scikit-learn

- 사이킷 런은 파이썬에서 사용할 수 있는 머신 러닝 framework이다.
- cf) Tensorflow : 딥러닝 프레임워크

- 참조 사이트 : http://scikit-learn.org/stable/index.html

- 특징
  - 다양한 분류기를 지원한다.
  - 머신 러닝의 결과를 검증하는 기능도 있다.(교차 검증)
  - 다양한 알고리즘 지원
    - 분류(Classification)
    - 회귀(Regression)
    - 클러스터링(Clustering)
    - 차원 축소(Dimensionality Reduction)

XOR 연산해보기

- 사이킷 런을 이용하여 XOR 연산을 수행해본다.
- 파일 이름 : xor_train_01.py

```
from sklearn import svm

# XOR의 계산 결과 데이터 --- (※1)
#P, Q, result
xor_data = [ # 입력과 출력을 한 꺼번에 작성
      [0, 0, 0],
      [0, 1, 1],
      [1, 0, 1],
      [1, 1, 0]
]

# 학습을 위해 데이터와 레이블 분리하기 --- (※2)
x = [] # 입력 데이터
y = [] # 출력 데이터

for row in xor_data:
      prediction = row[0]
      q = row[1]
      r = row[2]
      x.append([prediction, q])
      y.append( r )
```

x1	x2	y
0	0	0
1	0	1
0	1	1
1	1	0

```
# 데이터 학습시키기 --- (※3)
# svm 알고리즘을 이용하여 머신 러닝 수행
clf = svm.SVC()

# fit(학습용_데이터, 레이블_데이터) 함수 : 데이터를 학습시킨다.
clf.fit(x, y)

# 데이터 예측하기 --- (※4)
# predict(예측하고자_하는_데이터)
pre = clf.predict(x)

print("예측 결과:", pre)
# 결과 확인하기 --- (※5)
ok = 0; total = 0
for idx, answer in enumerate(y):
      prediction = pre[idx]
      if prediction == answer: ok += 1
      total += 1

print("정답률:", ok, "/", total, "=", ok/total)
```

```
# 출력 결과
예측 결과: [0 1 1 0]
정답률: 4 / 4 = 1.0
```

XOR 연산해보기

- scikit learn을 이용한 데이터와 레이블 분리하고, 정답률 계산 함수 사용해보기
- 파일 이름 : xor_train_02.py

```
import pandas as pd

from sklearn import svm, metrics

# XOR 연산
xor_input = [
      [0, 0, 0],
      [0, 1, 1],
      [1, 0, 1],
      [1, 1, 0]
]

# 입력을 학습 전용 데이터와 테스트 전용 데이터로 분류하기 --- (※1)
# 데이터 프레임으로 만든다.
xor_df = pd.DataFrame(xor_input)

# 슬라이싱을 이용하여 데이터와 레이블을 분리한다.
x   = xor_df.ix[:,0:1] # 데이터
y = xor_df.ix[:,2]    # 레이블

# 데이터 학습과 예측하기 --- (※2)
# svm 알고리즘을 이용하여 머신 러닝 수행
clf = svm.SVC()

# fit(학습용_데이터, 레이블_데이터) 함수 : 데이터를 학습시킨다.
clf.fit(x, y)
```

```
# predict(예측하고자_하는_데이터)
pre = clf.predict(x)

# 정답률 구하기 --- (※3)
# accuracy_score 함수를 이용하면 정답률을 쉽게 구해준다.
# accuracy_score(정답, 예측_결과_배열)
ac_score = metrics.accuracy_score(y, pre)

print("정답률 =", ac_score)
```

