파이썬 딥러닝

이성주

seongjoo@codebasic.io

jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [10]:
```
from sklearn.preprocessing import StandardScaler
```

$\frac{X-\mu}{\sigma} \rightarrow X'$

In [11]:
```
scaler = StandardScaler()
```

1) fit → σ, μ

In [12]:
```
X_std = scaler.fit_transform(X)
```

2) transform → $\frac{X-\mu}{\sigma}$

In [15]:
```
DataFrame(X_std, columns=X.columns)[:5]
```

Out[15]:

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoi
0	1.518613	-0.562250	0.232053	-1.169593	1.913905	0.808997	1.0348
1	0.246290	-0.499413	-0.827996	-2.490847	0.018145	0.568648	0.7336
2	0.196879	0.021231	1.109334	-0.268738	0.088358	0.808997	1.2155
3	1.691550	-0.346811	0.487926	-0.809251	0.930918	2.491446	1.4665
4	0.295700	0.227694	1.840403	0.451946	1.281985	0.808997	0.6633

jupyter Day 3

Logout

train_test_split(X , Xstd , y , ··· , test_size=0.3)

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [11]:
```
scaler = StandardScaler()
```

In [12]:
```
X_std = scaler.fit_transform(X)
```

In [15]:
```
DataFrame(X_std, columns=X.columns)[:5]
```

X ··· Xstd y

In [16]:
```
X_train, X_test, X_std_train, X_std_test, y_train, y_test = train_te
    X, X_std, y, test_size=0.3
)
```

In [ ]:

jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [21]:
```
test_score1 = model.score(X_test, y_test)
train_score1 = model.score(X_train, y_train)
```

In [22]:
```
test_score1, train_score1
```

Out[22]: (0.9629629629629291, 0.9758064516129325)

In [23]:
```
model.fit(X_std_train, y_train)
```

In [24]:
```
test_score2 = model.score(X_std_test, y_test)
train_score2 = model.score(X_std_train, y_train)
```

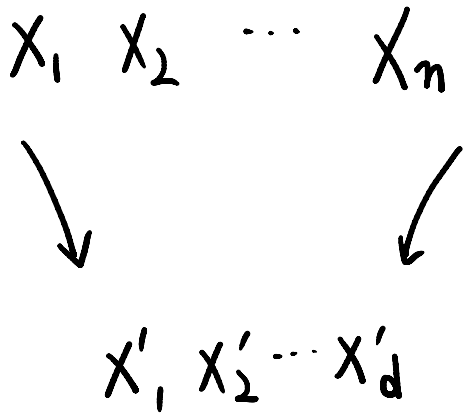
In [25]:
```
test_score2, train_score2
```

Out[25]: (0.9629629629629291, 1.0)

In [ ]:

차원축소

$$z = \sum wx = w_1 x_1 + w_2 x_2 + \cdots$$



jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

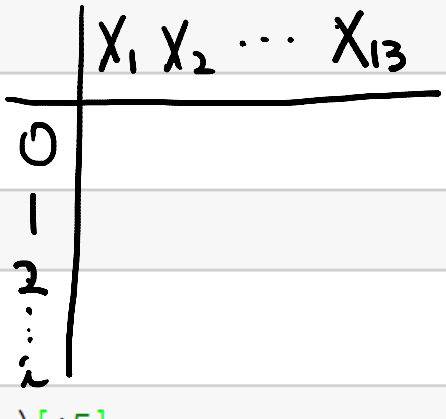
In [26]:
```
from sklearn.decomposition import PCA
```

In [27]:
```
pca = PCA(n_components=2)
```
비지도학습

In [28]:
```
X_pca = pca.fit_transform(X)
```
PCA.fit_transform

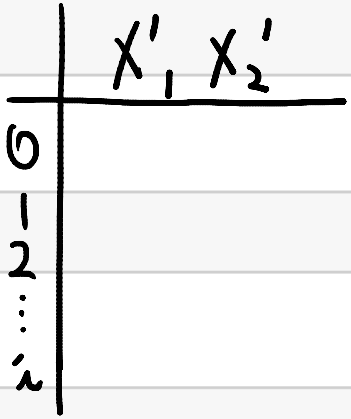
In [29]:
```
X.shape
```

Out[29]: (178, 13)



In [30]:
```
X_pca.shape
```

Out[30]: (178, 2)



In [31]:
```
DataFrame(X_pca)[:5]
```

Out[31]:

	0	1
0	318.562979	21.492131

jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [42]:
```
model.fit(X_train, y_train)
```
...

In [43]:
```
model.score(X_test, y_test)
```
Out[43]: 0.9629629629629291

In [44]:
```
model.fit(X_pca_tr, y_train)
```
...

In [45]:
```
model.score(X_pca_te, y_test)
```
Out[45]: 1.0

In [ ]:

특징의 개수가 줄었는데 성능이 향상되었다?

jupyter Day 3

Logout

Trusted | Python 3

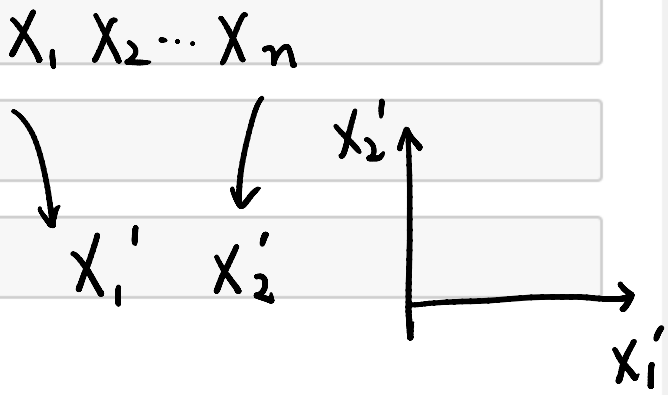
File Edit View Insert Cell Kernel Widgets Help

Code

In [26]:
```
from sklearn.decomposition import PCA
```

In [27]:
```
pca = PCA(n_components=2)
```

In [37]:
```
X_pca = pca.fit_transform(X_std)
```



In [38]:
```
X.shape
```

Out[38]: (178, 13)

In [39]:
```
X_pca.shape
```

Out[39]: (178, 2)

In [40]:
```
DataFrame(X_pca)[:5]
```

Out[40]:

	0	1
0	3.316751	-1.443463

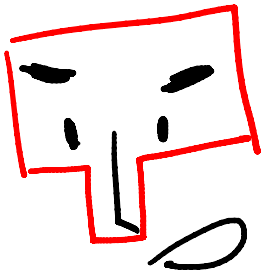
PCA 이전

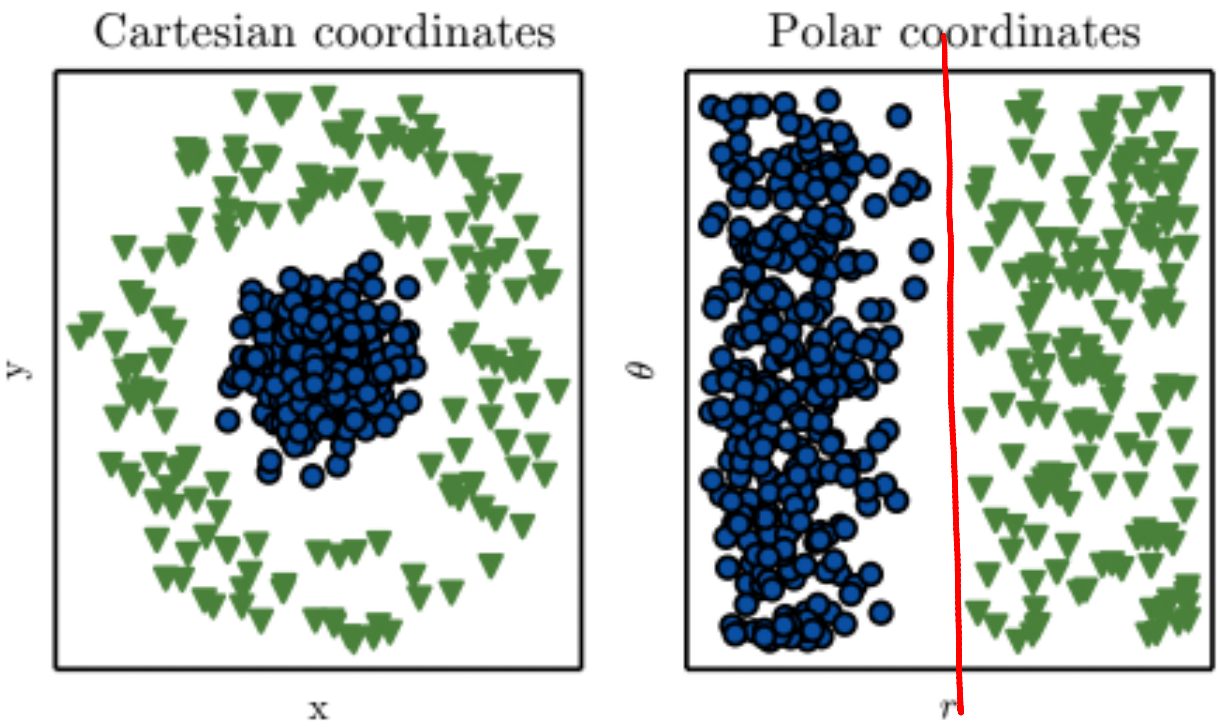
$$Z = \sum wx = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

