Machine learning

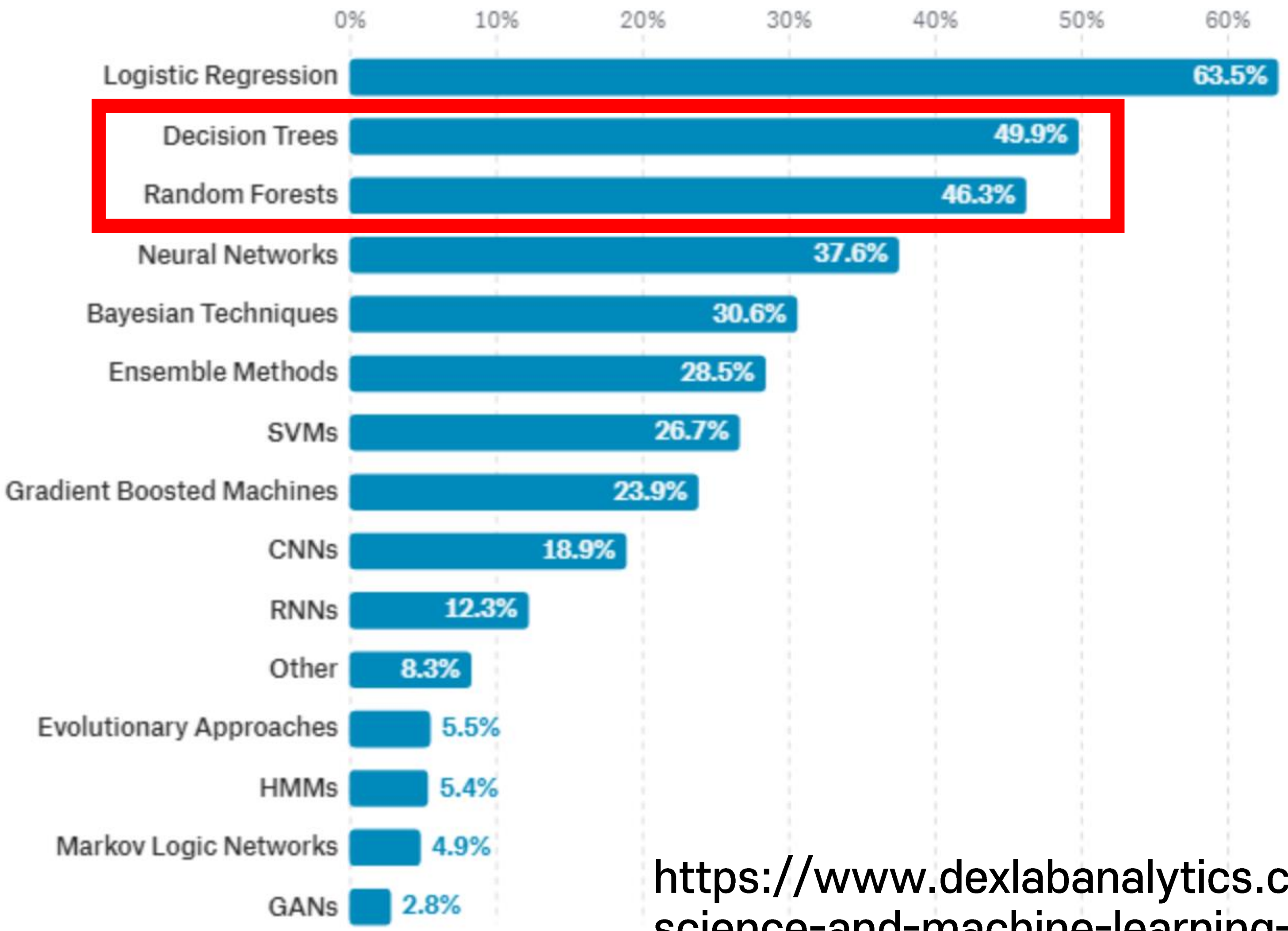
Python code

#LifeCode

Decision Tree

Random Forest

데이터과학자들이 많이 사용하는 머신러닝 기법



https://www.dexlabanalytics.com/blog/data-science-and-machine-learning-in-what-state-they-are-to-be-found

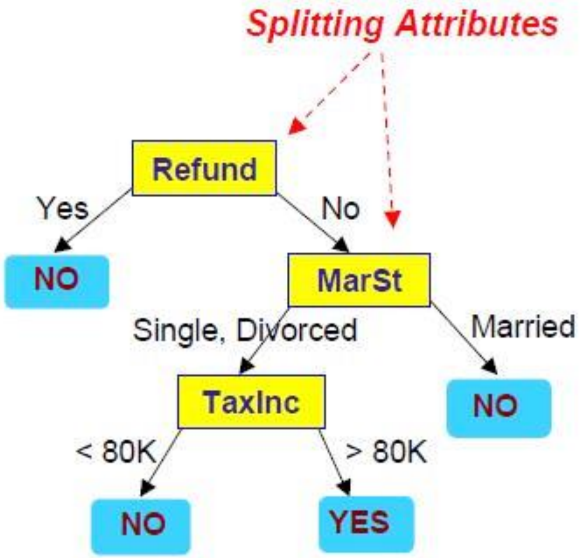
Decision Tree Classifier

- Feature에서부터 Label을 가장 잘 구분하는 선택지 힌트 구성

= Feature라는 뿌리에서 Label 나뭇잎까지 Tree 구성.

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

Entropy

= 엉망

Entropy

- Entropy
  = 엉망 (무질서, 어원: 안쪽 변화) 정도 표현

- 'Entropy가 커진다'는 의미는
  = 더 불확실 해진다.
  = 더 무질서 정보의 양 → '정보의 양?'
  (경우의 수) 많아진다.

Information Entropy, H(x)

$$H(X) = -\sum P(x_i)\log(P(x_i))$$

브라질 vs *아르헨티나* *축구*, *승리 확률 (0.5, 0.5)라면,*

$H1 =$ 0.5 $*$ -np.log(0.5) + 0.5 $*$ -np.log(0.5)

≒ 0.69

국가대표 vs *조기축구팀*, *승리 확률 (0.99, 0.01),*

$H2 =$ 0.99 $*$ -np.log(0.99) + 0.01 $*$ -np.log(0.01)

≒ 0.06

목표! : 엉망(엔트로피) 감소하는 것

$$Ent(D) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

전체 데이터 D의 엔트로피

$$Ent_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{D} * Ent(D_j)$$

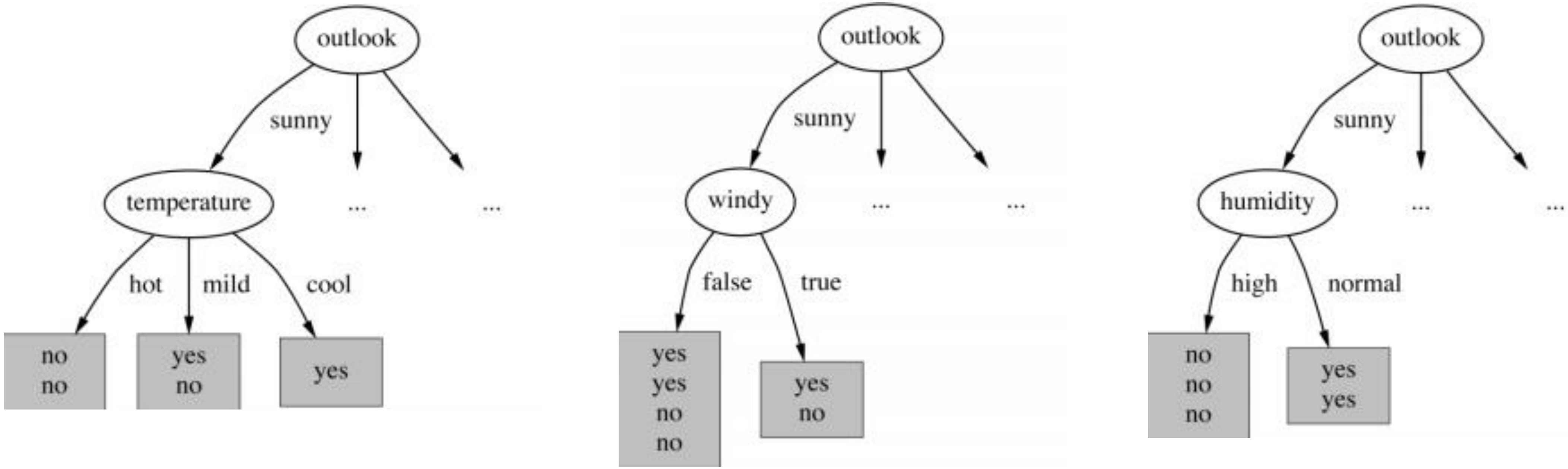
속성 A로 분류시 엔트로피

$$Gain(A) = Ent(D) - Ent_A(D)$$

A 속성의 정보 소득

Information Gain: 축구 사례

- Now we search for the best split at the next level:



Temperature = 0.571 Windy = 0.020 Humidity = 0.971

https://www3.nd.edu/~rjohns15/cse40647.sp14/www/home.php

sklearn.tree.DecisionTreeClassifier

class `sklearn.tree.DecisionTreeClassifier`(*, *criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort='deprecated', ccp_alpha=0.0*) ¶

Parameters:	**criterion : *{"gini", "entropy"}, default="gini"*** The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

엉망 = 불순도

- Impure (Not pure)   vs   순수 pure

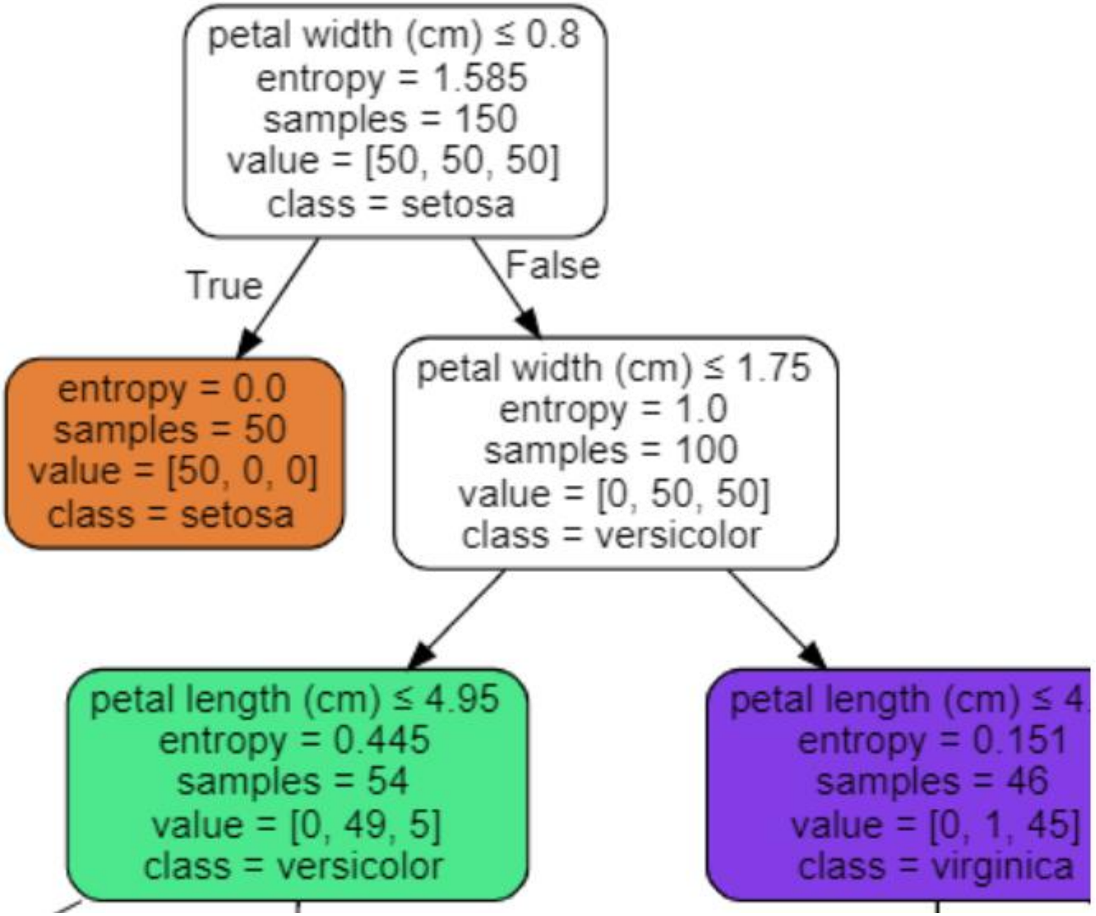
= Label   섞임   vs   모두 같음

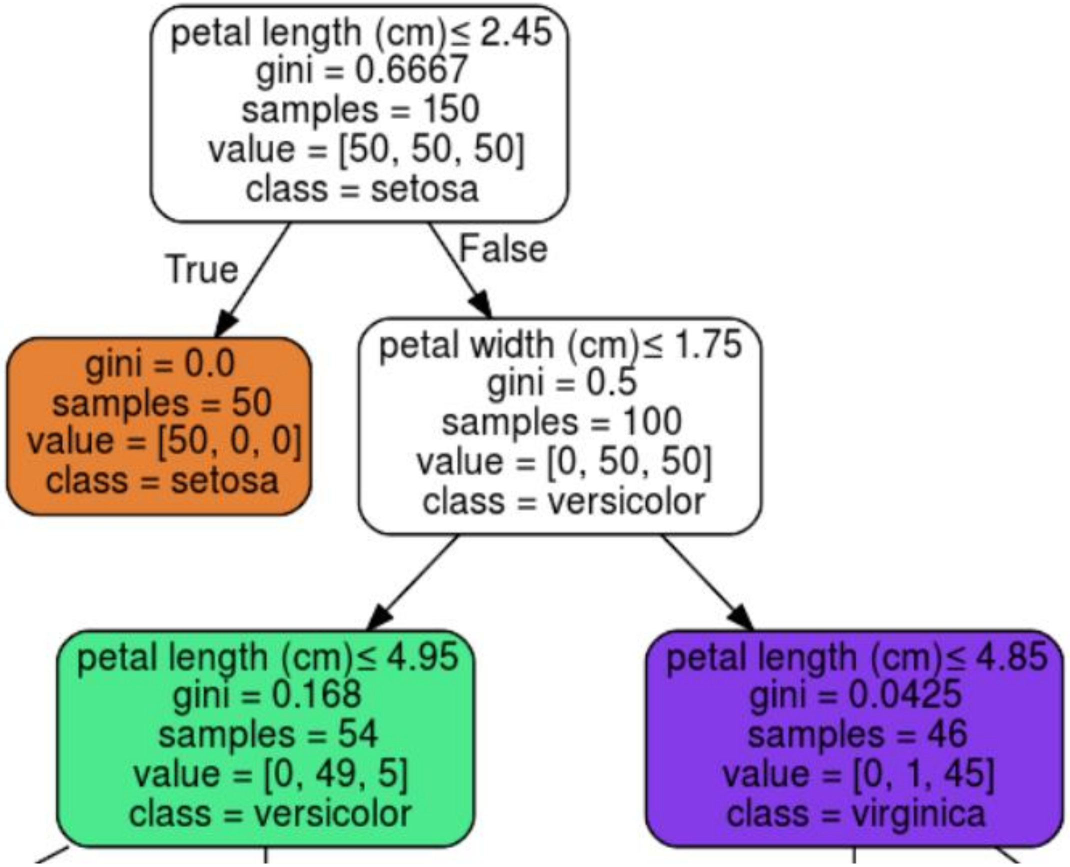
= Impurity 지표로 판단

‘entropy’ or ‘gini’