출력 결과

```
정답률 = 1.0
```

머신 러닝으로 붓꽃 품종 분류하기

- 꽃잎과 꽃받침의 길이와 폭을 이용하여 붓꽃의 품종을 분류하는 예시이다.
- 파일 이름 : iris_train_01.py, iris.csv

```
# 머신 러닝으로 붓꽃 품종 분류하기
from sklearn import svm, metrics
import random, re

# 붓꽃의 CSV 데이터 읽어 들이기 --- (※1)
csv = []

with open('iris.csv', 'r', encoding='utf-8') as fp:
    # 한 줄씩 읽어 들이기
    for line in fp:
        line = line.strip()     # 줄바꿈 제거
        cols = line.split(',') # 쉼표로 자르기
        # 문자열 데이터를 숫자로 변환하기
        fn = lambda n : float(n) if re.match(r'^[0-9\.]+$', n)
else n
        cols = list(map(fn, cols))
        csv.append(cols)

# 가장 앞 줄의 헤더 제거
del csv[0]

# 데이터 셔플하기(섞기) --- (※2)
random.shuffle(csv)
```

```
# 학습 전용 데이터와 테스트 전용 데이터 분할하기(2:1 비율) --- (※3)
total_len = len(csv)
train_len = int(total_len * 2 / 3) # 100개와 50개

x_train = []
y_train = []
x_test = []
y_test = []

for i in range(total_len):
    data  = csv[i][0:4]

    label = csv[i][4] # 품종

    if i < train_len:
        x_train.append(data)
        y_train.append(label)
    else:
        x_test.append(data)
        y_test.append(label)

# 데이터를 학습시키고 예측하기 --- (※4)
clf = svm.SVC()
```

머신 러닝으로 붓꽃 품종 분류하기

- 꽃잎과 꽃받침의 길이와 폭을 이용하여 붓꽃의 품종을 분류하는 예시이다.
- 파일 이름 : iris_train_01.py, iris.csv

```
# fit(학습용_데이터, 레이블_데이터) 함수 : 데이터를 학습시킨다.
clf.fit(x_train, y_train)

prediction = clf.predict(x_test)

# 정답률 구하기 --- (※5)
# accuracy_score 함수를 이용하면 정답률을 쉽게 구해준다.
# accuracy_score(정답, 예측_결과_배열)
ac_score = metrics.accuracy_score(y_test, prediction)

print("정답률 =", ac_score)

cl_report = metrics.classification_report( y_test, prediction)

print("\n리포트 =", cl_report)
```

```
출력 결과
정답률 = 0.96

리포트 =                    precision    recall  f1-score
support

     Iris-setosa       1.00      1.00      1.00        15
Iris-versicolor       0.94      0.94      0.94        18
 Iris-virginica       0.94      0.94      0.94        17

      avg / total       0.96      0.96      0.96        50
```

머신 러닝으로 붓꽃 품종 분류하기

- scikit learn에는 훈련 전용데이터와 테스트 전용 데이터로 분할해주는 train_test_split 메소드가 있다.
- 파일 이름 : iris_train_02.py, iris.csv

```
import pandas as pd
from sklearn import svm, metrics
from sklearn.model_selection import train_test_split

# 붓꽃의 CSV 데이터 읽어 들이기 --- (※1)
csv = pd.read_csv('iris.csv')

# 필요한 열 추출하기 --- (※2)
csv_data =
csv[["SepalLength","SepalWidth","PetalLength","PetalWidth"]]

csv_label = csv["Name"]

# 학습 전용 데이터와 테스트 전용 데이터로 나누기 --- (※3)
x_train, x_test, y_train, y_test = \
      train_test_split(csv_data, csv_label)

# 데이터 학습시키고 예측하기 --- (※4)
clf = svm.SVC()

clf.fit(x_train, y_train)

prediction = clf.predict(x_test)
```

```
# 정답률 구하기 --- (※5)
ac_score = metrics.accuracy_score(y_test, prediction)
print("정답률 =", ac_score)

cl_report = metrics.classification_report( y_test, prediction)

print("\n리포트 =", cl_report)
```

```
출력 결과
정답률 = 0.973684210526

리포트 =                    precision    recall  f1-score
support

     Iris-setosa       1.00      1.00      1.00         8
Iris-versicolor        0.93      1.00      0.96        13
 Iris-virginica        1.00      0.94      0.97        17

     avg / total       0.98      0.97      0.97        38
```