PCA 이후 (n_components=2)

$$Z' = \sum wx = w_1' x_1' + w_2' x_2'$$

표현





jupyter Day 3

Logout

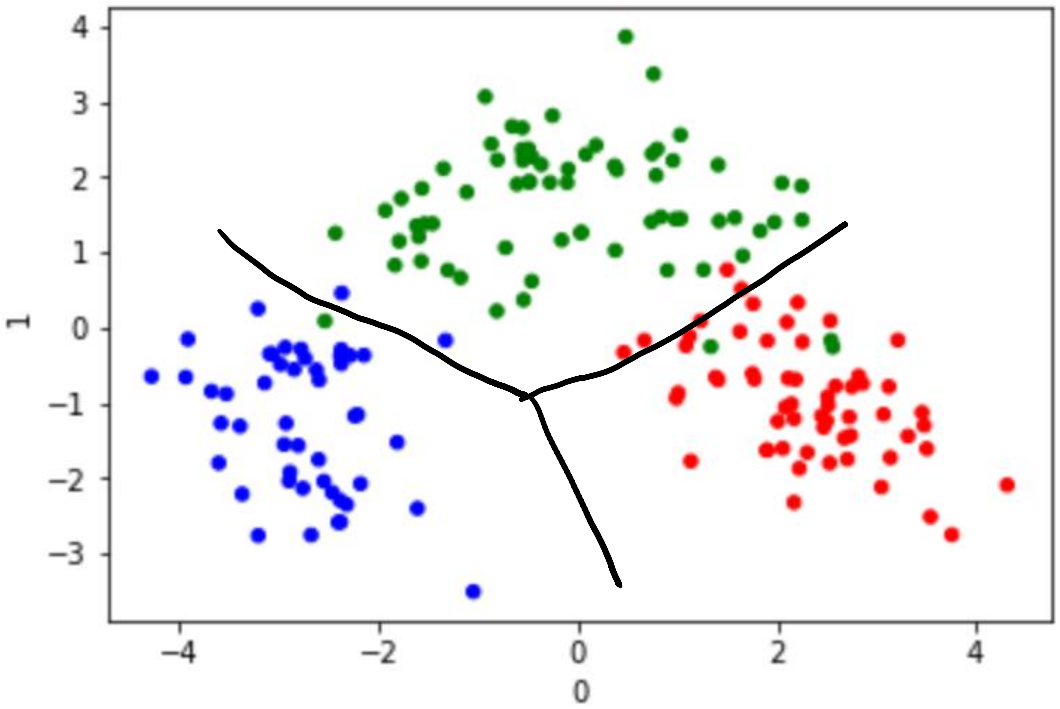
Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [53]:
```
DataFrame(X_pca).plot(kind='scatter', x=0, y=1, c=colormap)
```

Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0xc59ee80>



In [ ]:

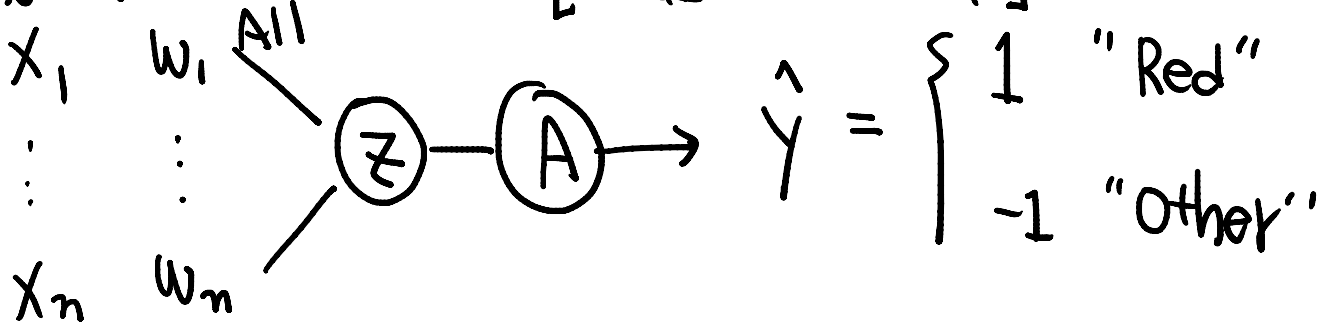
다중분류 (OvR)

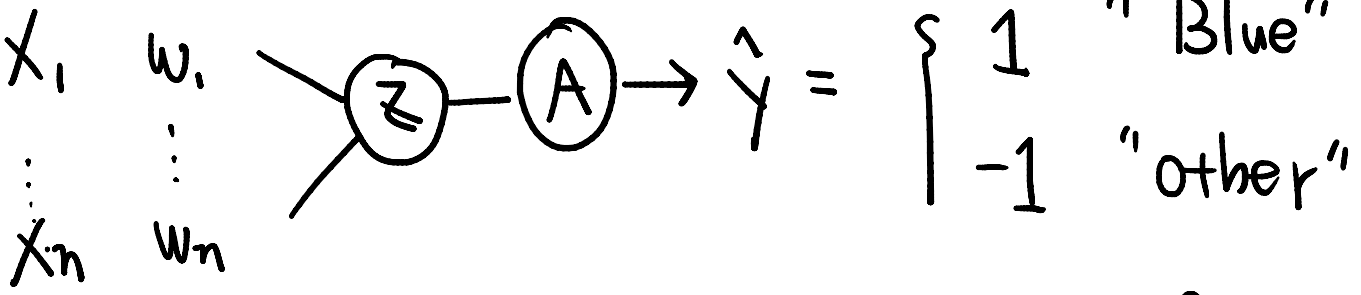
OvA

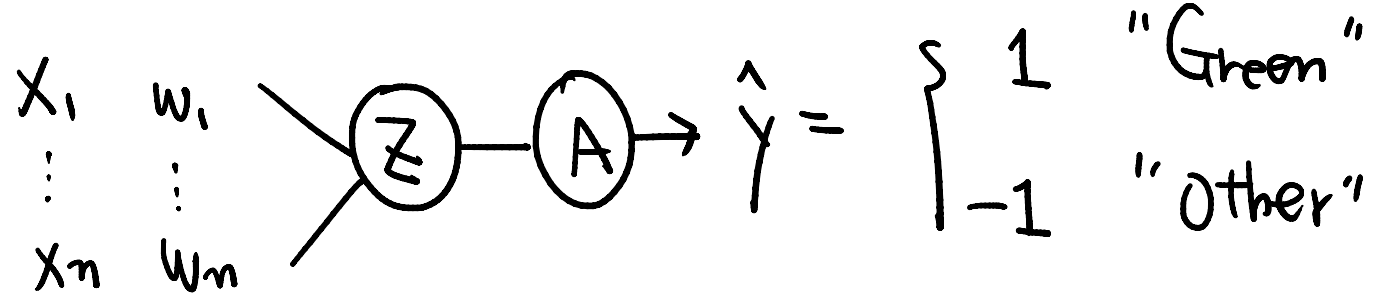
One-Vs.-Rest

$3 \times n$

$$\begin{bmatrix} w_{R1} & \cdots & w_{RN} \\ w_{B1} & \cdots & w_{BN} \\ w_{G1} & \cdots & w_{GN} \end{bmatrix}$$