Decision Tree, 꽃잎 examples

속성: 너비, 불순도 기준 : entropy(=IG) vs 속성: 길이, 불순도 기준 : gini

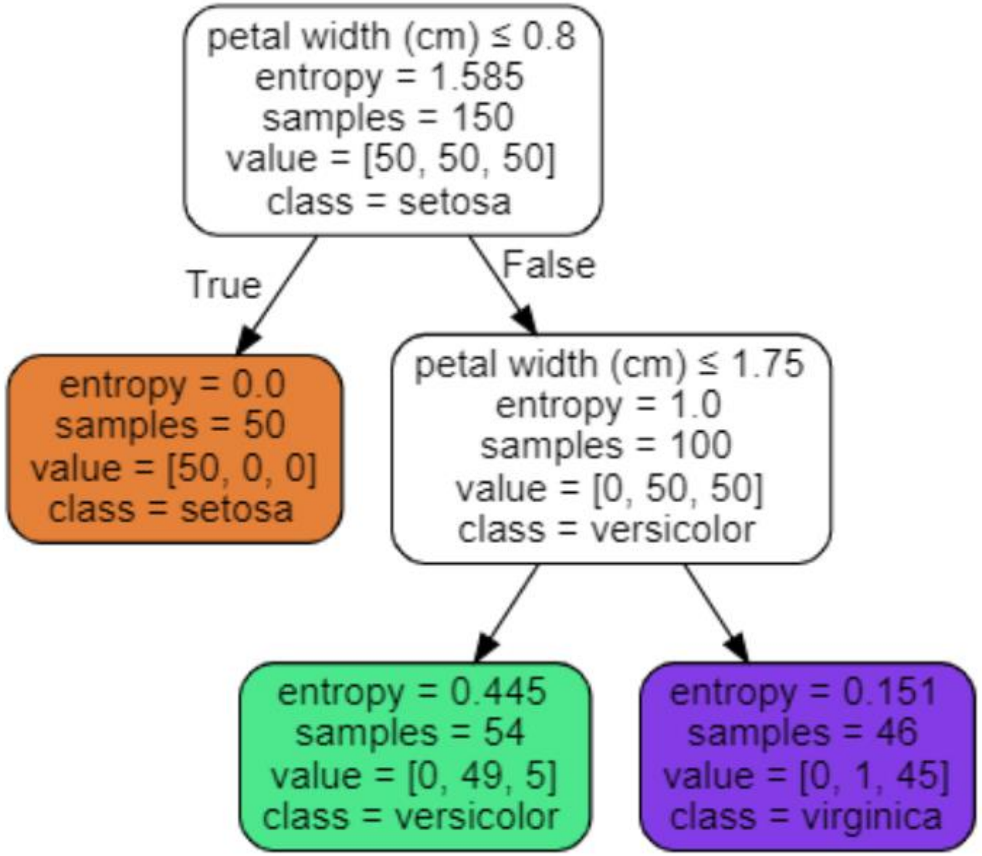




가지치기(프루닝 pruning)

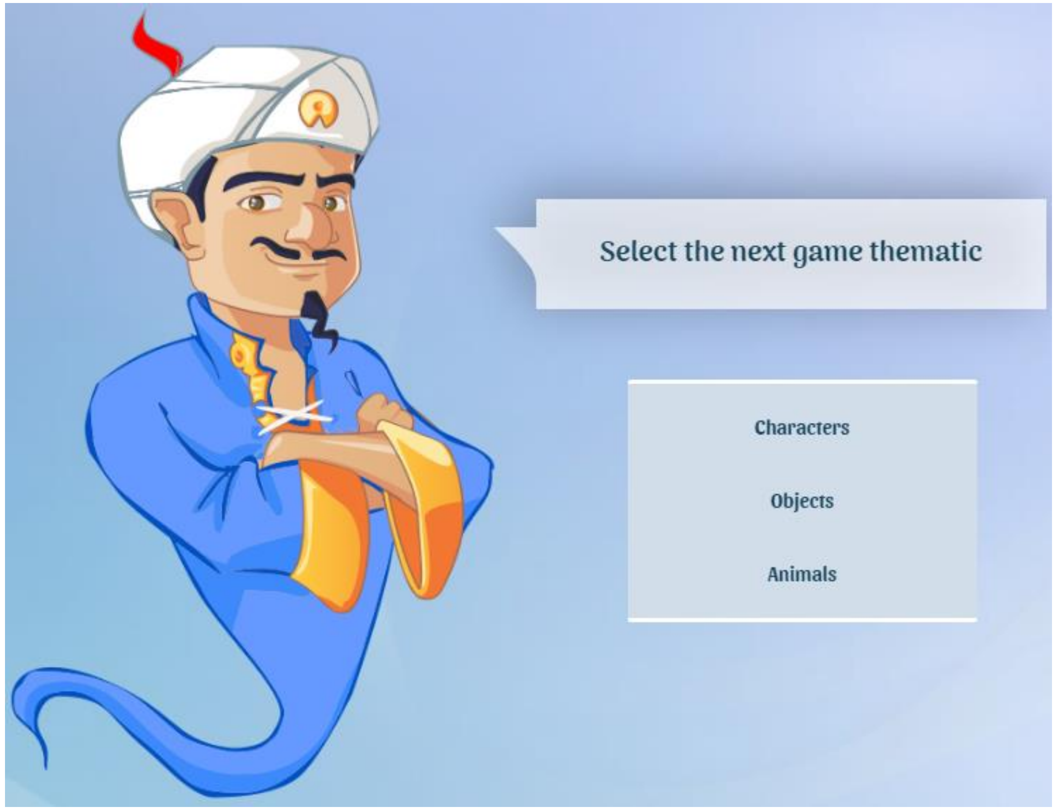
```
clf3 = tree.DecisionTreeClassifier(criterion='entropy', max_depth=2)
clf3.fit(iris.data, iris.target)
```

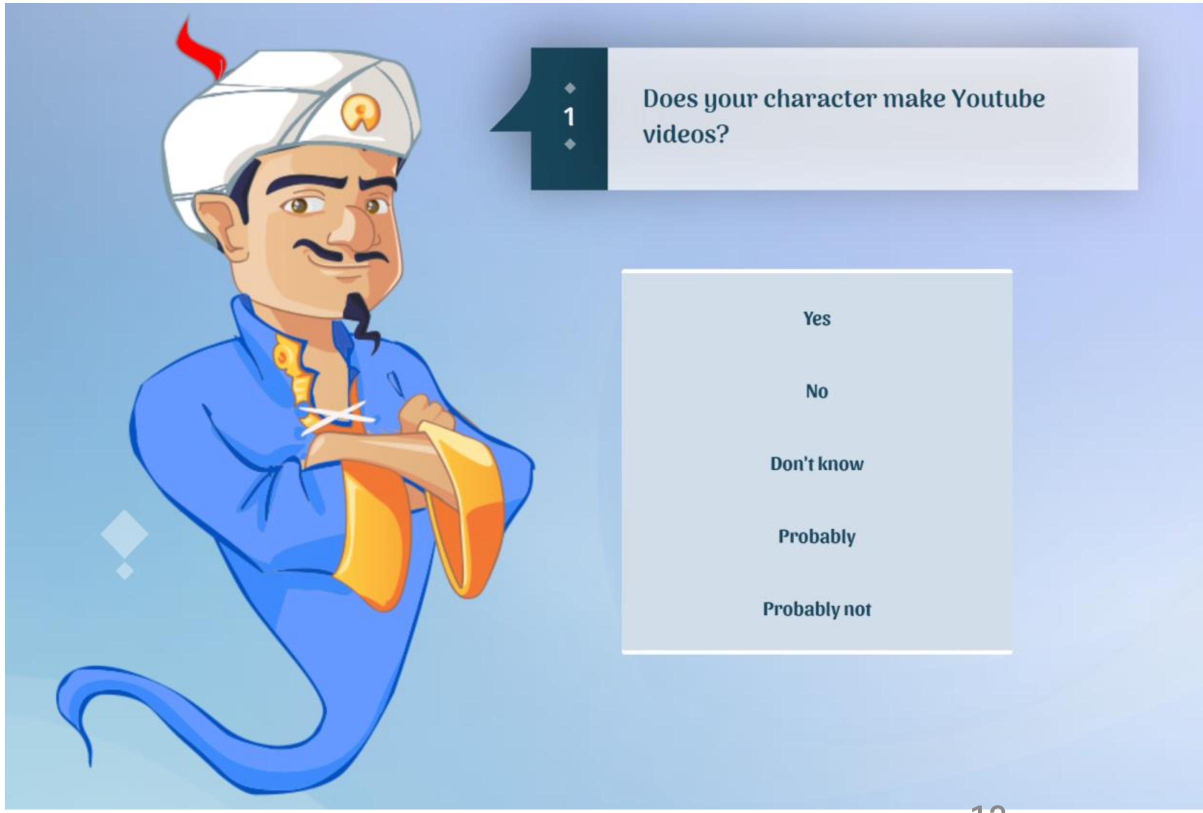
속성: 너비, 불순도 기준: **entropy(=IG)**



gini

- 선택을 모아서 지니가 대상을 추측하는 게임. a.k.a. 스무고개





https://en.akinator.com/game

gini

- **Measurement of inequality 같지않음 지표**

- **by Corrado Gini (Italian statistician)**

Parameters:	***criterion : {"gini", "entropy"}, default="gini"*** The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

gini

- x_1 속성 → y 두 label로 나누고 싶을 때

x_1	1	2	3	4	5	6	7	8
y	0	0	0	1	1	1	1	1

모인 샘플 들끼리 비슷함 = 순수함

If we split at $x_1 < 3.5$, we get an optimal split.
If we split at $x_1 < 4.5$, we make a mistake (misclassification).

