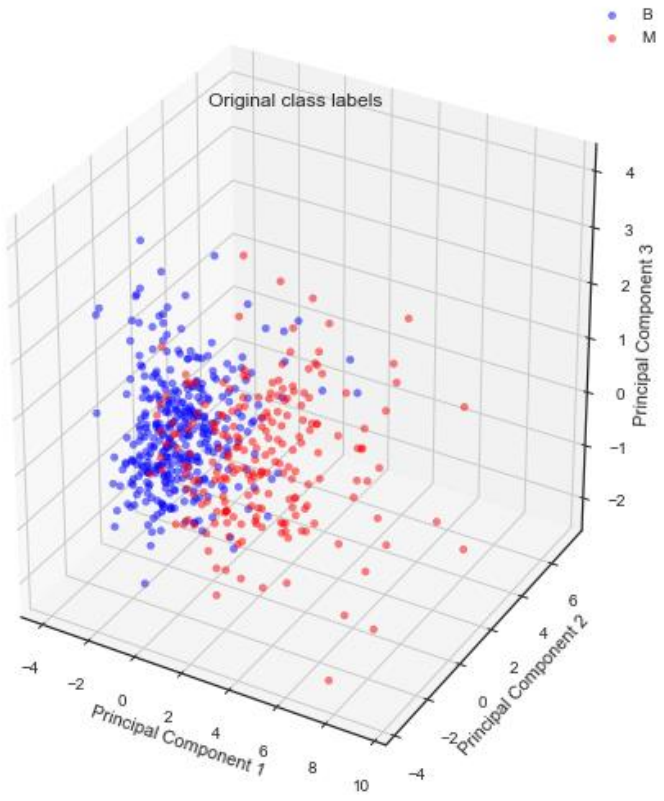
- Cluster 간 최소거리 = 10 (method='ward', metric='euclidean')