서포트 벡터 머신

- 참조 사이트
  - http://cafe.naver.com/ugcadman/1203

서포트 벡터 머신

- 서포트 벡터 머신을 사용하기 위하여 샘플용 엑셀 파일을 만들어 본다.
- 파일 이름 : bmi_create.py

```
import random

# 무작위로 2만명의 데이터 만들기

# BMI를 계산해서 레이블을 리턴하는 함수
# BMI 공식 : 몸무게 / ( 키 * 키 )
def calc_bmi(h, w):
    bmi = w / (h/100) ** 2
    if bmi < 18.5: return "thin"
    if bmi < 25: return "normal"
    return "fat"

# 출력 파일 준비하기
fp = open("bmi.csv", "w", encoding="utf-8")

fp.write("height,weight,label\n")

# 무작위로 데이터 생성하기
cnt = {"thin":0, "normal":0, "fat":0}
```

```
for i in range(20000):
    h = random.randint(120,200)
    w = random.randint(35, 80)

    label = calc_bmi(h, w) # 함수 호출

    cnt[label] += 1

    fp.write("{0},{1},{2}\n".format(h, w, label))

fp.close()

print("ok,", cnt)
```

출력 결과

```
ok, {´normal´: 5933, ´fat´: 7593, ´thin´: 6474}
```

서포트 벡터 머신

- 키와 몸무게 데이터를 이용하여 SVM으로 학습시켜 보고, 비만을 정확하게 맞출 수 있는지 테스트 해보도록 한다.
- 파일 이름 : bmi_test.py, bmi.csv

```
from sklearn import svm, metrics
from sklearn.model_selection import train_test_split
import pandas as pd

# 키와 몸무게 데이터 읽어 들이기 --- (※1)
tbl = pd.read_csv("bmi.csv")

# 칼럼(열)을 자르고 정규화하기 --- (※2)
label = tbl["label"]

w = tbl["weight"] / 100 # 최대 100kg라고 가정
h = tbl["height"] / 200 # 최대 200cm라고 가정
wh = pd.concat([w, h], axis=1)

# 학습 전용 데이터와 테스트 전용 데이터로 나누기 --- (※3)
x_train, x_test, y_train, y_test = \
            train_test_split(wh, label)

# 데이터 학습하기 --- (※4)
clf = svm.SVC()

clf.fit(x_train, y_train)
```

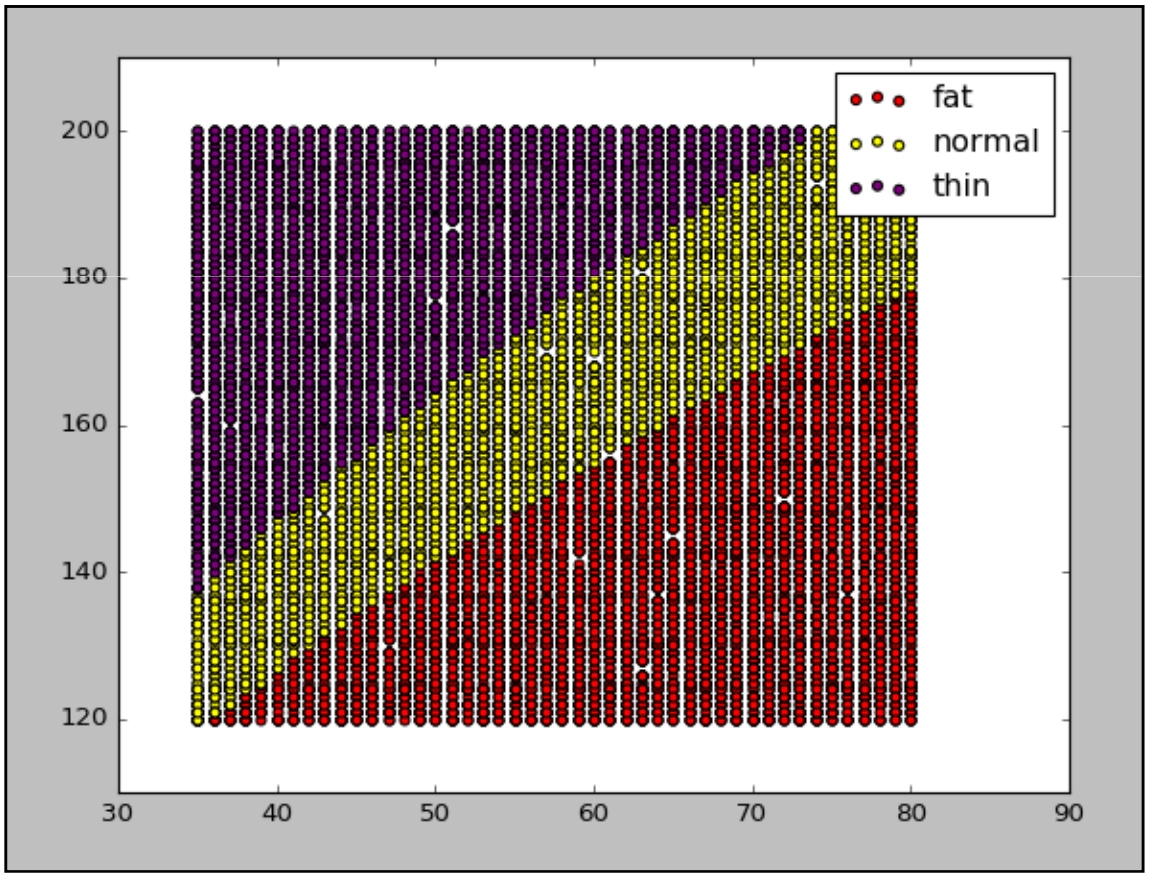
```
# 데이터 예측하기 --- (※5)
predict = clf.predict(x_test)

# 결과 테스트하기 --- (※6)
ac_score = metrics.accuracy_score(y_test, predict)

print("정답률 =", ac_score)

cl_report = metrics.classification_report(y_test, predict)

print("\n리포트 =\n", cl_report)
```

```
정답률 = 0.9896

리포트 =
              precision    recall  f1-score   support

         fat       1.00      0.98      0.99      1852
      normal       0.97      1.00      0.98      1488
        thin       1.00      0.99      0.99      1660

avg / total       0.99      0.99      0.99      5000
```