Idea: *A better split should make the samples "pure" (homogeneous).*

Gini Index

The Gini index is defined as:

$$\text{Gini} = 1 - \sum_{i=1}^{K} {p_k}^2$$

where p_k denotes the proportion of instances belonging to class k $(K = 1, \ldots, k)$.

마케팅 미션: 어떤 사람이 컴퓨터를 살까?

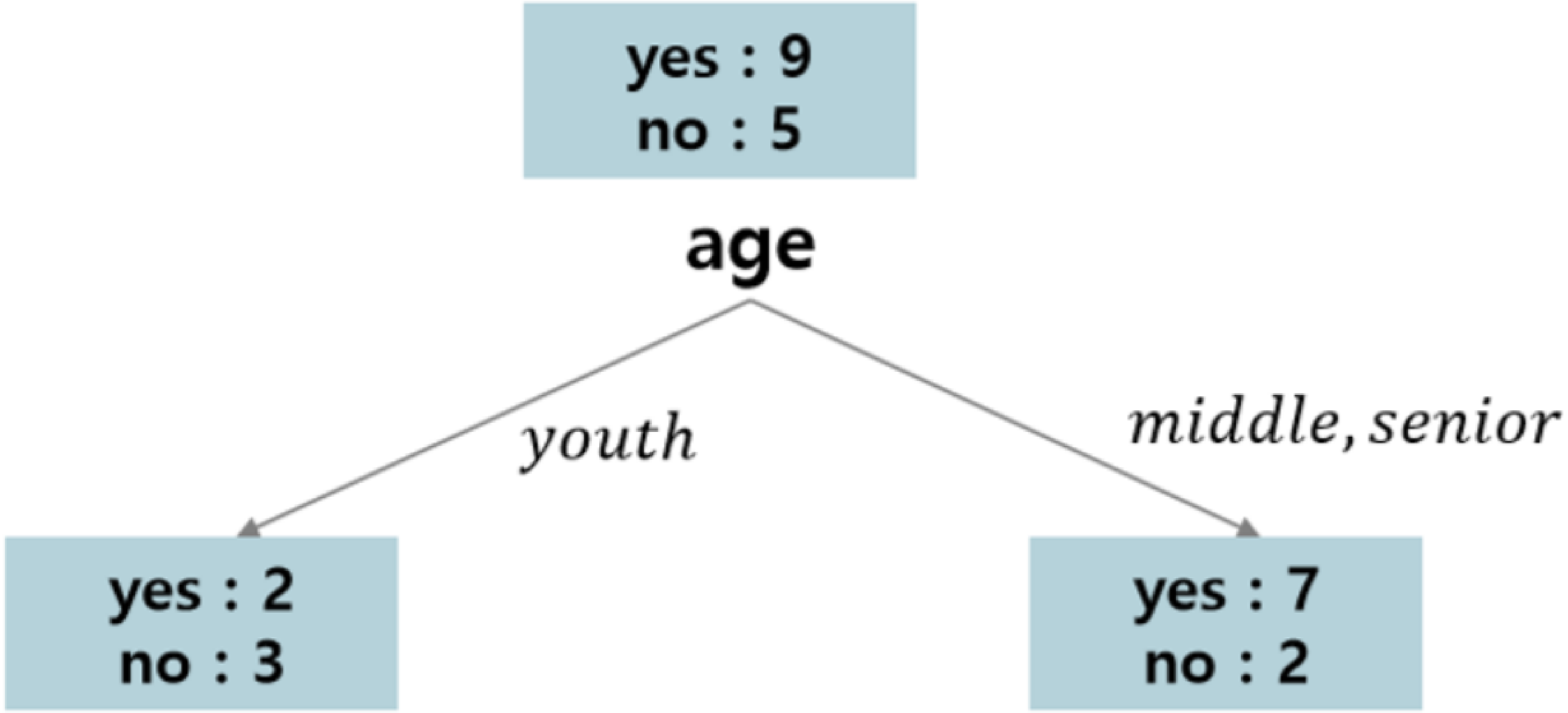
Case, 개인

	age	income	student	credit_rating	class_buys_computer
0	youth	high	no	fair	no
1	youth	high	no	excellent	no
2	middle_aged	high	no	fair	yes
3	senior	medium	no	fair	yes
4	senior	low	yes	fair	yes
5	senior	low	yes	excellent	no
6	middle_aged	low	yes	excellent	yes
7	youth	medium	no	fair	no
8	youth	low	yes	fair	yes
9	senior	medium	yes	fair	yes
10	youth	medium	yes	excellent	yes
11	middle_aged	medium	no	excellent	yes
12	middle_aged	high	yes	fair	yes
13	senior	medium	no	excellent	no

age 연령대로 구분해보면 될까요?

	Yes	No
Youth	2	3
Middle + Senior	7	2

Sklearn에서 제공하는 특정 함수는 Binary Splitting만 허용



$$\text{Gini} = 1 - \sum_{i=1}^{K} {p_k}^2$$

age = youth

(D$_{\text{Group A}}$/D) * Gini$_{\text{Group A}}$ + (D$_{\text{Group ~A}}$/D) * Gini$_{\text{Group ~A}}$

$$G(age = youth) = \frac{5}{14}\left(1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2\right) + \frac{9}{14}\left(1 - \left(\frac{7}{9}\right)^2 - \left(\frac{2}{9}\right)^2\right) = 0.394$$

$$G(age = middle) = \frac{4}{14}\left(1 - \left(\frac{4}{4}\right)^2\right) + \frac{10}{14}\left(1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2\right) = \mathbf{0.357}$$

$$G(age = senior) = \frac{5}{14}\left(1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right) + \frac{9}{14}\left(1 - \left(\frac{6}{9}\right)^2 - \left(\frac{3}{9}\right)^2\right) = 0.457$$

Iris 아이리스, 붓꽃

Iris setosa　Iris versicolor　Iris virginica

꽃 세가지 종류(0:Setosa, 1:Versicolor, 2:Virginica)

숫자 cm: Petal(꽃잎) 길이, Petal 폭

https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

Sir Ronald Aylmer Fisher (1936)

LogisticRegressionGD

[1] 하고싶은 메쏘드가 무엇인가?