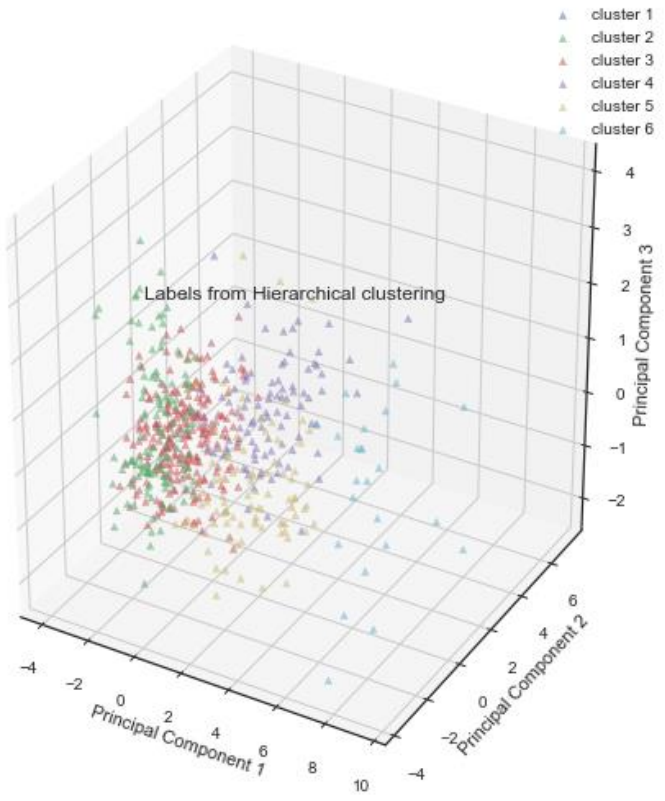




Hierarchical Clustering 실습

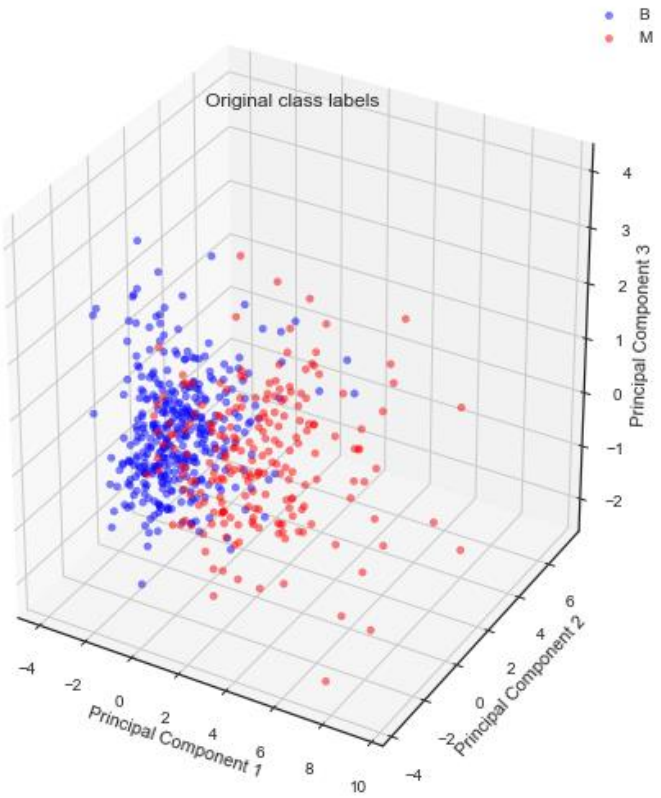
- Cluster 간 최소거리 = 20 (method='ward', metric='euclidean')

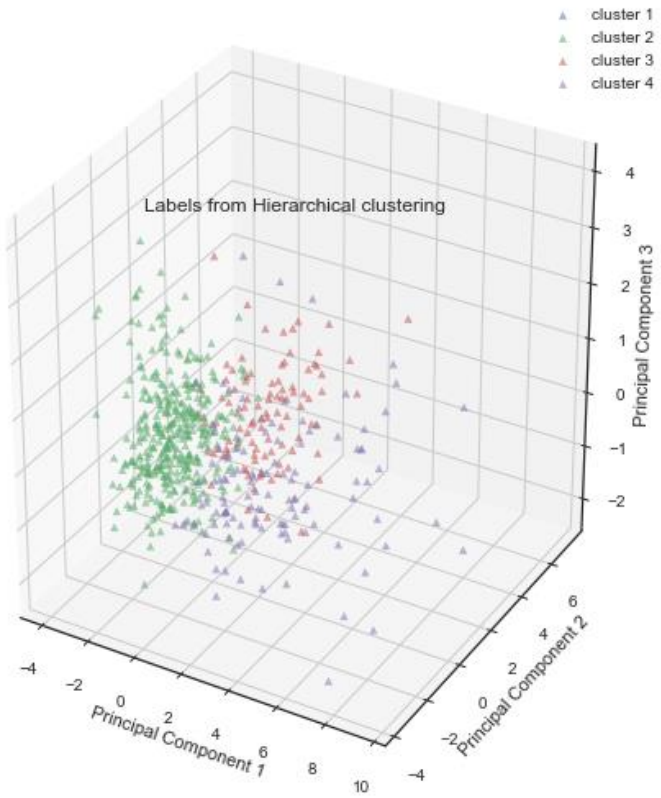




Hierarchical Clustering 실습

- Cluster 간 최소거리 = 30 (method='ward', metric='euclidean')





K-means Clustering 예제

K-means Clustering 실습

- 모듈 불러오기

```
# Import modules
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

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

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score, silhouette_samples
```

K-means Clustering 실습

- 데이터 불러오기

```
# Import data
data = pd.read_csv("./0_Data/BreastCancerWisconsin.csv")
print("- Data has {} rows and {} columns.".format(*data.shape))
print("- Column names: ", list(data.columns))
```

- X와 y로 나누기

```
# For clustering, we will only be using the X variables
X = data.drop(['diagnosis'], axis=1)
X = X.iloc[:, :10]
y = data['diagnosis']
```

- 데이터 정규화

```
# Standardize dataset columnwise, to have zero mean and unit variance
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

K-means Clustering 실습

- K-means Clustering 모델 학습 및 시각화 (cluster 개수 변경해보기)

```
# Perform K means clustering
n_clusters = 3
km = KMeans(n_clusters=n_clusters, random_state=2015010720)
km.fit(X)