jupyter Day 3

Logout

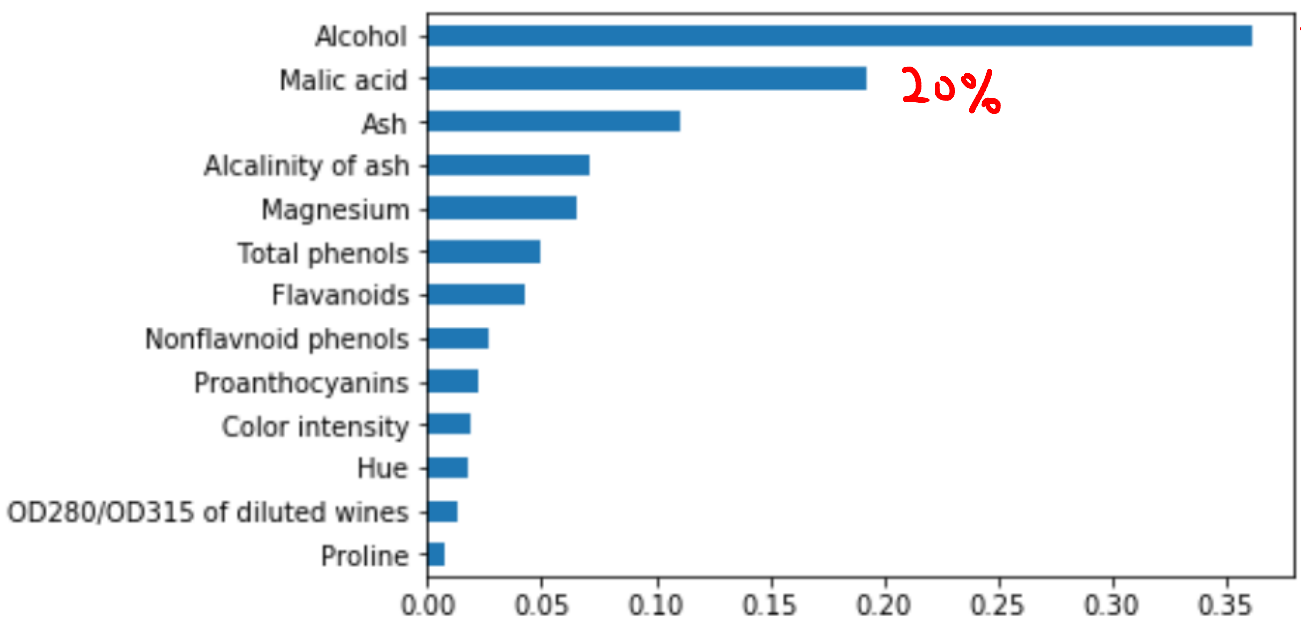
Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [63]:
```
기여도.sort_values().plot(kind='barh')
```

Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0xd473b00>



In [ ]:

jupyter Day 3

Logout

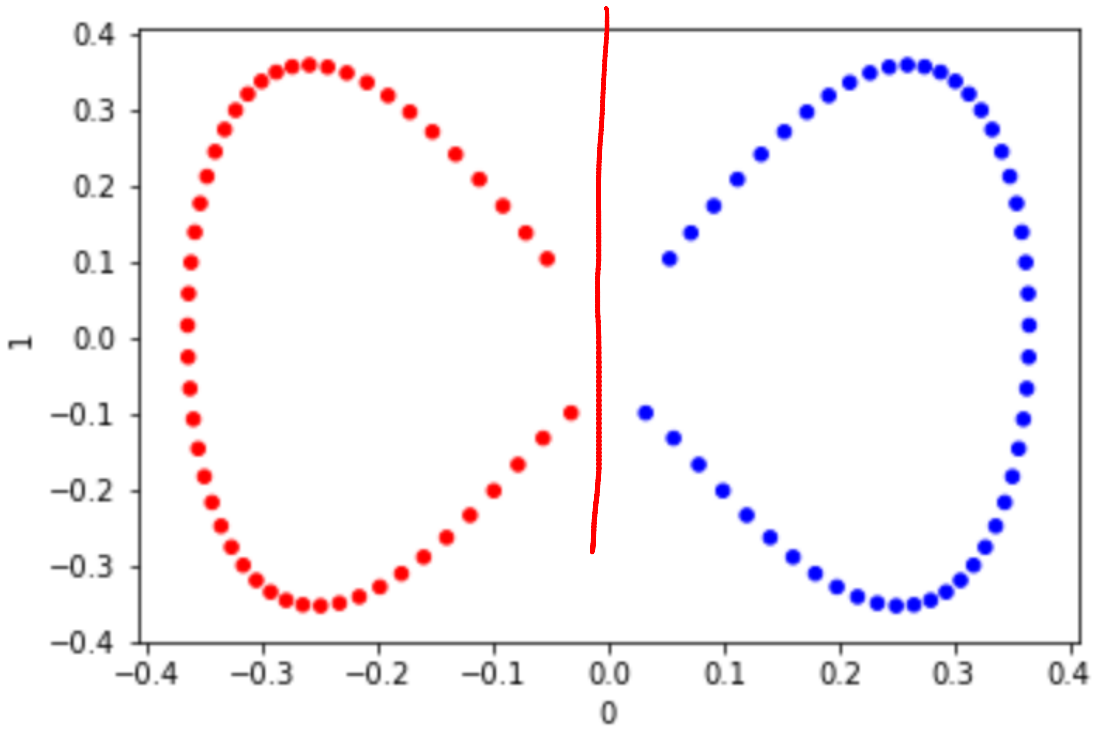
Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [76]:
```
DataFrame(X_kpca).plot(kind='scatter', x=0, y=1, c=colormap)
```

Out[76]: <matplotlib.axes._subplots.AxesSubplot at 0xd6af4a8>



In [ ]:

jupyter Day 3

Logout

Trusted | Python 3

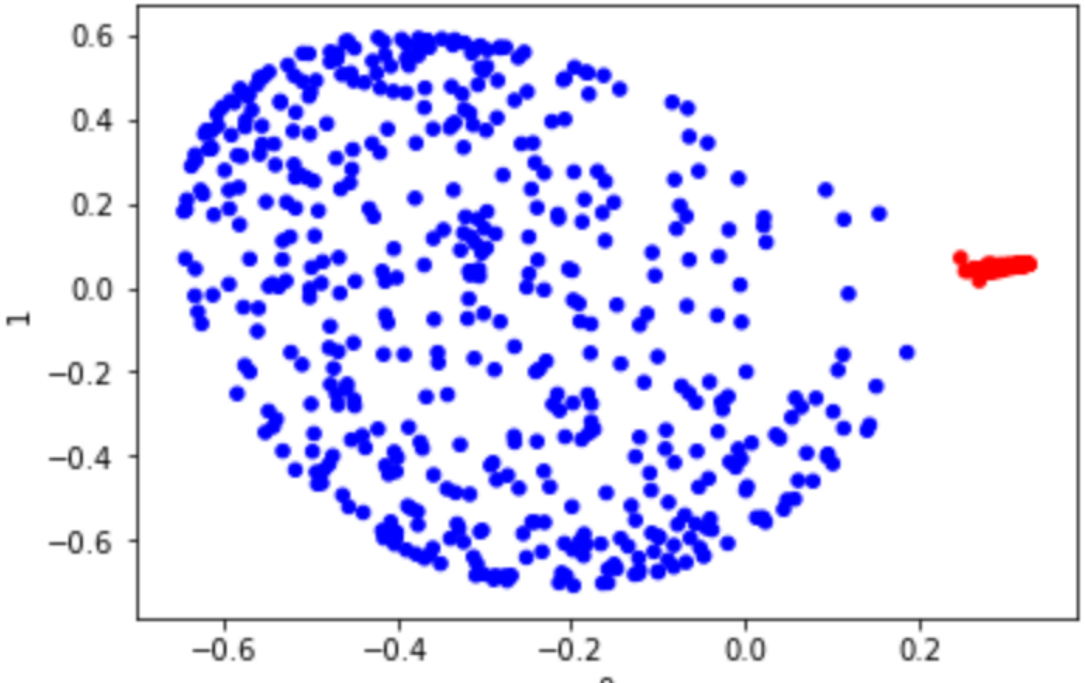
File Edit View Insert Cell Kernel Widgets Help

Code

In [82]:
```
X_kpca = kpca.fit_transform(X)
```

In [83]:
```
DataFrame(X_kpca).plot(kind='scatter', x=0, y=1, c=colormap)
```

Out[83]: <matplotlib.axes._subplots.AxesSubplot at 0xe9d4240>



jupyter Day 3

LR 모델을 훈련/평가

1) PCA가 도움이 되나?

2) n-components의 값은?