데이터 분포 확인하기

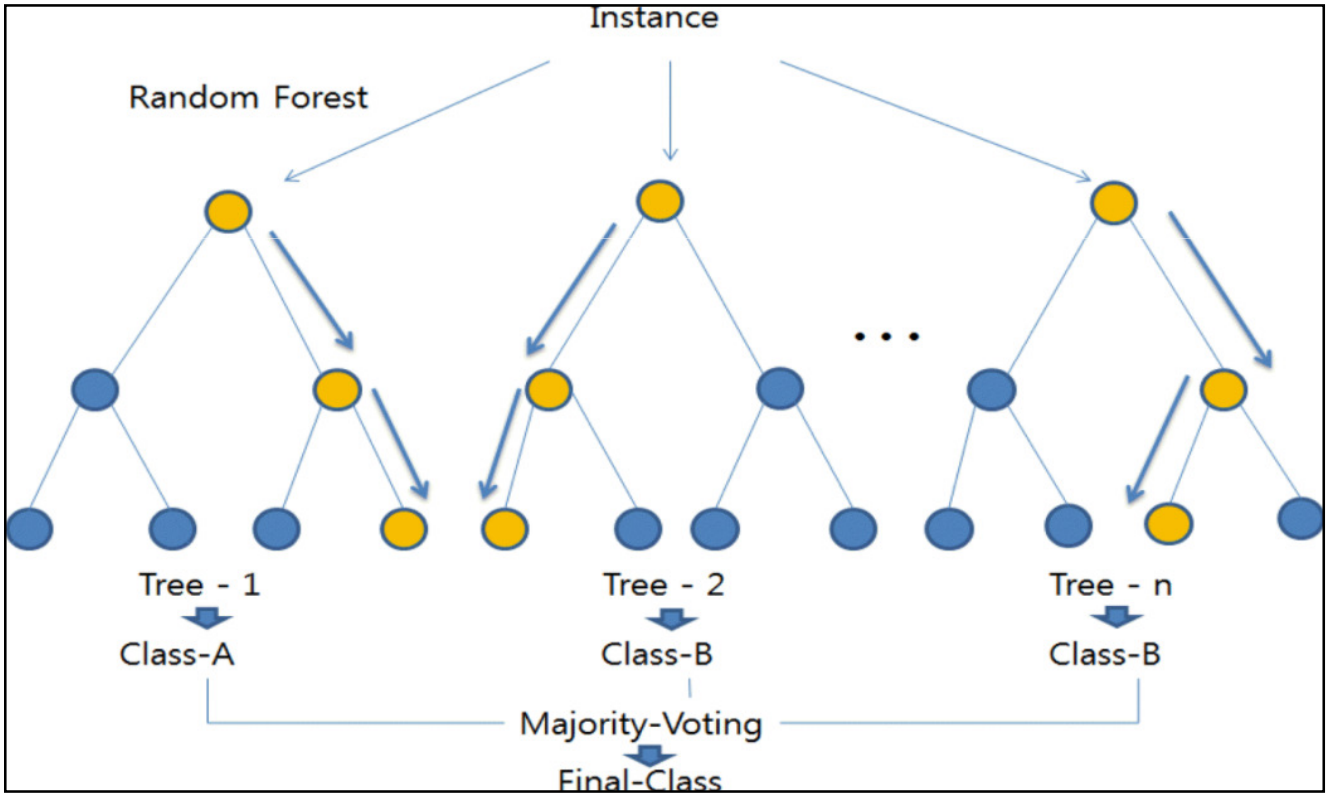
- 일반적으로 데이터의 분포는 산포도를 그려보면 알 수 있다.
- BMI 공식을 이용하여 데이터를 만들었으므로 당연히 분포가 잘 되었겠지만 이를 그림으로 한 번 그려 보도록 한다.
- 파일 이름 : bmi_plot.py

```
import matplotlib.pyplot as plt
import pandas as pd

# Pandas로 CSV 파일 읽어 들이기
tbl = pd.read_csv("bmi.csv", index_col=2)

# 그래프 그리기 시작
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

# 서브 플롯 전용 - 지정한 레이블을 임의의 색으로 칠하기
def scatter(lbl, color):
    b = tbl.loc[lbl]
    ax.scatter(b["weight"],b["height"], c=color, label=lbl)

scatter("fat",    "red")
scatter("normal", "yellow")
scatter("thin",   "purple")
ax.legend()

plt.savefig("bmi-test.png")
plt.show()
```