# Get predicted labels from K means clustering
labels_km = km.predict(X)


# Visualize to see the result
pca = PCA(n_components=2)
Z = pca.fit_transform(X)
plt.figure(1)
for i in range(max(labels_km) + 1):
    plt.scatter(Z[labels_km == i, 0], Z[labels_km == i, 1],
                label='Cluster {}'.format(i + 1), alpha=.5)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```

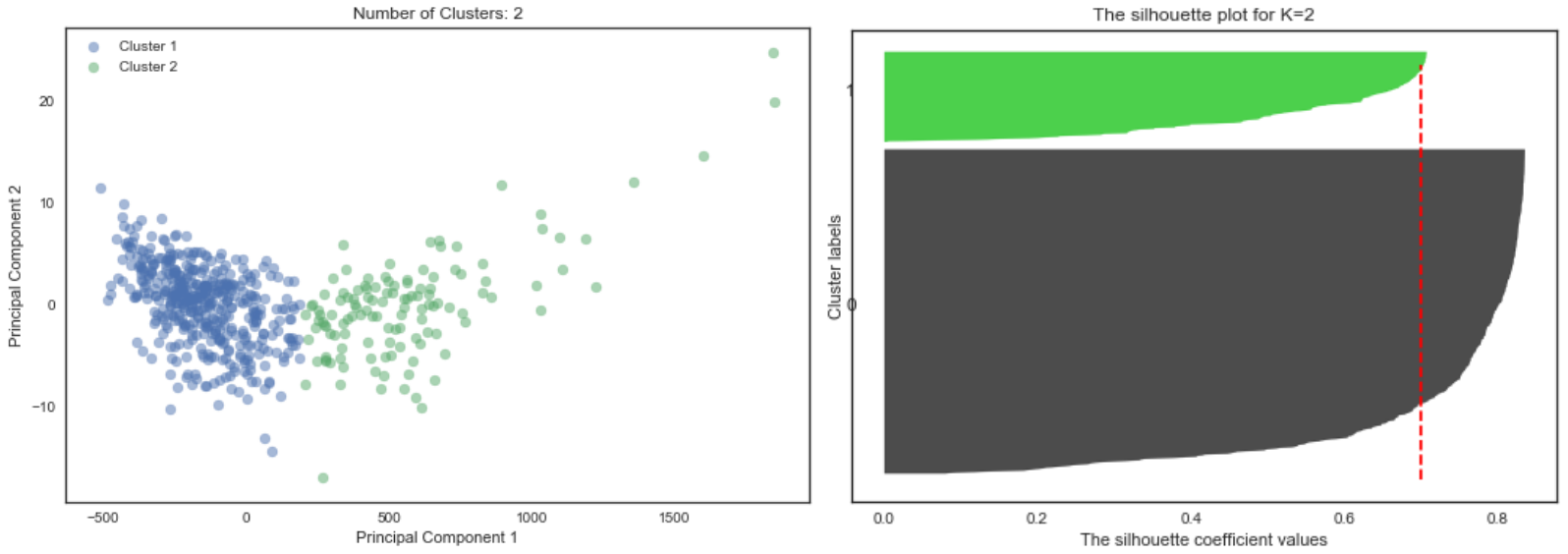
K-means Clustering 실습

- Silhouette score 계산 및 시각화 (cluster 개수 변경해보기)

```
# Silhouette scores
silhouette_avg = silhouette_score(X, labels_km)
print("For n_clusters =", n_clusters,
      "| The average silhouette_score is :", silhouette_avg)

# Compute the silhouette scores for each sample
sample_silhouette_values = silhouette_samples(X, labels_km)

# Visualize
plt.figure(2)
y_lower = 10
for i in range(n_clusters):
    ith_cluster_silhouette_values = sample_silhouette_values[labels_km == i]
    ith_cluster_silhouette_values.sort()
    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    color = cm.Spectral(float(i) / n_clusters)
    plt.fill_betweenx(np.arange(y_lower, y_upper),
                      0, ith_cluster_silhouette_values,
                      facecolor=color, edgecolor=color, alpha=0.7)
    plt.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
    y_lower = y_upper + 10
plt.vlines(x=silhouette_avg, ymin=0, ymax=X.shape[0], color="red", linestyle="--")
plt.title("The silhouette plot for K={}".format(n_clusters))
plt.xlabel("The silhouette coefficient values")
plt.ylabel("Cluster labels")
plt.yticks([])
plt.show()
```