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

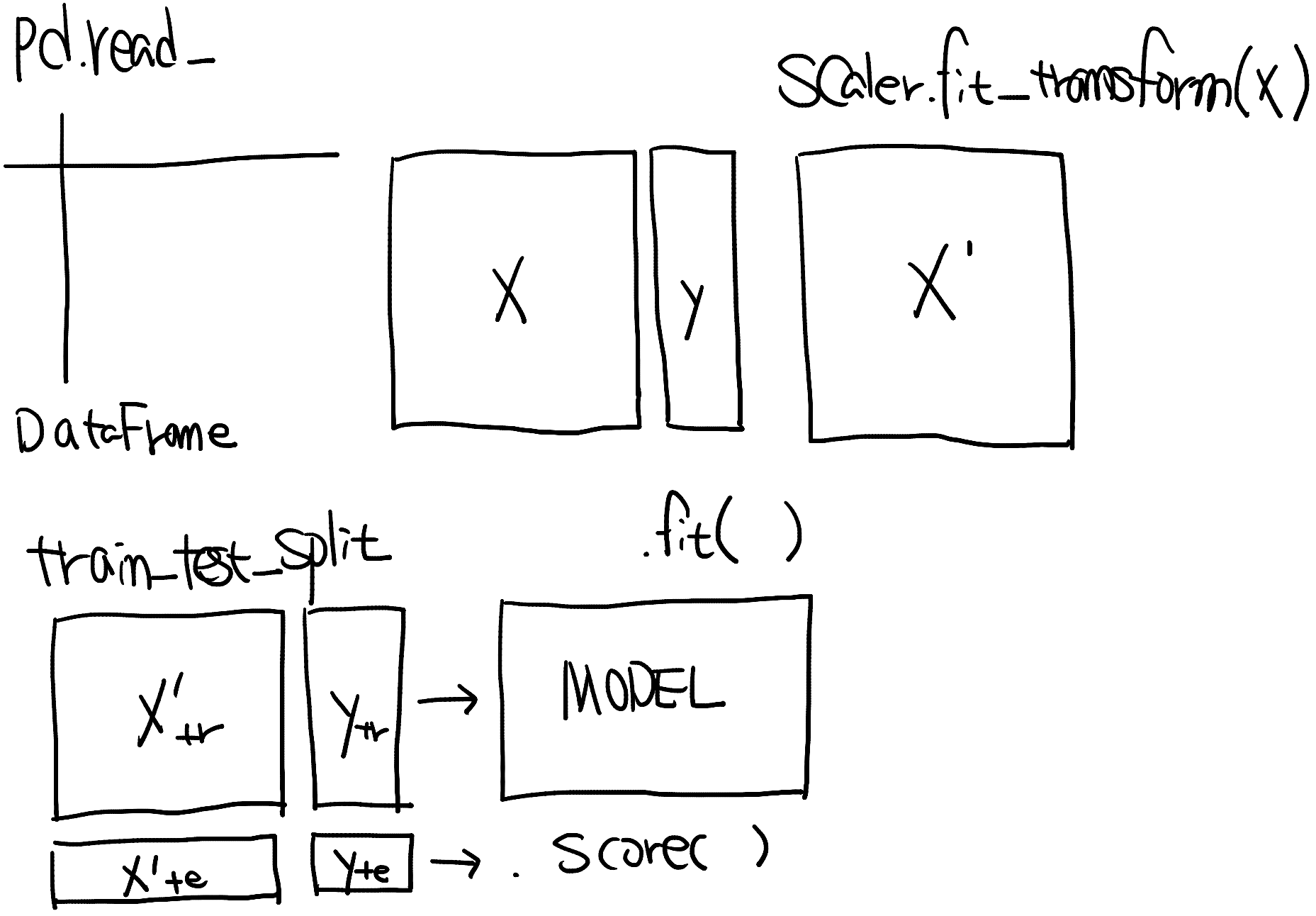
In [85]: `pd.read_csv('data/wdbc.data', header=None)`

Out[85]:

환자ID → 0, y → 1

	0	1	2	3	4	5	6	7	8
0	842302	M	17.990	10.38	122.80	1001.0	0.11840	0.27760	0.300100
1	842517	M	20.570	17.77	132.90	1326.0	0.08474	0.07864	0.086900
2	84300903	M	19.690	21.25	130.00	1203.0	0.10960	0.15990	0.197400
3	84348301	M	11.420	20.38	77.58	386.1	0.14250	0.28390	0.241400
4	84358402	M	20.290	14.34	135.10	1297.0	0.10030	0.13280	0.198000
5	843786	M	12.450	15.70	82.57	477.1	0.12780	0.17000	0.157800
6	844359	M	18.250	19.98	119.60	1040.0	0.09463	0.10900	0.112700
7	84458202	M	13.710	20.83	90.20	577.9	0.11890	0.16450	0.093660
8	844981	M	13.000	21.82	87.50	519.8	0.12730	0.19320	0.185900
9	84501001	M	12.460	24.04	83.97	475.9	0.11860	0.23960	0.227300
10	845636	M	16.020	23.24	102.70	797.8	0.08206	0.06669	0.032990

y = { M "악성", B "양성"


Pd.read_
DataFrame
X
y
Scaler.fit_transform(X)
X'
train_test_split
.fit( )
X'tr
ytr
MODEL
X'te
yte
. score( )

```
    results, index=[entry[0] for entry in X_sets])
report.columns = ['test', 'train']
```

In [112]:
```
report
```

Out[112]:

	test	train
raw	0.935673	0.967337
scaled	0.982456	0.994975
pca	0.947368	0.957286

In [ ]:

jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

```
    results.append((test_score, train_score))
```

In [120]:
```
report = DataFrame(
    results, index=[entry[0] for entry in X_sets])
report.columns = ['test', 'train']
```

In [121]:
```
report
```

Out[121]:

	test	train
raw	0.953216	0.959799
✓ scaled	0.953216	0.987437
pca2	0.959064	0.954774
pca3	0.970760	0.944724

$X_1\ X_2 \cdots X_{30}$

$X_1'\ X_2'$

$X_1'\ X_2'\ X_3'$

In [ ]:

jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [118]:
```
X_tr, X_te, X_std_tr, X_std_te, ₩
X_pca2_tr, X_pca2_te, X_pca3_tr, X_pca3_te,₩
y_train, y_test = train_test_split(
    X, X_std, X_pca2, X_pca3, y, test_size=0.3)
```

In [96]:
```
model = LogisticRegression(C=1.0)
```

In [119]:
```
X_sets = [
    ('raw', X_tr, X_te),
    ('scaled', X_std_tr, X_std_te),
    ('pca2', X_pca2_tr, X_pca2_te),
    ('pca3', X_pca3_tr, X_pca3_te)
]

results = []
for setname, X_train, X_test in X_sets:
    model.fit(X_train, y_train)
    test_score = model.score(X_test, y_test)
    train_score = model.score(X_train, y_train)
    results.append((test_score, train_score))
```

jupyter Day 3

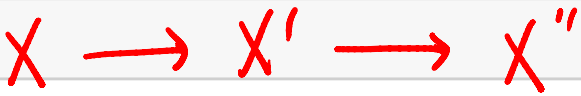
Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [90]:
```
X.shape
```



Out[90]: (569, 30)

In [91]:
```
scaler = StandardScaler()
```

In [92]:
```
X_std = scaler.fit_transform(X)
```

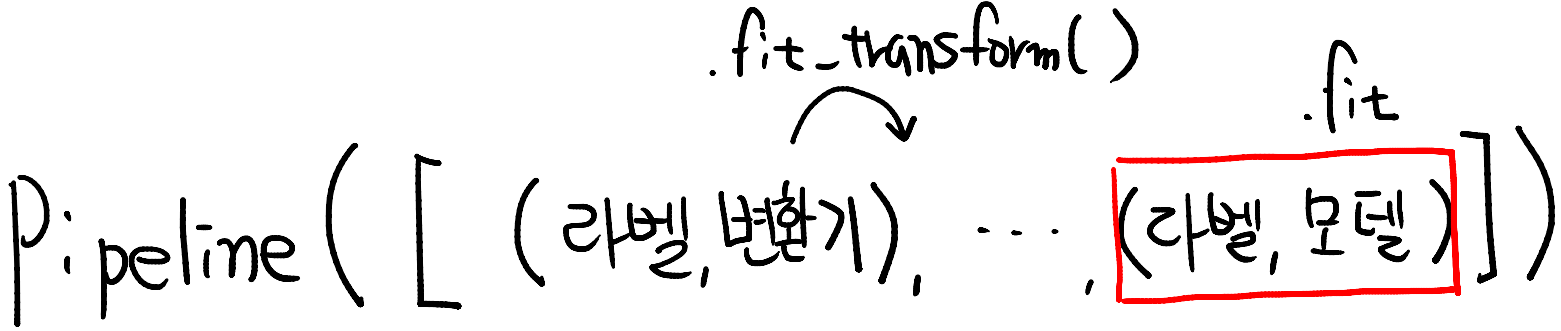
In [93]:
```
pca2 = PCA(n_components=2)
```

In [117]:
```
X_pca2 = pca2.fit_transform(X_std)
```

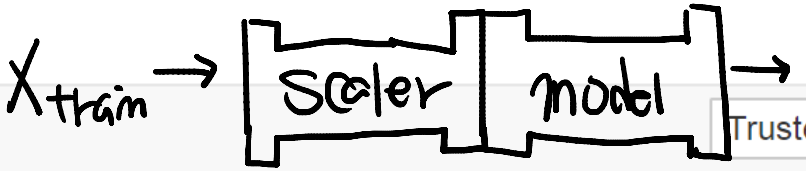
In [113]:
```
pca3 = PCA(n_components=3)
```