[2] 그런 메쏘드들을

틀로 묶어서 마련해둔 것은 무엇인가

```
X_train_01_subset = X_train[(y_train == 0) | (y_train == 1)]
y_train_01_subset = y_train[(y_train == 0) | (y_train == 1)]

lrgd = LogisticRegressionGD(eta=0.05, n_iter=1000, random_state=1)
lrgd.fit(X_train_01_subset,
         y_train_01_subset)
```

[1]

method

정의 완료

```python
def fit(self, X, y):
    """훈련 데이터 학습
    """
    rgen = np.random.RandomState(self.random_state)
    # 표준편차(scale)가 0.01인 정규 분포에서 뽑은 랜덤한 작은수
    self.w_ = rgen.normal(loc=0.0, scale=0.01, size=1 + X.shape[1])
    self.cost_ = []

    for i in range(self.n_iter):
        net_input = self.net_input(X)
        output = self.activation(net_input)
        errors = (y - output)
        self.w_[1:] += self.eta * X.T.dot(errors)
        self.w_[0] += self.eta * errors.sum()

        # 오차 제곱합 대신 로지스틱 비용을 계산합니다.
        cost = -y.dot(np.log(output)) - ((1 - y).dot(np.log(1 - output)))
        self.cost_.append(cost)
    return self
```

```python
X_train_01_subset = X_train[(y_train == 0) | (y_train == 1)]
y_train_01_subset = y_train[(y_train == 0) | (y_train == 1)]

lrgd = LogisticRegressionGD(eta=0.05, n_iter=1000, random_state=1)
lrgd.fit(X_train_01_subset,
         y_train_01_subset)
```

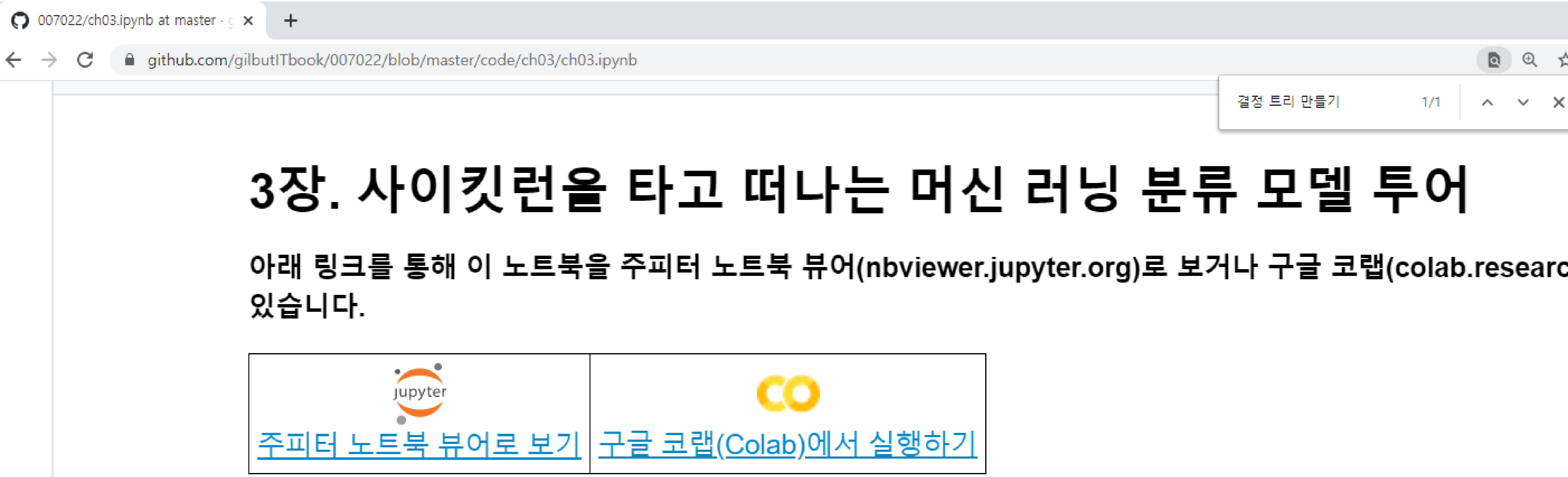
[2]

메쏘드를 묶은 이름 class

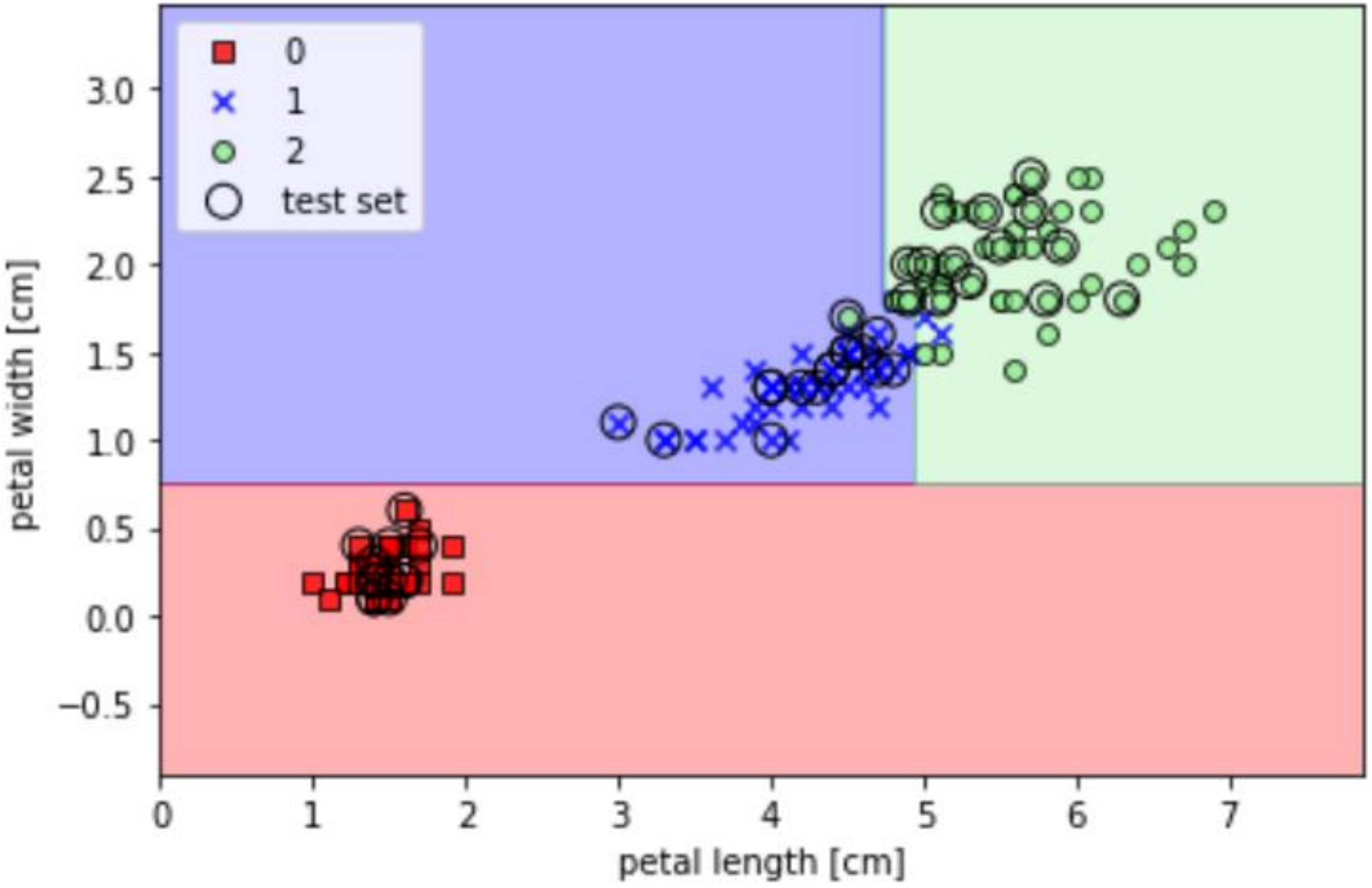
```
class LogisticRegressionGD(object):
    """경사 하강법을 사용한 로지스틱 회귀 분류기
    """
        def fit(self, X, y):
            """훈련 데이터 학습
            """
            rgen = np.random.RandomState(self.random_state)
            # 표준편차(scale)가 0.01인 정규 분포에서 뽑은 랜덤한 작은수
            self.w_ = rgen.normal(loc=0.0, scale=0.01, size=1 + X.shape[1])
            self.cost_ = []

            for i in range(self.n_iter):
                net_input = self.net_input(X)
                output = self.activation(net_input)
                errors = (y - output)
                self.w_[1:] += self.eta * X.T.dot(errors)
                self.w_[0] += self.eta * errors.sum()

                # 오차 제곱합 대신 로지스틱 비용을 계산합니다.
                cost = -y.dot(np.log(output)) - ((1 - y).dot(np.log(1 - output)))
                self.cost_.append(cost)
            return self
```

```
X_train_01_subset = X_train[(y_train == 0) | (y_train == 1)]
y_train_01_subset = y_train[(y_train == 0) | (y_train == 1)]

lrgd = LogisticRegressionGD(eta=0.05, n_iter=1000, random_state=1)
lrgd.fit(X_train_01_subset,
         y_train_01_subset)
```

sklearn 학습 챕터에서 '결정 트리 만들기' 검색



In [32]:
```python
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(criterion='gini',
                              max_depth=4,
                              random_state=1)
tree.fit(X_train, y_train)

X_combined = np.vstack((X_train, X_test))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X_combined, y_combined,
                      classifier=tree, test_idx=range(105, 150))

plt.xlabel('petal length [cm]')
plt.ylabel('petal width [cm]')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```




