의사결정나무 실습

의사결정나무(DT) 실습 #1 (분류)

- 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, 4,5장 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)

```

의사결정나무(DT) 실습 #1 (분류)

- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2)

- 의사결정나무 API: Manual
  - DecisionTreeClassifier 클래스 호출
    - criterion : 불순도 알고리즘 옵션, default='gini' 가능 파라미터={entropy,gini}

```
# Classification Tree# 또는 from sklearn import DecisionTreeClassifier

# 과적합 사례 확인 법: max_depth 3=> 5=> 7 높일 수록 학습 데이터 정확도 올라라고 테스트 정확도 고정
#dtc = tree.DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=1)
dtc = tree.DecisionTreeClassifier(criterion='entropy', max_depth=7, random_state=1)
dtc.fit(X_train, y_train)
y_train_pred = dtc.predict(X_train)  # Training accuracy
y_test_pred = dtc.predict(X_test)    # Test accuracy

```

의사결정나무(DT) 실습 #1 (분류)

- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2)

- 의사결정나무 API: Accuracy & Confusion Matrix
  - 3, 4, 5 장 실습과 동일

```
[20]  1 from sklearn.metrics import accuracy_score
      2 print(accuracy_score(y_train, y_train_pred))
      3 print(accuracy_score(y_test, y_test_pred))
```

```
0.9904761904761905
0.9777777777777777
```

```
[21]  1 from sklearn.metrics import confusion_matrix
      2 print(confusion_matrix(y_test,y_test_pred))
```

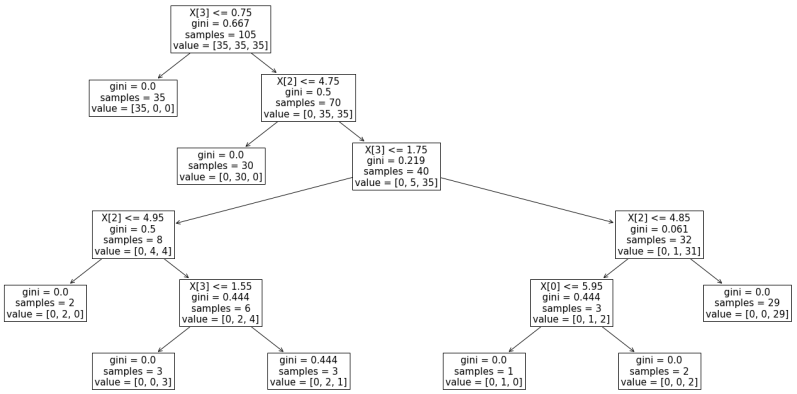
```
[[15  0  0]
 [ 0 15  0]
 [ 0  1 14]]
```

의사결정나무(DT) 실습 #1 (분류)

- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2)

- Plot: tree.plot_tree [API]

```
1 import matplotlib.pyplot as plt
2 #tree.plot_tree(dtc.fit(X_train,y_train))
3 fig, ax = plt.subplots(figsize=(25, 12))
4 tree.plot_tree(dtc.fit(X_train, y_train), fontsize=15)
5 plt.savefig('tree_high_dpi', dpi=100)
```

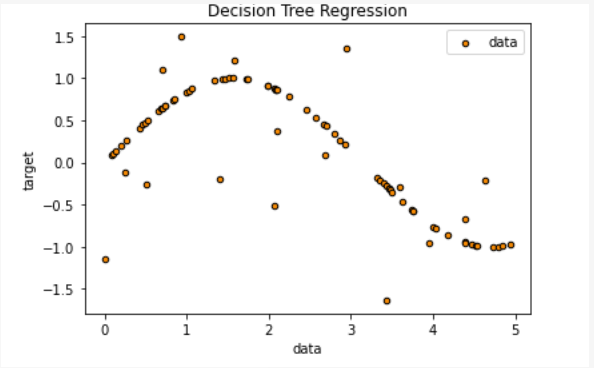


Values =: 각 클래스별 데이터 수

의사결정나무(DT) 실습 #2 (회귀)

- 데이터셋: 합성데이터

```
# Create a random dataset
rng = np.random.RandomState(1)
X = np.sort(5 * rng.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - rng.rand(16))

```



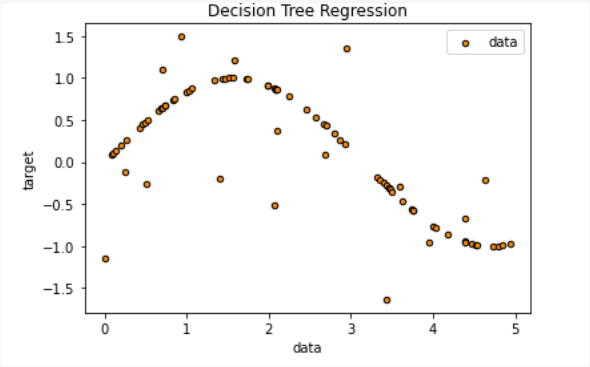
- 의사결정나무 API: [Manual]
  - DecisionTreeRegressor 클래스 호출
    - criterion : 불순도 알고리즘, {“mse”, “friedman_mse”, “mae”}, default=”mse”

```
[22]  # Fit regression model
      regr_1 = DecisionTreeRegressor(max_depth=2)
      regr_2 = DecisionTreeRegressor(max_depth=5)
      regr_1.fit(X, y)
      regr_2.fit(X, y)

      y_pred_1 = regr_1.predict(X)
      y_pred_2 = regr_2.predict(X)

```

의사결정나무(DT) 실습 #2 (회귀)

- 데이터셋: 합성데이터

```
# Create a random dataset
rng = np.random.RandomState(1)
X = np.sort(5 * rng.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - rng.rand(16))

```



- 의사결정나무 API: MSE/RMSE [API]
  - Mean_squared_error: squared=True ➔ MSE, False➔RMSE

```
from sklearn.metrics import mean_squared_error

print(mean_squared_error(y, y_pred_1))
print(mean_squared_error(y, y_pred_2))
```

```
0.12967126328231798
0.025236948989861896
```