In [114]:
```
X_pca3 = pca3.fit_transform(X_std)
```

In [118]:
```
X_tr, X_te, X_std_tr, X_std_te, ₩
X_pca2_tr, X_pca2_te, X_pca3_tr, X_pca3_te,₩
```


.fit_transform( )
.fit
Pipeline([(라벨, 변환기), ... , (라벨, 모델)])

jupyter Day 3



Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

```
    ('model', LogisticRegression(C=1.))
])
```

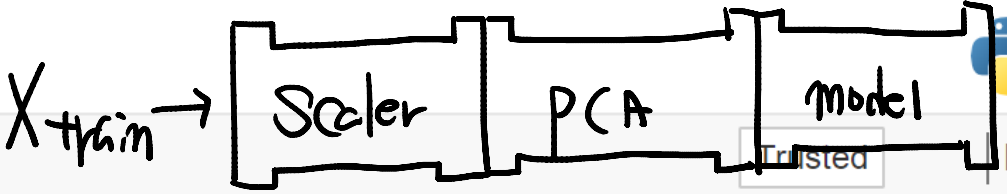
In [127]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
```

In [128]:
```
pipe.fit(X_train, y_train)
```

Out[128]:
```
Pipeline(steps=[('scaler', StandardScaler(copy=True, with_mean=Tru
e, with_std=True)), ('model', LogisticRegression(C=1.0, class_weigh
t=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_j
obs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=
0.0001,
          verbose=0, warm_start=False))])
```

In [ ]:

jupyter Day 3



Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

...

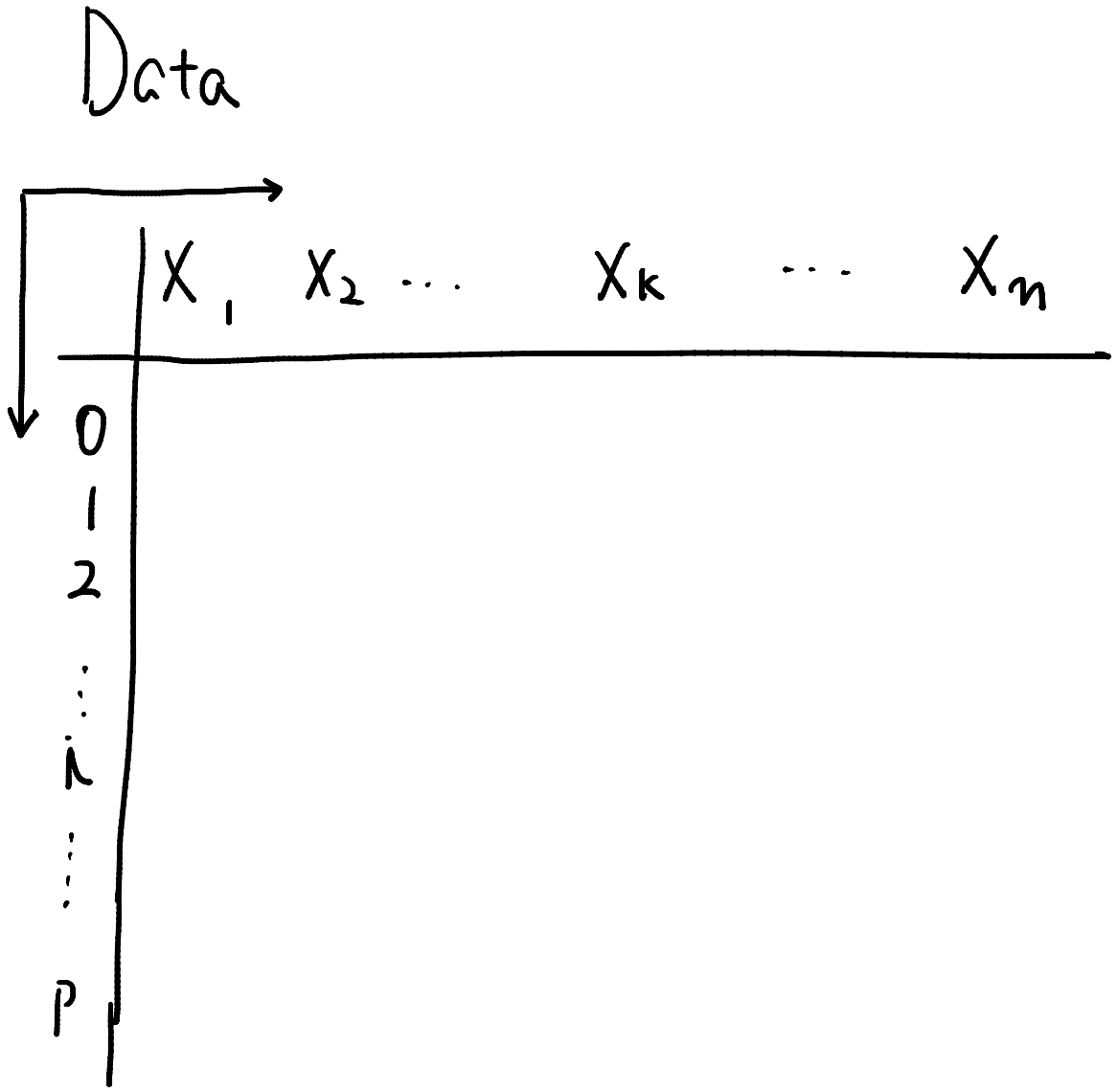
In [129]:
```
pipe.score(X_test, y_test)
```

Out[129]:
```
0.97660818713450293
```

In [130]:
```
pipe_pca2.fit(X_train, y_train)
```

Out[130]:
```
Pipeline(steps=[('scaler', StandardScaler(copy=True, with_mean=Tru
e, with_std=True)), ('pca', PCA(copy=True, iterated_power='auto', n
_components=2, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)), ('model', LogisticReg
ression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_j
obs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=
0.0001,
          verbose=0, warm_start=False))])
```

In [ ]:


Data
X_1 X_2 ... X_k ... X_n
0
1
2
i
p

jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

샘플 개수를 10, 20, ... 100%로 변경하면서 모델 평가

In [140]:
```
train_sizes, train_scores, val_scores = learning_curve(
    estimator=pipe,
    X=X_train, y=y_train,