```
[10] print("특성 중요도:\n{}".format(tree.feature_importances_))
```

```
특성 중요도:
[0.42708333 0.57291667]
```

In [33]:
```python
from pydotplus import graph_from_dot_data
from sklearn.tree import export_graphviz

dot_data = export_graphviz(tree,
                           filled=True,
                           rounded=True,
                           class_names=['Setosa',
                                        'Versicolor',
                                        'Virginica'],
                           feature_names=['petal length',
                                          'petal width'],
                           out_file=None)
graph = graph_from_dot_data(dot_data)
graph.write_png('tree.png')
```

[1] Setosa: 꽃잎 너비 <= 0.75 (35 cases) 한번에!

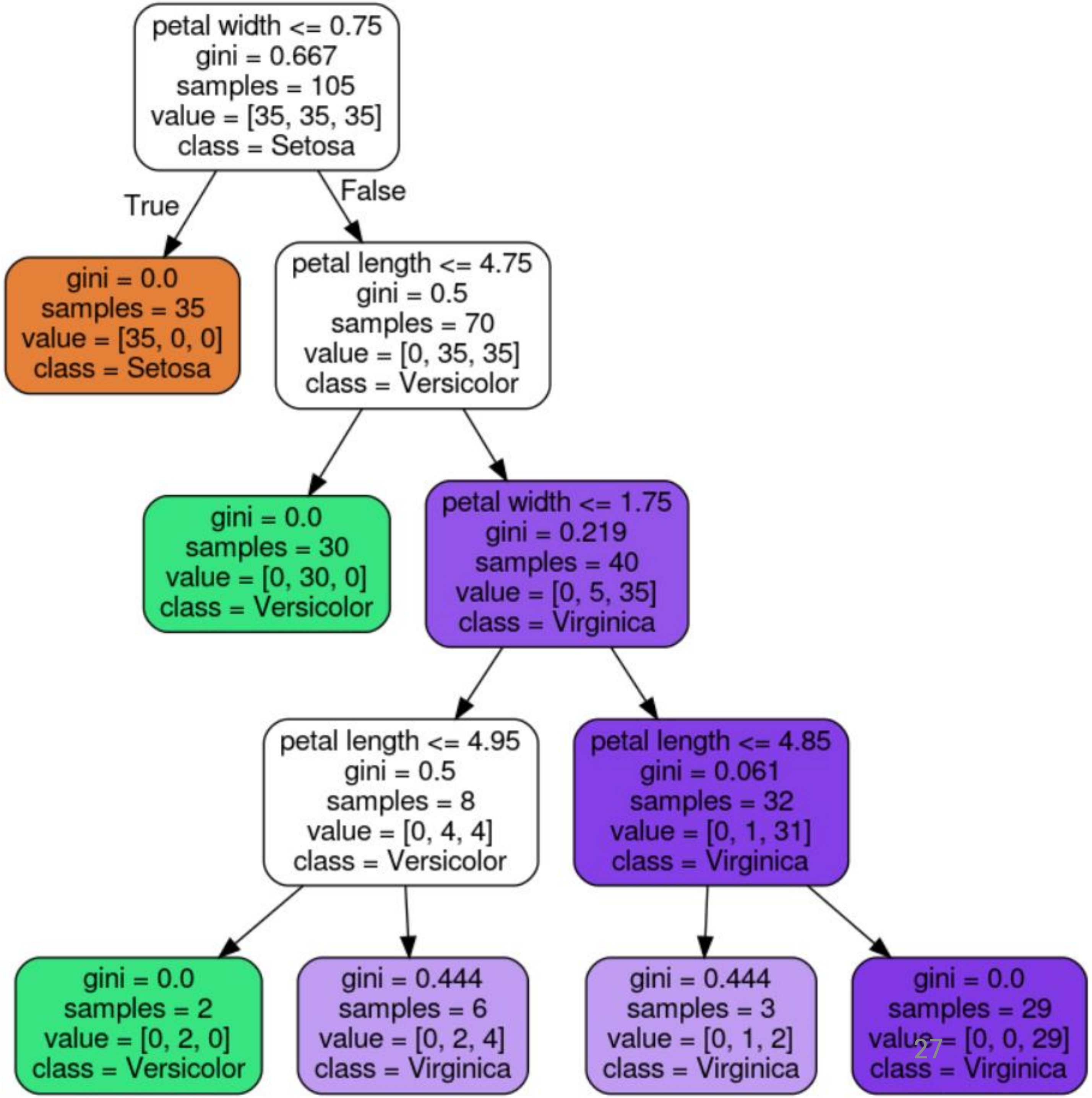
[2] Versicolor:

Depth 1 - 꽃잎 너비 > 0.75 (70 cases)
Depth 2 - 꽃잎 길이 <= 4.75 (30 cases)
Depth 3 - 꽃잎 너비 <= 1.75 (8 cases)
Depth 4 - 꽃잎 길이 <= 4.95 (2 cases)

[3] Virginica

Depth 1 - 꽃잎 너비 > 0.75 (0 cases)
Depth 2 - 꽃잎 길이 > 4.75 (40 cases)
Depth 3 - 꽃잎 너비 > 1.75 (32 cases)
Depth 4 - 꽃잎 길이 > 4.95 (2+4 cases)

Gini coefficient 어떤가요?



sklearn.model_selection.cross_val_s

```
sklearn.model_selection.cross_val_score(estimator, X, y=None, *, groups=None, scoring=Non
verbose=0, fit_params=None, pre_dispatch='2*n_jobs', error_score=nan) ¶
```

Evaluate a score by cross-validation

Read more in the User Guide.

cv : *int, cross-validation generator or an iterable*

Determines the cross-validation splitting strategy

- None, to use the default 5-fold cross validatio
- int, to specify the number of folds in a `(Strat`

Parameters:

estimator : *estimator object implementing 'fit'*

The object to use to fit the data.

X : *array-like of shape (n_samples, n_features)*

The data to fit. Can be for example a list, or an array.

y : *array-like of shape (n_samples,) or (n_samples, n_outputs), default=N*

The target variable to try to predict in the case of supervised learning.

DT Accuracy 정확도

cv=10-fold

대개, cv의 결과(여기선 10개)를 평균하여 표현함

```
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0)
cross_val_score(clf, X_train, y_train, cv=10)
# iris = load_iris()
# cross_val_score(clf, iris.data, iris.target, cv=10)
```

```
array([1.        , 0.90909091, 0.90909091, 1.        , 0.90909091,
       1.        , 0.9       , 0.9       , 0.9       , 0.9       ])
```

랜덤 포레스트로 여러 개의 결정 트리 연결

