판별 분석 실습

Discriminant analysis

선형판별분석(LDA) 실습

- 실습 코드: 링크 (블랙보드, 깃허브 참고)
- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔ (0,1,2)

- 데이터 로더, 범주형 데이터 변환, 데이터 분할
  - 3, 4장 실습과 동일

```
# Iris data 불러오기
import seaborn as sns # seaborn을 불러옴.
iris=sns.load_dataset('iris') # iris라는 변수명으로 Iris data를 download함.
X=iris.drop('species',axis=1) # 'species'열을 drop하고 특성변수 X를 정의함.
y_=iris['species'] # 'species'열을 label y를 정의함.

from sklearn.preprocessing import LabelEncoder # LabelEncoder() method를 불러옴
classle=LabelEncoder()
y=classle.fit_transform(iris['species'].values) # species 열의 문자형을 범주형 값으로 전환

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y, test_size=0.3, random_state=123, stratify=y)

```

선형판별분석(LDA) 실습

- 실습 코드: 링크
- 데이터셋: Iris 데이터 ➔ (150 x 5) 즉, n_samples = 150, n_features = 5
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2) 즉, n_classes = 3

- 선형판별분석 API: Manual
  - LinearDiscriminantAnalysis 클래스 호출
    - n_components : 사영할 축의 차원

```
# Iris data에 대한 LDA 적합
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
cld=LinearDiscriminantAnalysis(store_covariance=True)

cld.fit(X_train, y_train) # LDA 적합
y_train_pred=cld.predict(X_train)
y_test_pred=cld.predict(X_test)
```

선형판별분석(LDA) 실습 #1

- 실습 코드: 링크
- 데이터셋: Iris 데이터 ➔ (150 x 5) 즉, n_samples = 150, n_features = 5
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2) 즉, n_classes = 3

- 선형판별평가 API: Accuracy & Confusion Matrix
  - 3, 4장 실습과 동일

```
10 from sklearn.metrics import accuracy_score
11 print(accuracy_score(y_train, y_train_pred)) # train data에 대한 accuracy
12 print(accuracy_score(y_test, y_test_pred)) # test data에 대한 accuracy
```

```
0.9714285714285714
0.9777777777777777
```

```
[24] 1 # 분류 결과
     2 from sklearn.metrics import confusion_matrix
     3 print(confusion_matrix(y_test, y_test_pred))  # 각 행은 setosa, versicolor, virginica
```

```
[[15  0  0]
 [ 0 14  1]
 [ 0  0 15]]
```

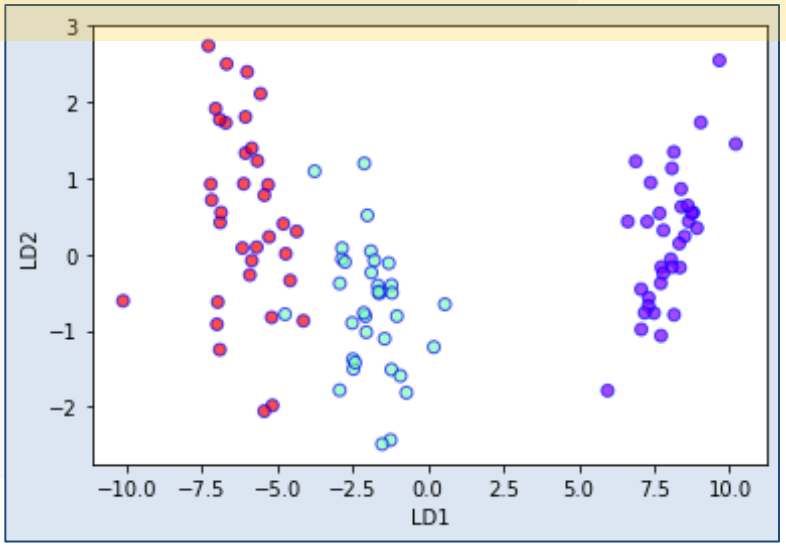
선형판별분석(LDA) 실습 #1

- 실습 코드: 링크
- 데이터셋: Iris 데이터 ➔ (150 x 5) 즉, n_features = 5
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2) 즉, n_classes = 3

- Plot: transformed data n_components: min(n_classes − 1, n_features) ➔ 2

```python
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
cld=LinearDiscriminantAnalysis()
X_lda = cld.fit_transform(X_train, y_train)

from matplotlib import pyplot as plt
plt.xlabel('LD1')
plt.ylabel('LD2')
plt.scatter(
    X_lda[:,0],
    X_lda[:,1],
    c=y_train,
    cmap='rainbow',
    alpha=0.7,
    edgecolors='b'
)
```



이차판별분석(QDA) 실습 #1

- 실습 코드: 링크
- 데이터셋: Iris 데이터 ➔ (150 x 5) 즉, n_features = 5
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2) 즉, n_classes = 3

- 데이터 로더, 범주형 데이터 변환, 데이터 분할
  - 3, 4장 실습과 동일

- 이차판별분석 API: Manual
  - QuadraticDiscriminantAnalysis 클래스 호출

```
1 from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
2
3 cqd=QuadraticDiscriminantAnalysis(store_covariance=True)
4
5 cqd.fit(X_train, y_train) # QDA 적합
6 y_train_pred=cqd.predict(X_train)
7 y_test_pred=cqd.predict(X_test)
8
```

이차판별분석(QDA) 실습 #1

- 실습 코드: 링크
- 데이터셋: Iris 데이터 ➔ (150 x 5) 즉, n_features = 5
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2) 즉, n_classes = 3

- Accuracy & Confusion Matrix

```
 9 from sklearn.metrics import accuracy_score
10 print(accuracy_score(y_train, y_train_pred)) # train data에 대한 accuracy
11 print(accuracy_score(y_test, y_test_pred)) # test data에 대한 accuracy
```

```
0.9809523809523809
0.9777777777777777
```

```
[28]  1 # 분류 결과
      2 from sklearn.metrics import confusion_matrix
      3 print(confusion_matrix(y_test, y_test_pred))
```

```
[[15  0  0]
 [ 0 14  1]
 [ 0  0 15]]
```

이차판별분석(QDA) 실습 #1

- 실습 코드: 링크
- 데이터셋: Iris 데이터 ➔ (150 x 5) 즉, n_features = 5
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2) 즉, n_classes = 3

- Accuracy & Confusion Matrix

```
 9 from sklearn.metrics import accuracy_score
10 print(accuracy_score(y_train, y_train_pred)) # train data에 대한 accuracy
11 print(accuracy_score(y_test, y_test_pred)) # test data에 대한 accuracy
```

```
0.9809523809523809
0.9777777777777777
```

```
[28]  1 # 분류 결과
      2 from sklearn.metrics import confusion_matrix
      3 print(confusion_matrix(y_test, y_test_pred))
```

```
[[15  0  0]
 [ 0 14  1]
 [ 0  0 15]]
```