    train_sizes=np.linspace(0.1, 1., 10),
    cv=3
)
```

교차확인 →

S E 구간개수

In [141]:
```
np.linspace(0.1, 1., 10)
```

Out[141]:
```
array([ 0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ])
```

10% 20% 100%

In [ ]:

jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [142]:
```
DataFrame(train_scores, index=train_sizes)
```

Out[142]:

	0	1	2
26	1.000000	1.000000	1.000000
52	1.000000	1.000000	1.000000
79	1.000000	1.000000	1.000000
105	1.000000	0.971429	0.971429
132	0.984848	0.977273	0.977273
158	0.987342	0.974684	0.981013
184	0.989130	0.978261	0.983696
211	0.995261	0.981043	0.985782
237	0.983122	0.978903	0.987342
264	0.988636	0.984848	0.984848

model.score(Xtrain, ytrain)

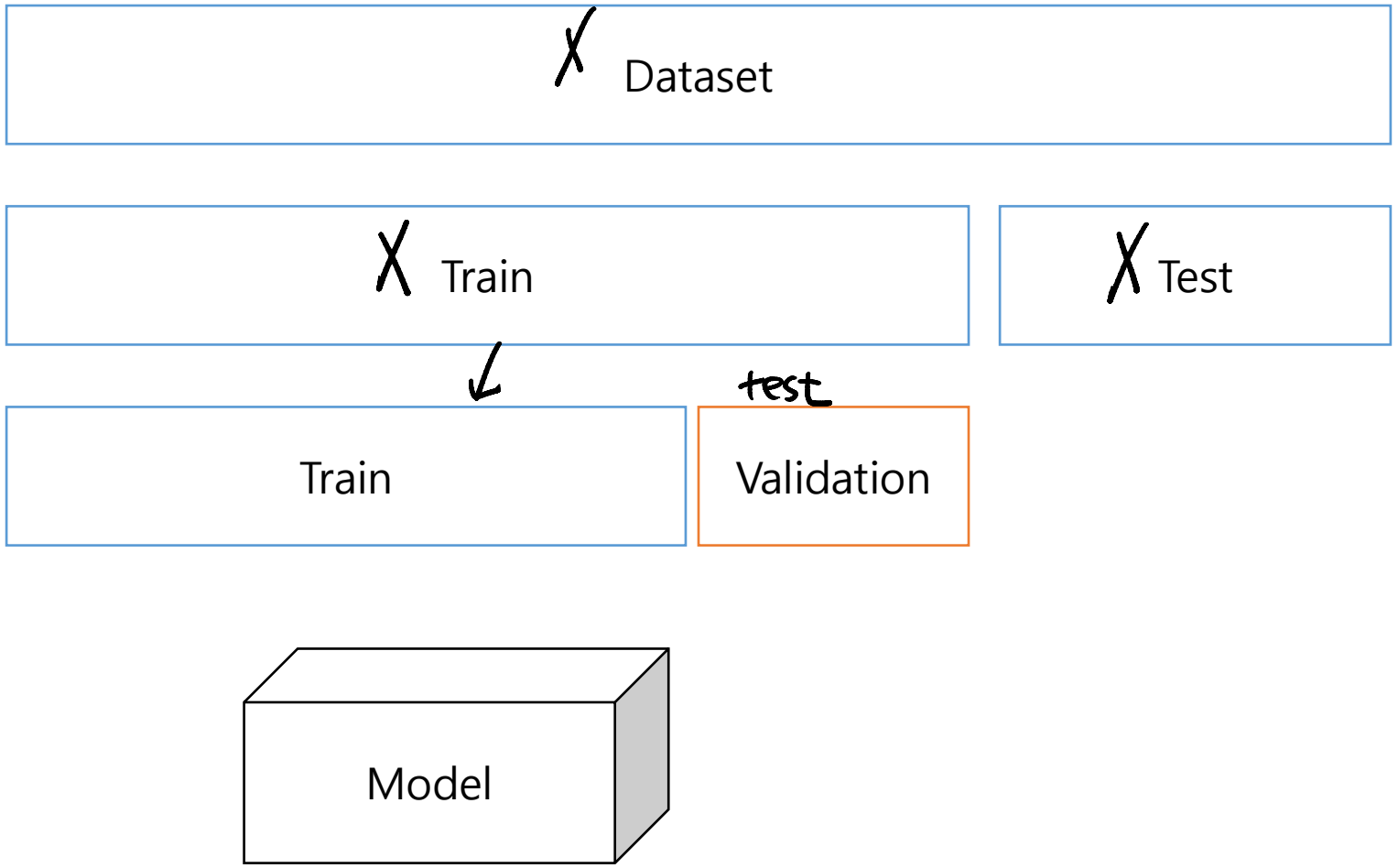
샘플

10%

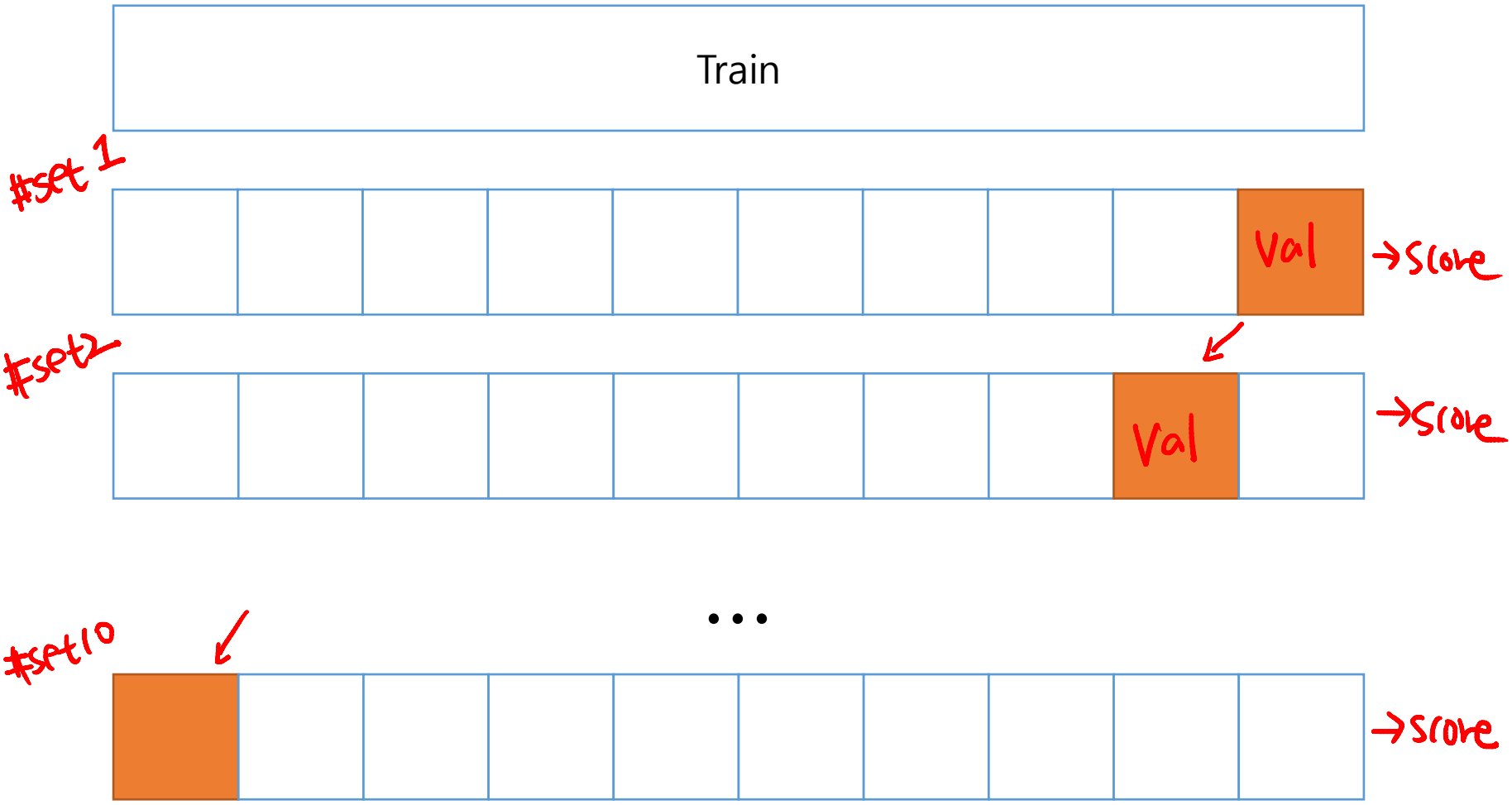
20%

100%

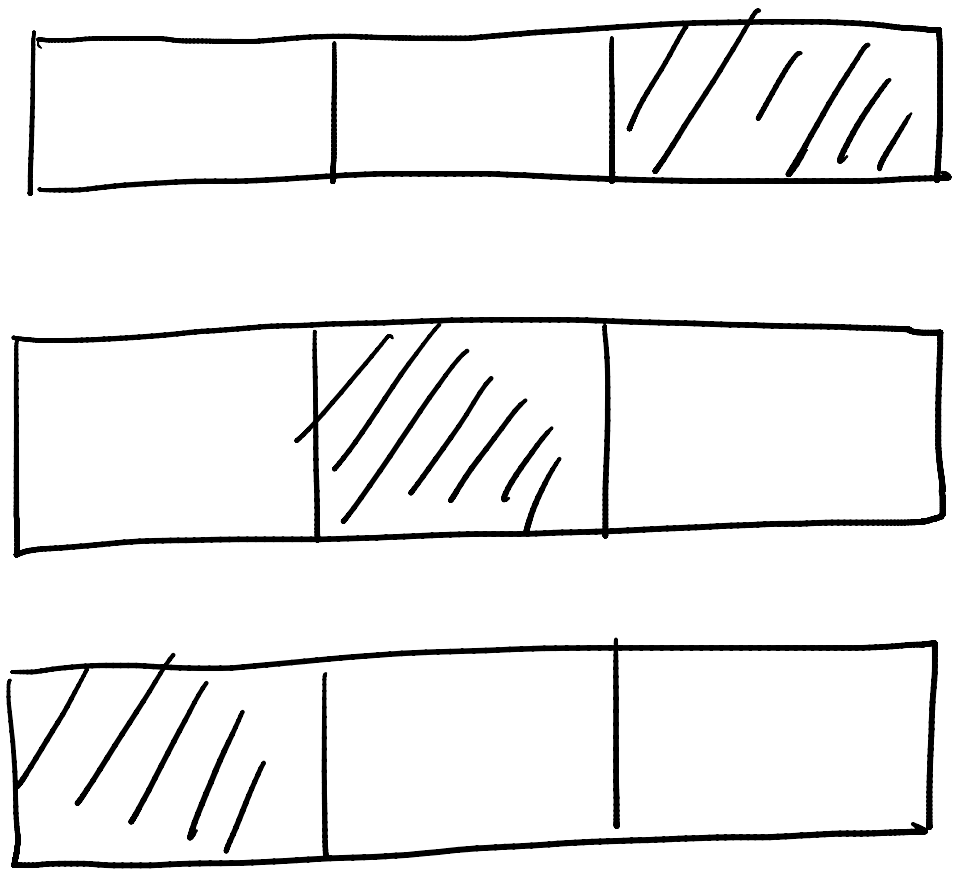
Holdout



k-fold 교차 확인



CV=3



jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [152]:
```
train_results.mean(1)
```

Out[152]:

1

0

평균

	0	1	2
26	1.000000	1.000000	1.000000
52	1.000000	1.000000	1.000000
79	1.000000	1.000000	1.000000
105	1.000000	0.971429	0.971429
132	0.984848	0.977273	0.977273
158	0.987342	0.974684	0.981013
184	0.989130	0.978261	0.983696
211	0.995261	0.981043	0.985782
237	0.983122	0.978903	0.987342
264	0.988636	0.984848	0.984848

jupyter Day 3

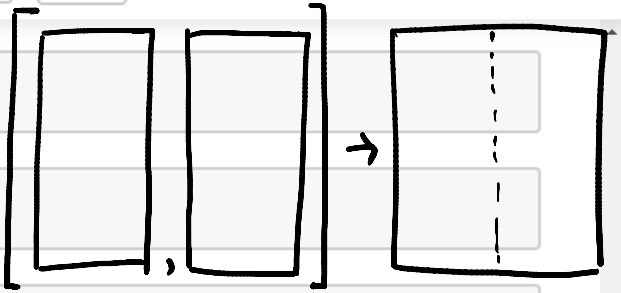
Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

pd.concat



In [154]:
```
train_mean = train_results.mean(1)
```

In [156]:
```
val_mean = val_results.mean(1)
```

In [158]:
```
learning_results = pd.concat([train_mean, val_mean], axis=1)
learning_results.columns = ['train', 'val']
```

In [159]:
```
learning_results
```

Out[159]:

	train	val
26	1.000000	0.949683