```
from sklearn.ensemble import RandomForestClassifier

forest = RandomForestClassifier(criterion='gini',
                                n_estimators=25,
                                random_state=1,
                                n_jobs=2)
forest.fit(X_train, y_train)
```

The default value of `n_estimators` changed from 10 to 100 in 0.22.

researchgate.net/publication/

ResearchGate

Search for

See all › 157 Citations

See all › 47 References

Chapter

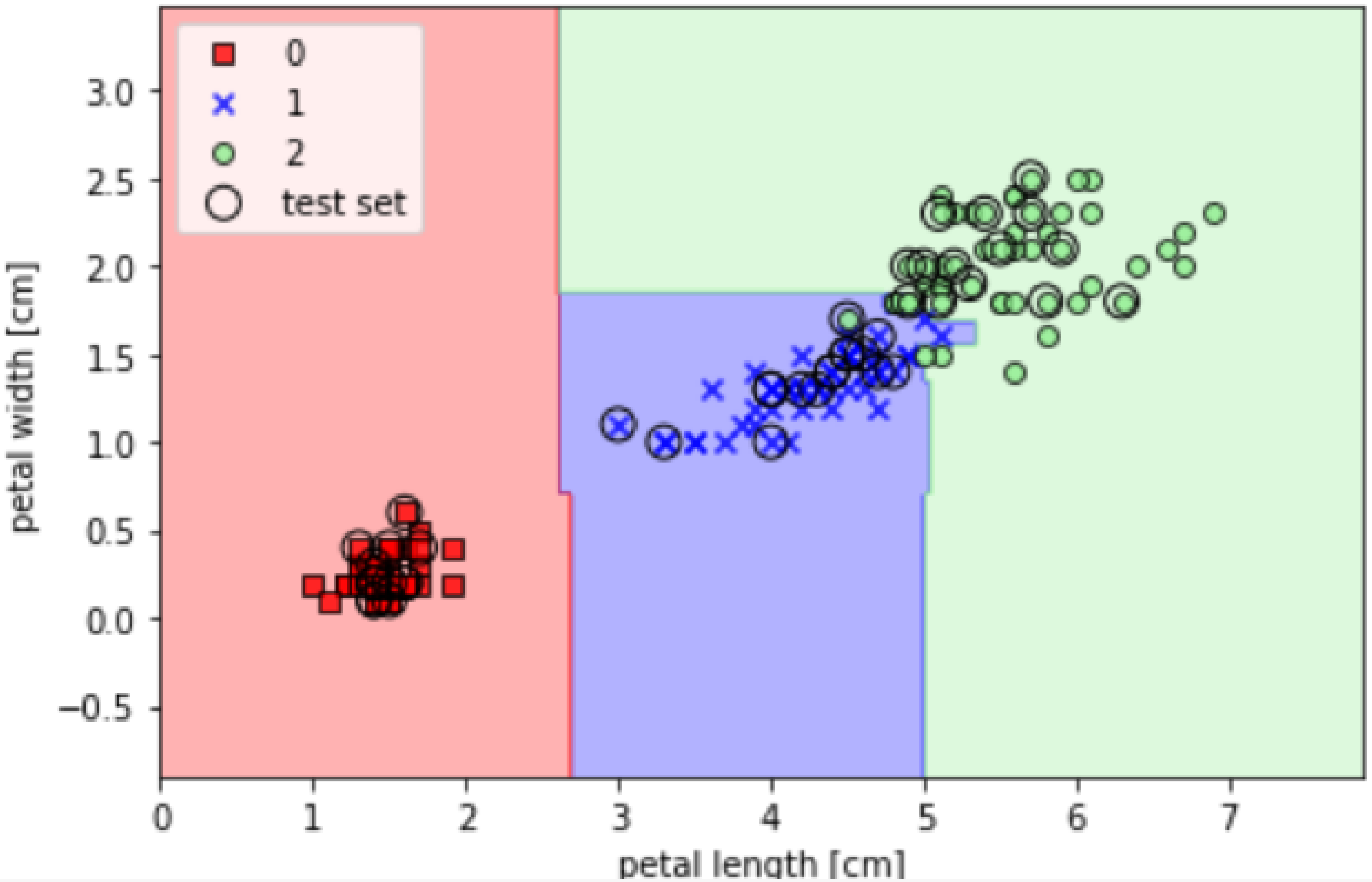
from book Machine learning and data mining in pattern recognition. 8th international conference, MLDM 2012, Berlin, Germany, July 13–20, 2012. Proceedings

How Many Trees in a Random Forest?

Conference Paper · July 2012 *with* 22,293 Reads

랜덤 포레스트로 여러 개의 결정 트리 연결하기

```
from sklearn.ensemble import RandomForestClassifier

forest = RandomForestClassifier(criterion='gini',
                                n_estimators=25,
                                random_state=1,
                                n_jobs=2)
forest.fit(X_train, y_train)

plot_decision_regions(X_combined, y_combined,
                      classifier=forest, test_idx=range(105, 150))

plt.xlabel('petal length [cm]')
plt.ylabel('petal width [cm]')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```




```
[23] print("특성 중요도:₩n{}".format(forest.feature_importances_))
```

```
특성 중요도:
[0.52140135 0.47859865]
```

RF Accuracy 정확도

```
[17] from sklearn.model_selection import cross_val_score
     from sklearn.ensemble import RandomForestClassifier

     forest = RandomForestClassifier(criterion='gini',
                                     n_estimators=25,
                                     random_state=1,
                                     n_jobs=2)
     cross_val_score(forest, X_train, y_train, cv=10)
```

```
array([1. , 1. , 1. , 1. , 1. , 1. , 0.9, 0.9, 0.9, 0.9])
```

```
from sklearn.model_selection import cross_val_score
```