랜덤 포레스트

- 랜덤 포레스트는 2001년에 레오 브라이만이 제안한 머신 러닝 알고리즘이다.
- 집단 학습을 기반으로 고정밀 분류, 회귀, 클러스터링 등을 구현하는 것이다.
- 학습 전용 데이터를 기반으로 다수의 의사 결정 트리를 만들고, 그것을 토대로 다수결로 결과를 도출하는 기법이다.



버섯 데이터 셋

- UCI 머신 러닝 레포지트리에 있는 독버섯과 관련 데이터를 이용하여 머신 러닝 학습하기
- 8,124 종류의 버섯의 특징과 독이 있는 지 등의 정보가 들어 있는 데이터 셋이다.
- 이것을 csv 형식으로 다운로드 하여 학습을 진행하도록 한다.
- 컬럼에 대한 세부 설명은 사이트를 참조하도록 한다.
- 참조 사이트
  - https://archive.ics.uci.edu/ml/datasets/Mushroom

Mushroom Data Set

Download: Data Folder, Data Set Description

Abstract: From Audobon Society Field Guide; mushrooms described in terms of physical characteristics; classification: poisonous or edible

Data Set Characteristics:	Multivariate	**Number of Instances:**	8124	**Area:**	Life
Attribute Characteristics:	Categorical	**Number of Attributes:**	22	**Date Donated**	1987-04-27
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	306498

버섯 데이터 셋 다운로드

- 다음 코드를 이용하여 데이터를 로컬 컴퓨터에 다운로드 받도록 한다.
- 파일 이름 : mushroom_download.py

```
import urllib.request as req
local= "mushroom.csv"

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data"

req.urlretrieve(url, local)

print("ok")
```

