PCA 실습

PCA 실습 #1

- 데이터셋: 와인데이터
- 학습/시험 데이터: X, 학습/시험 데이터 라벨: Y

① 데이터 로드

```
[1]  1 # 데이터 로드
     2 from sklearn.datasets import load_wine
     3 data = load_wine()
     4 X = data.data
     5 Y = data.target
```

② 데이터 분할 (학습/테스트)

```
[ ]  1 # 학습/테스트 데이터 나누기
     2 from sklearn.model_selection import train_test_split
     3 X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3,random_state=1,stratify=Y)
     4
```

③ 데이터 정규화

```
[3]  1 # 데이터 정규화
     2 from sklearn.preprocessing import StandardScaler
     3 std = StandardScaler()
     4 X_train_std = std.fit_transform(X_train)
     5 X_test_std = std.transform(X_test)
```

PCA 실습 #1

- 데이터셋: 와인데이터
- 학습/시험 데이터: X, 학습/시험 데이터 라벨: Y

④ 데이터 차원 축소

```
[4]  1 # PCA를 통해 데이터 차원 축소
     2 from sklearn.decomposition import PCA
     3
     4 lpca = PCA(n_components=4)
     5 X_train_pca = lpca.fit_transform(X_train_std)
     6 X_test_pca = lpca.transform(X_test_std)
```

← PCA의 n_components 결정은 어떻게?

⑤ 모델 학습

```
[5]  1 # 분류기 학습 및 테스트
     2 from sklearn.linear_model import LogisticRegression
     3
     4 lr = LogisticRegression()
     5 lr.fit(X_train_pca, Y_train)
     6 Y_train_pred = lr.predict(X_train_pca)
     7 Y_test_pred = lr.predict(X_test_pca)
     8
     9 from sklearn import metrics
    10 print(metrics.accuracy_score(Y_train, Y_train_pred))
    11 print(metrics.accuracy_score(Y_test, Y_test_pred))
    12 print(metrics.confusion_matrix(Y_test, Y_test_pred))
```

```
0.9758064516129032
0.9629629629629629
[[18  0  0]
 [ 2 19  0]
 [ 0  0 15]]
```

PCA 실습 #1

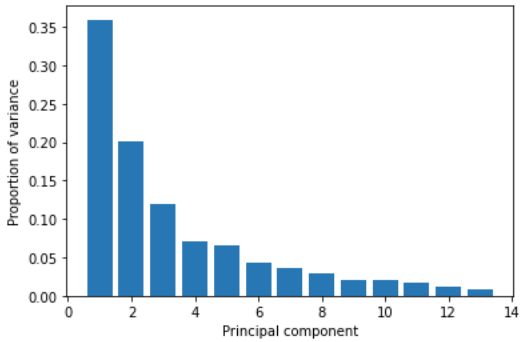
- 데이터셋: 와인데이터
- 학습/시험 데이터: X, 학습/시험 데이터 라벨: Y

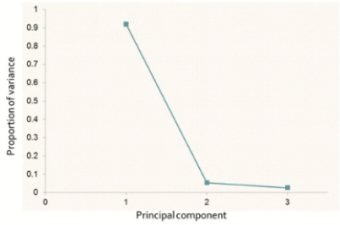
⑥ PCA 주축의 수 결정법

```
[5]  1 # PCA의 주축 개수 구하기 위한 분석
     2 # covariance
     3 #
     4 import numpy as np
     5 scov=np.cov(X_train_std.T)
     6 eigen_vals, eigen_vecs=np.linalg.eig(scov)
     7 print('Eigenvalues \n%s' %eigen_vals)
```

```
Eigenvalues
[4.7095539  2.63606471 1.55728758 0.93422662 0.85129454 0.5709688
 0.46462025 0.37764772 0.10409155 0.14951983 0.21165109 0.2630501
 0.27571434]
```

```
1 # Explained variance ratio
2 total = sum(eigen_vals)
3 var_exp = [(i / total) for i in sorted(eigen_vals, reverse=True)]
4
5 import matplotlib.pyplot as plt
6 plt.bar(range(1,14), var_exp)
7 plt.ylabel('Proportion of variance')
8 plt.xlabel('Principal component')
9 plt.show()
```





첫번째 주성분으로 설명가능한 데이터 비율

$$= \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}$$

$$= \frac{2.7596}{0.0786 + 0.1618 + 2.7596}$$

$$= 0.920$$