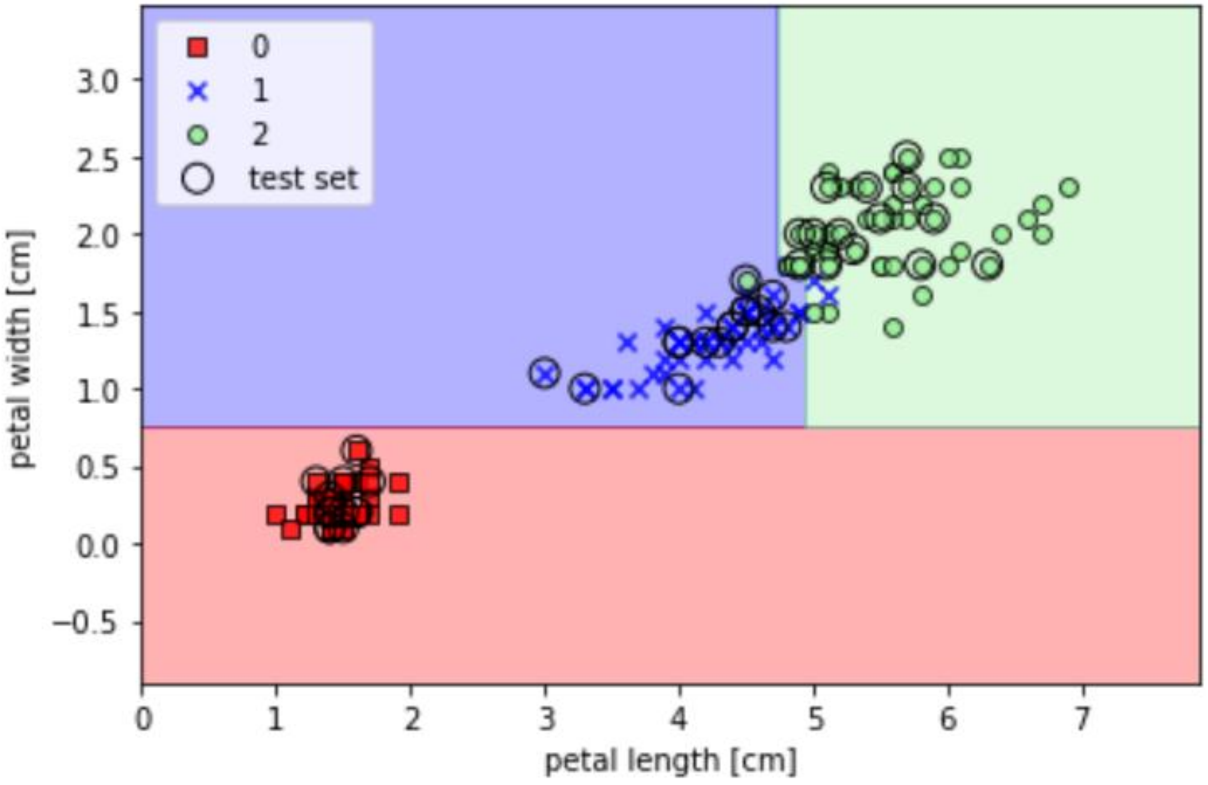
DT vs RF Accuracy 정확도

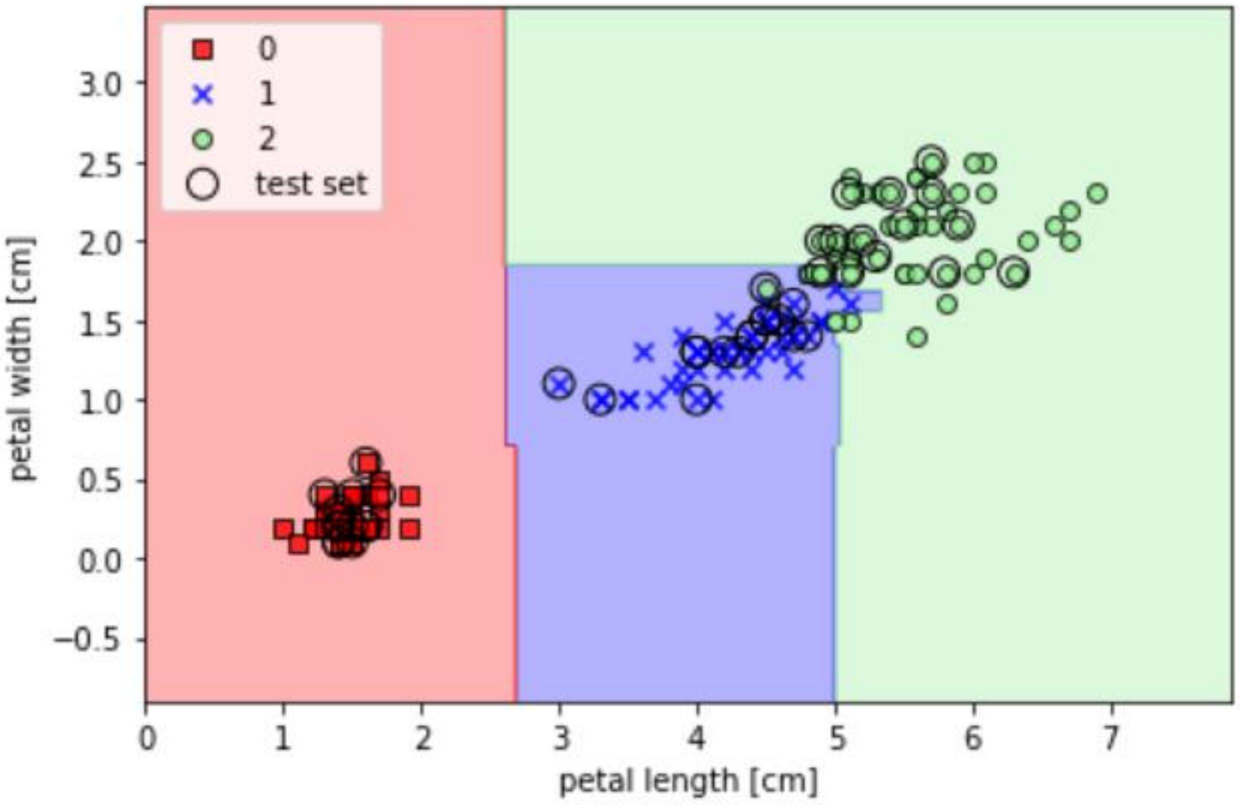
```
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0)
cross_val_score(clf, X_train, y_train, cv=10)
# iris = load_iris()
# cross_val_score(clf, iris.data, iris.target, cv=10)
```

```
array([1.        , 0.90909091, 0.90909091, 1.        , 0.90909091,
       1.        , 0.9       , 0.9       , 0.9       , 0.9       ])
```

```
[17] from sklearn.model_selection import cross_val_score
     from sklearn.ensemble import RandomForestClassifier

     forest = RandomForestClassifier(criterion='gini',
                                     n_estimators=25,
                                     random_state=1,
                                     n_jobs=2)
     cross_val_score(forest, X_train, y_train, cv=10)
```

```
array([1. , 1. , 1. , 1. , 1. , 1. , 0.9, 0.9, 0.9, 0.9])
```






```
print("특성 중요도:₩n{}".format(tree.feature_importances_))
```

```
특성 중요도:
[0.42708333 0.57291667]
```

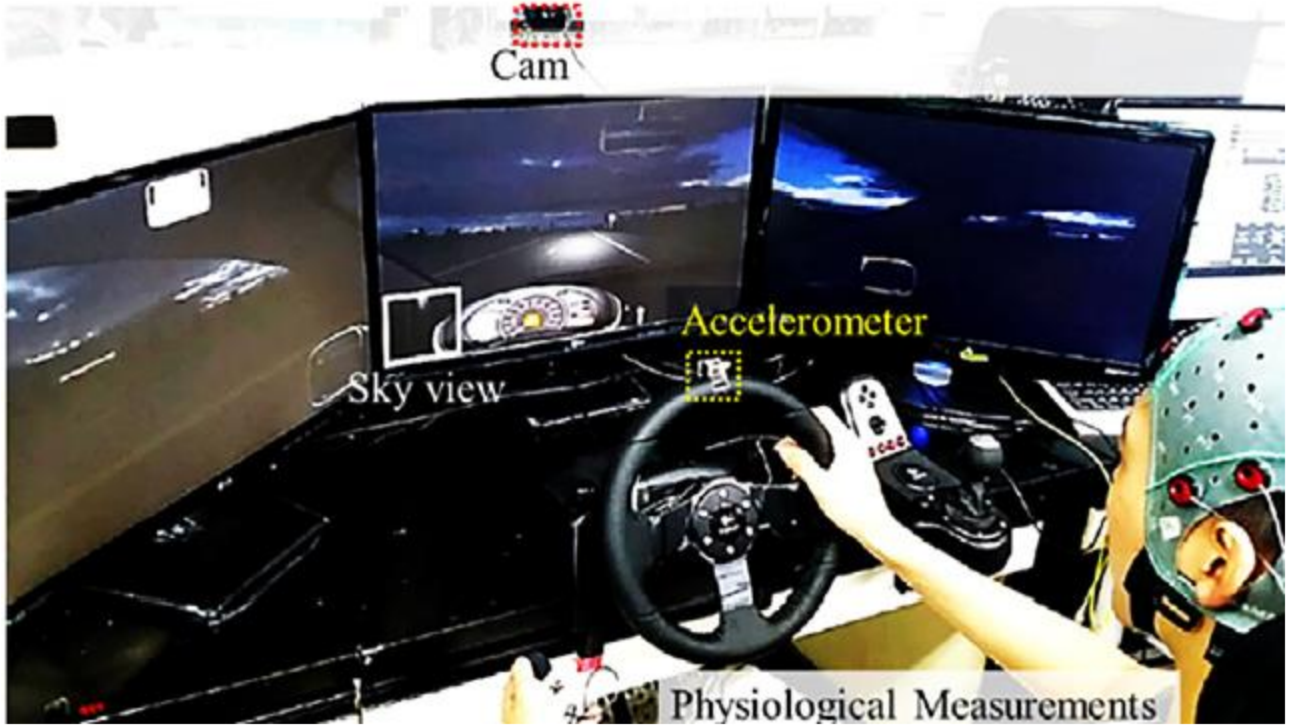
```
[23] print("특성 중요도:₩n{}".format(forest.feature_importances_))
```

```
특성 중요도:
[0.52140135 0.47859865]
```

훈련할 특징Feature 들의
학습 순서에 따라
결과도 달라질 수 있겠네요?

‘Feature Filter’ approach

- 특징들의 우선순위가 ‘바라보는 시각’에 따라 다를 수 있음
- 뇌전도 (Electro+Encephalo+Gram) = 뇌파 측정 결과



참조 Reference

'Feature Filter' approach

- 바라보는 시각 = Filter (다양한 정보이론 수식 들)
- 뇌전도 (Electro+Encephalo+Gram) 특징들의 순위가 다름.

Top seven ranked features of the ear canal EEG from each ranking method for the best kappa coefficient. P denotes the power.

Ranking	1	2	3	4	5	6	7
Relief-F	EEG_PmEn	EEG_HFD	EEG_H	delta_GF	EEG_SpEn	theta/alpha	alpha/beta
Mutual-I	EEG_HFD	EEG_PmEn	EEG_SpEn	EEG_H	alpha/beta	alpha_BandP	theta_BandP
Fisher-S	EEG_PmEn	EEG_HFD	theta_GF	EEG_H	theta_PeakP	EEG_SpEn	beta_GF
Composite	EEG_PmEn	EEG_HFD	EEG_H	EEG_SpEn	delta_GF	theta_GF	alpha/beta