버섯 분류하기

- 랜덤 포레스트를 사용해 버섯을 분류해주는 프로그램을 작성하시오.
- 파일 이름 : mushroom_train_01.py

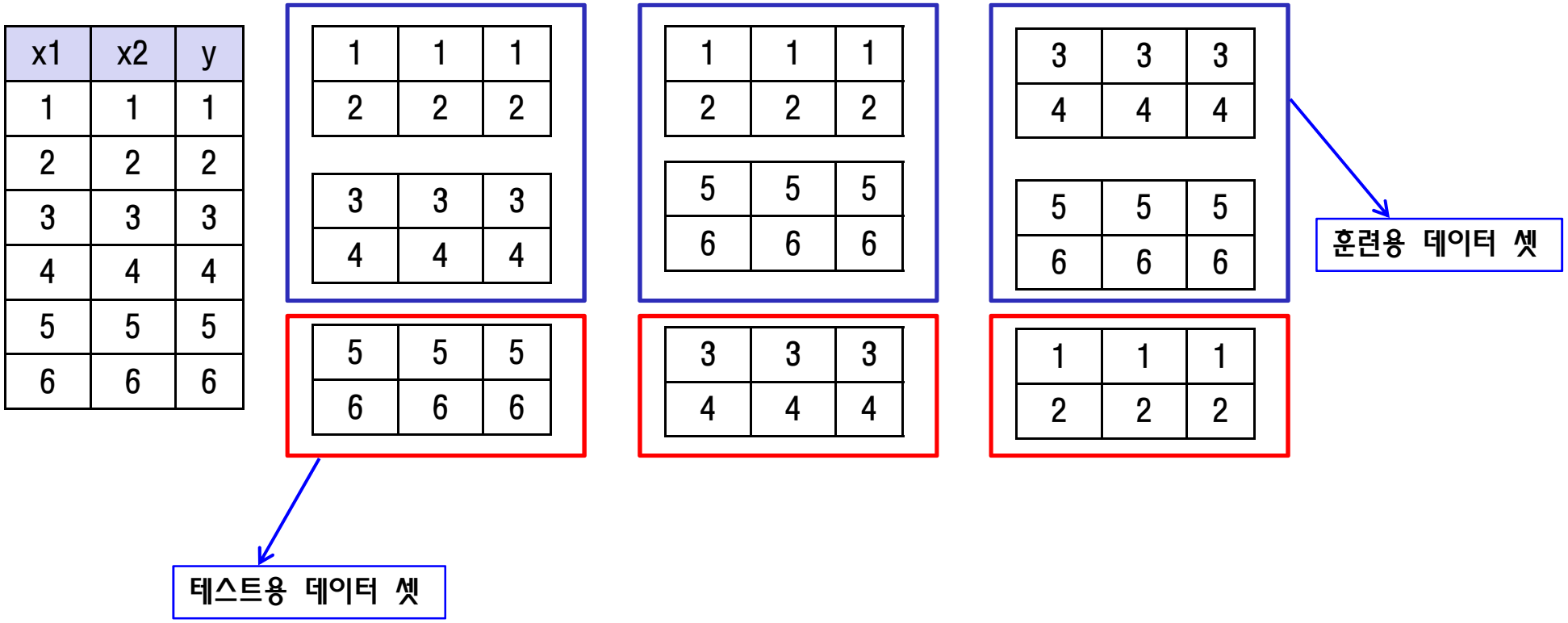
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.model_selection import train_test_split

# 데이터 읽어 들이기--- (※1)
mr = pd.read_csv("mushroom.csv", header=None)

# 데이터 내부의 기호를 숫자로 변환하기--- (※2)
y = []
x = []

for row_index, row in mr.iterrows():
    y.append(row.ix[0]) # 독의 유무(p_독성, e_식용)

    row_data = []

    for v in row.ix[1:]:
        row_data.append(ord(v))
    x.append(row_data)

# 학습 전용과 테스트 전용 데이터로 나누기 --- (※3)
x_train, x_test, y_train, y_test = \
    train_test_split(x, y)
```

```
# 데이터 학습시키기 --- (※4)
clf = RandomForestClassifier()

clf.fit(x_train, y_train)

# 데이터 예측하기 --- (※5)
predict = clf.predict(x_test)

# 결과 테스트하기 --- (※6)
ac_score = metrics.accuracy_score(y_test, predict)

print("정답률 =", ac_score)

cl_report = metrics.classification_report(y_test, predict)

print("리포트 =\n", cl_report)
```

```
출력 하기
정답률 = 1.0
리포트 =
             precision    recall  f1-score   support

          e       1.00      1.00      1.00      1055
          p       1.00      1.00      1.00       976

avg / total       1.00      1.00      1.00      2031
```

교차 검증

- 머신 러닝 모델의 타당성을 검증하는 방법 중의 하나로 Cross-validation이라는 것이 있다.
- 특정 데이터를 훈련용과 테스트용으로 분할하여 각각 학습과 테스트를 수행하여 그 타당성을 검증하는 방법이다.
- K 분할 교차 검증(K-fold cross validation)
- K = 3이라고 하면, 3 그룹의 분류 정밀도를 구하여 이것의 평균 분류 정밀도를 구하는 방식이다.



교차 검증 사용해보기

- 붓꽃 데이터를 사용하여 교차 검증하는 프로그램을 작성하세요.
- K의 값은 5로 설정하도록 한다.
- 파일 이름 : cross_iris_01.py