52	1.000000	0.942145
79	1.000000	0.949683
105	0.980952	0.967360
132	0.979798	0.967398
158	0.981013	0.969923

jupyter Day 3

Logout

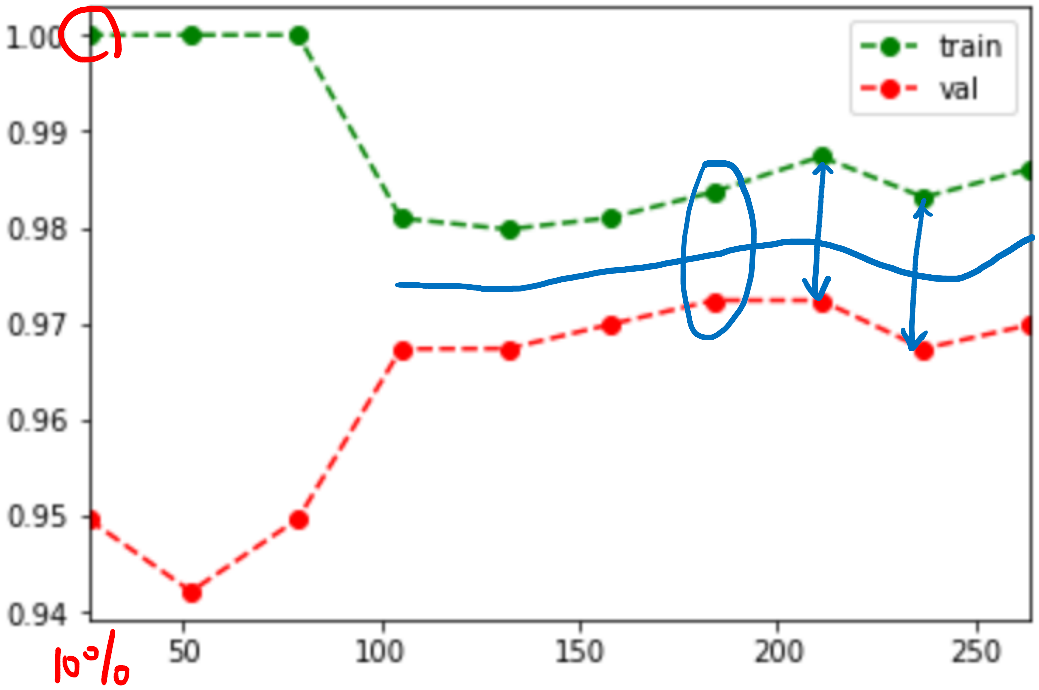
Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [160]:
```
learning_results.plot(style=['go--', 'ro--'])
```

Out[160]: <matplotlib.axes._subplots.AxesSubplot at 0xebccac8>



In [ ]:

jupyter Day 3

Logout

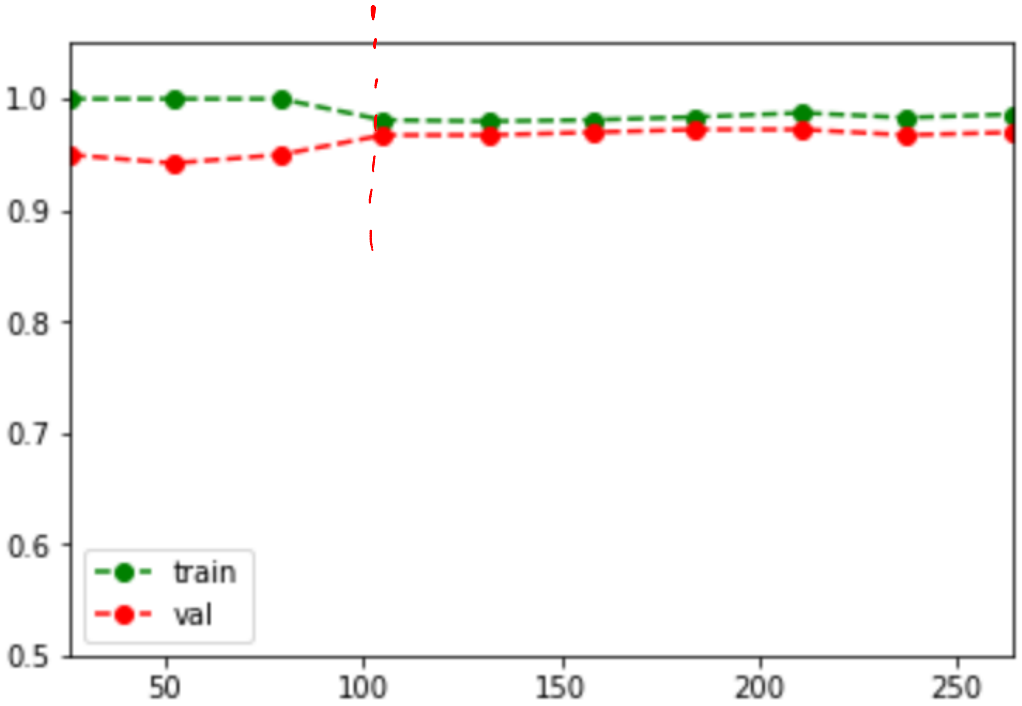
Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [161]:
```
learning_results.plot(style=['go--', 'ro--'], ylim=(0.5, 1.05))
```

Out[161]: <matplotlib.axes._subplots.AxesSubplot at 0xc4e0940>



In [ ]:

jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

하이퍼 파라메터에 따른 성능 평가

In [162]:
```
from sklearn.model_selection import validation_curve
```

In [163]:
```
params = [0.001, 0.01, 0.1, 1., 10., 100., 1000.]
```

In [164]:
```
train_scores, val_scores = validation_curve(
    estimator=pipe,
    X=X_train, y=y_train,
    param_name='model__C',
    param_range=params,
    cv=10
)
```

라벨__인자

'model__C'

pipeline([...

('model', LogisticRegression(C))

In [ ]:

jupyter Day 3

Logout

Trusted | Python 3

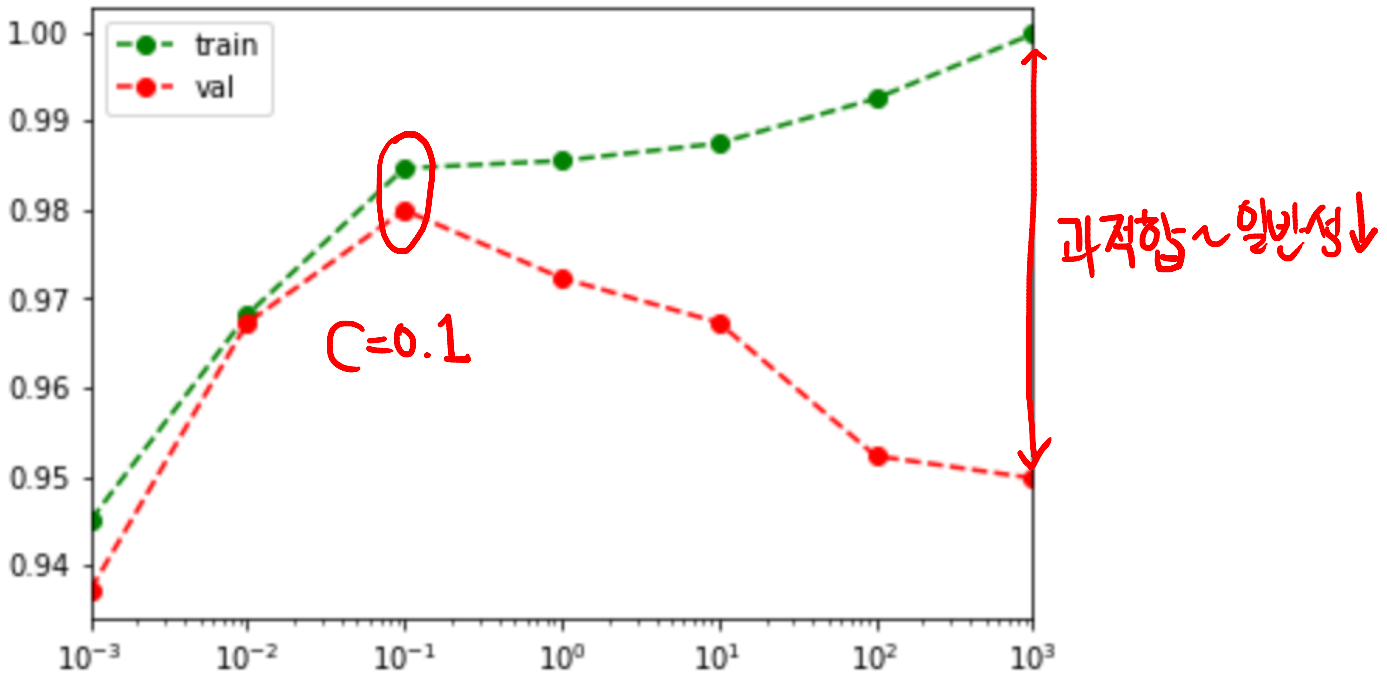
File Edit View Insert Cell Kernel Widgets Help

Code

In [172]:
```
param_results.columns = ['train', 'val']
```

In [174]:
```
param_results.plot(style=['go--', 'ro--'], logx=True)
```

Out[174]: <matplotlib.axes._subplots.AxesSubplot at 0xecd1668>



연습

다음의 모델의 C값의 최적값은 얼마인지 살펴봅니다.

In [175]:
```
from sklearn.svm import SVC
```

In [176]:
```
SVC(kernel='linear')
```

Out[176]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma='auto', kernel='lin
ear',
  max_iter=-1, probability=False, random_state=None, shrinking=Tru
e,
  tol=0.001, verbose=False)
```