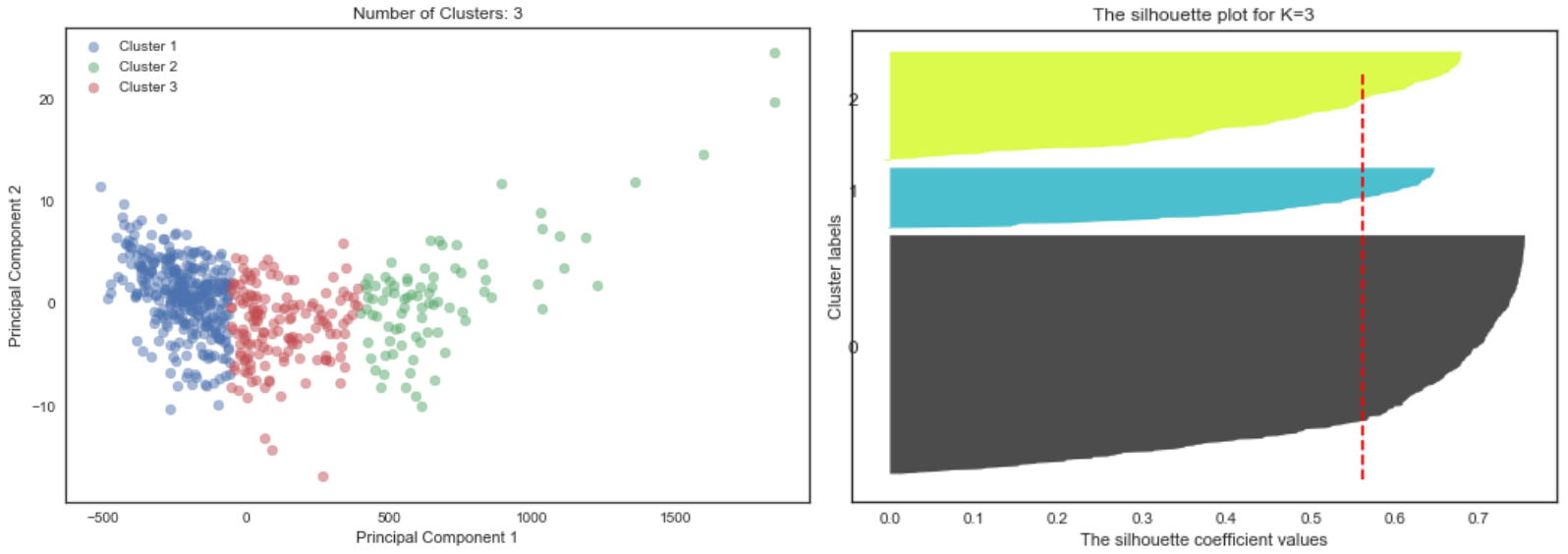
K-means Clustering 실습

- Cluster 개수 = 2



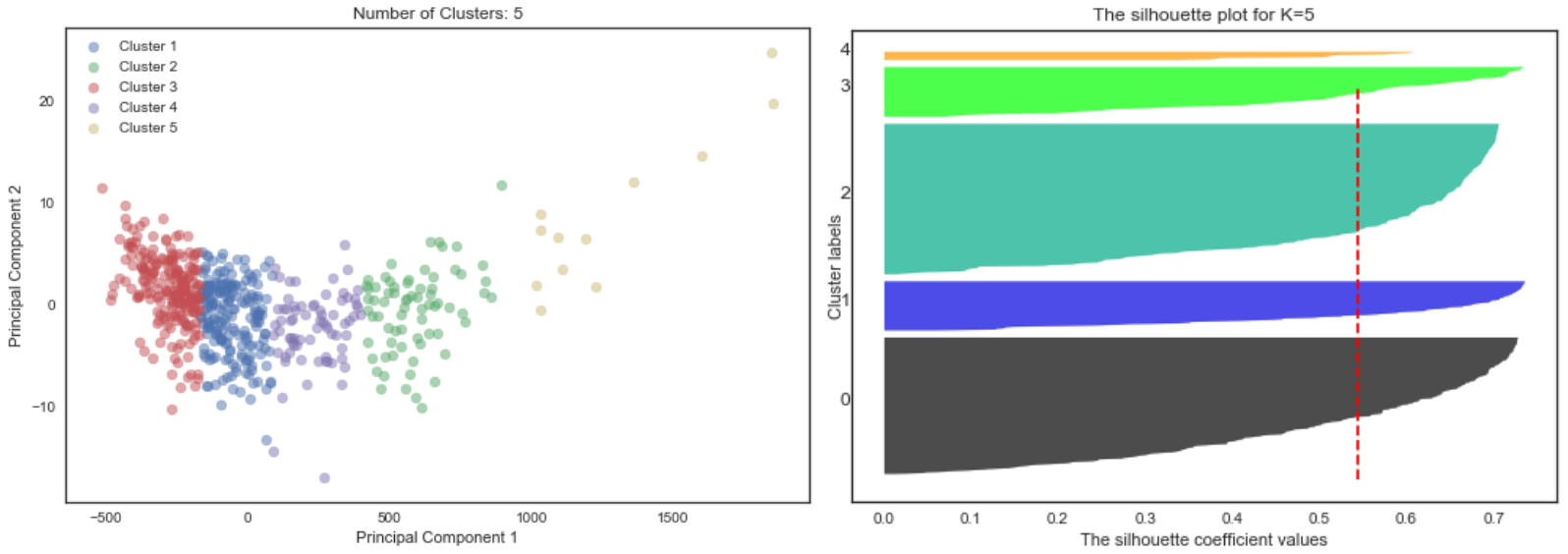
K-means Clustering 실습

- Cluster 개수 = 3



K-means Clustering 실습

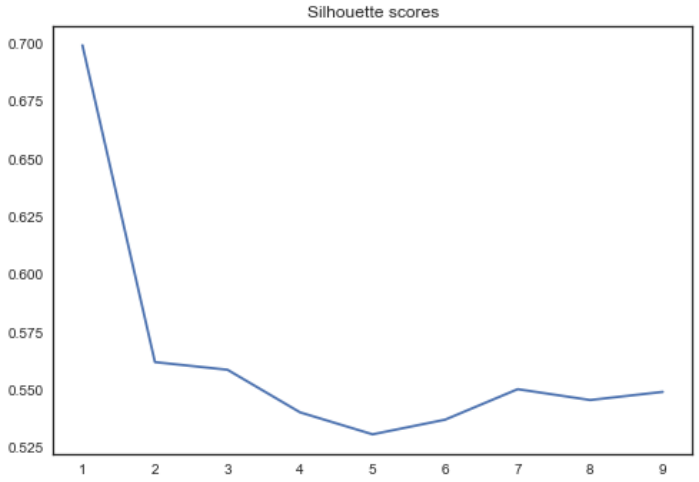
- Cluster 개수 = 5



K-means Clustering 실습

- Cluster 개수에 따른 silhouette score 계산 및 비교

```
# Find the best K
sil_scores = []
list_n_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10]
for n_clusters in list_n_clusters:
    km = KMeans(n_clusters=n_clusters)
    labels_km = km.fit_predict(X)
    sil_scores.append(silhouette_score(X, labels_km))

plt.figure(3)
plt.plot(range(1, len(sil_scores) + 1), sil_scores)
plt.title('Silhouette scores')
plt.show()
```



Clustering 실습

데이터 설명

- **PCA 실습과 동일한 데이터를 활용**
- Clustering 방법론 적용 / PCA를 통한 시각화 / 최적의 군집 추출 및 특징 분석

- **실습 데이터 1: P2P 대출 상환 데이터**
  - P2P 대출 관련 데이터
  - 22개의 변수 / 28,784개 관측치로 구성

- **실습 데이터 2: 대출 연체 데이터**
  - 대출 연체 예측을 위한 데이터 (성별, 대출 금액 등)
  - 27개의 변수 / 43,386개의 관측치로 구성

- **각 폴더내의 Description.txt 파일 참조**