```
from sklearn import svm, metrics
import random, re

# 붓꽃의 CSV 파일 읽어 들이기 --- (※1)
lines = open('iris.csv', 'r', encoding='utf-8').read().split("\n")

f_tonum = lambda n : float(n) if re.match(r'^[0-9\.]+$', n) else
n

f_cols  = lambda li: list(map(f_tonum,li.strip().split(',')))
csv = list(map(f_cols, lines))

del csv[0] # 헤더 제거하기

random.shuffle(csv) # 데이터 섞기

# 데이터를 K개로 분할하기 --- (※2)
K = 5
csvk = [ [] for i in range(K) ]
# print(csvk) # 5개의 리스트 [[], [], [], [], []]

# print(len(csv)) # 150
for i in range(len(csv)):
    # 데이터를 5개의 리스트에 각각 입력한다.
    csvk[i % K].append(csv[i])
```

훈련	테스트
BCDE	A
ACDE	B
ABDE	C
ABCE	D
ABCD	E

```
# 리스트를 훈련 전용 데이터와 테스트 전용 데이터로 분할하는 함수
def split_data_label(rows):
    data = []; label = []
    for row in rows:
        data.append(row[0:4])
        label.append(row[4])
    return (data, label)

# 정답률 구하기 --- (※3)
def calc_score(test, train):
    test_f, test_l = split_data_label(test)
    train_f, train_l = split_data_label(train)
    # 학습시키고 정답률 구하기
    clf = svm.SVC()
    clf.fit(train_f, train_l)
    pre = clf.predict(test_f)
    return metrics.accuracy_score(test_l, pre)
```

교차 검증 사용해보기

- 붓꽃 데이터를 사용하여 교차 검증하는 프로그램을 작성하세요.
- K의 값은 5로 설정하도록 한다.
- 파일 이름 : cross_iris_01.py

```
# K개로 분할해서 정답률 구하기 --- (※4)
score_list = []

for testc in csvk : # 5번 반복
    # print (testc)
    # testc 이외의 데이터를 훈련 전용 데이터로 사용하기
    trainc = []
    for i in csvk:
        if i != testc: trainc += i

    sc = calc_score(testc, trainc)

    score_list.append(sc)

print("각각의 정답률 =", score_list)

print("평균 정답률 =", sum(score_list) / len(score_list))
```

```
출력 결과
각각의 정답률 = [1.0, 0.9666666666666667,
0.9666666666666667, 0.9666666666666667,
0.9333333333333335]

평균 정답률 = 0.966666666667
```

교차 검증 사용해보기

- scikit-learn에서 제공해주는 교차 검증기를 사용해보도록 한다.
- 파일 이름 : cross_iris_02.py

```
import pandas as pd
from sklearn import svm, model_selection

# 붓꽃의 CSV 데이터 읽어 들이기 --- (※1)
csv = pd.read_csv('iris.csv')

# 리스트를 훈련 전용 데이터와 테스트 전용 데이터로 분할하기 --- (※2)
data = csv[["SepalLength","SepalWidth","PetalLength","PetalWidth"]]
label = csv["Name"]

# 크로스 밸리데이션하기 --- (※3)
clf = svm.SVC() # clf를 학습 평가기 객체라고 한다.

# cross_val_score 함수는 크로스 밸리데이션을 수행해주는 함수이다.
# cross_val_score(학습 평가기, 학습용 데이터, 정답 레이블, cv=분할수)
scores = model_selection.cross_val_score(clf, data, label, cv=5)

print("각각의 정답률 =", scores)

print("평균 정답률 =", scores.mean())
```

```
출력 결과
각각의 정답률 = [ 0.96666667   1.
0.96666667   0.96666667   1.          ]

평균 정답률 = 0.98
```