In [ ]:
```

```

jupyter Day 3

Logout

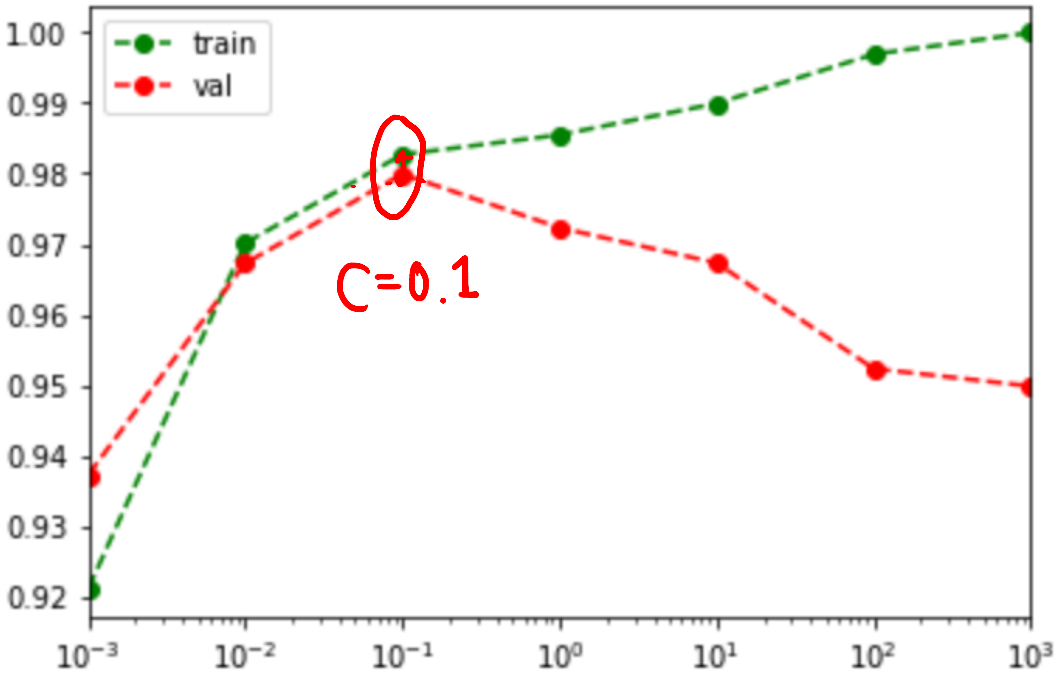
Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

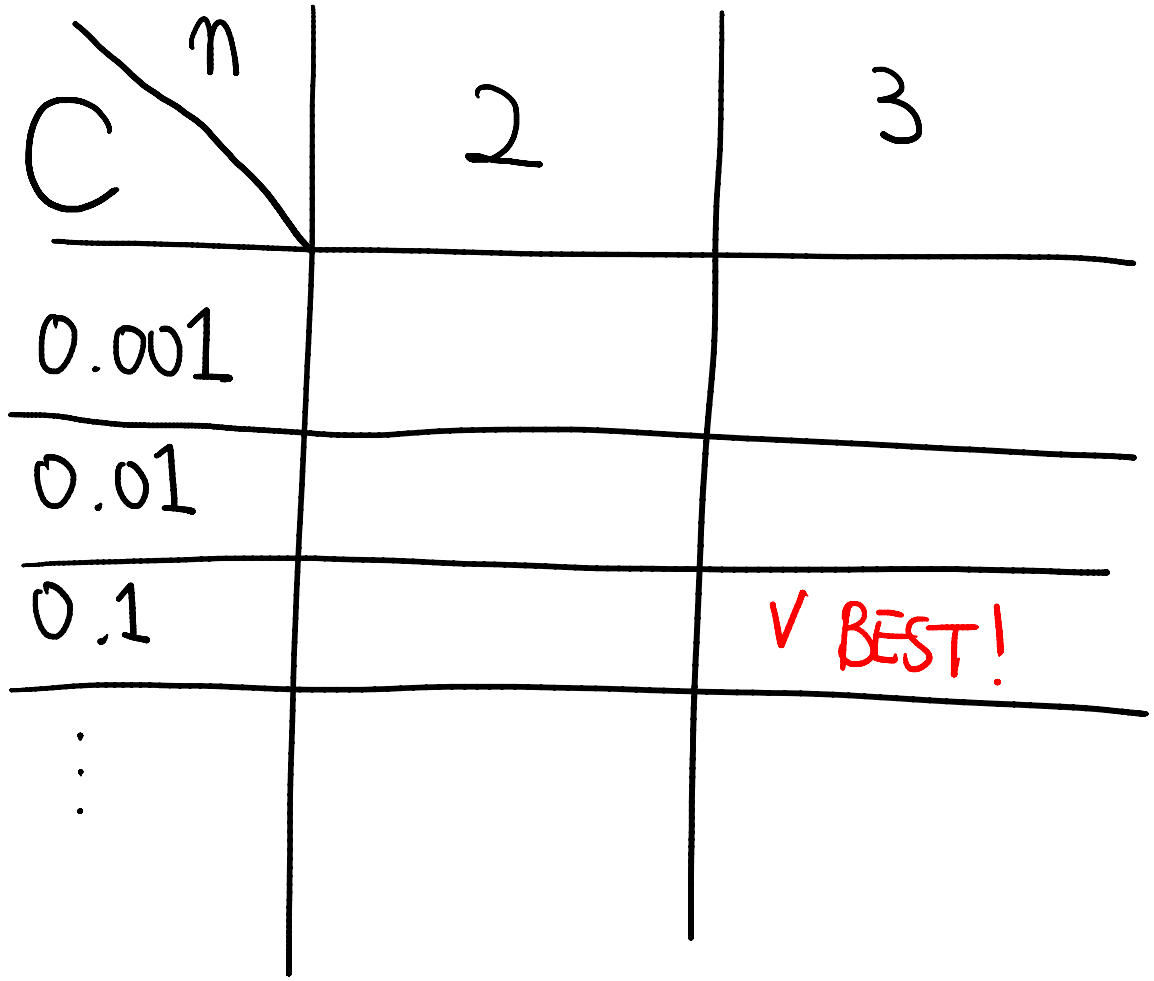
In [193]:
```
param_results.plot(style=['go--', 'ro--'], logx=True)
```

Out[193]: <matplotlib.axes._subplots.AxesSubplot at 0xce9d240>



In [ ]:

그리드 탐색 (Grid Search)



jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

OvR

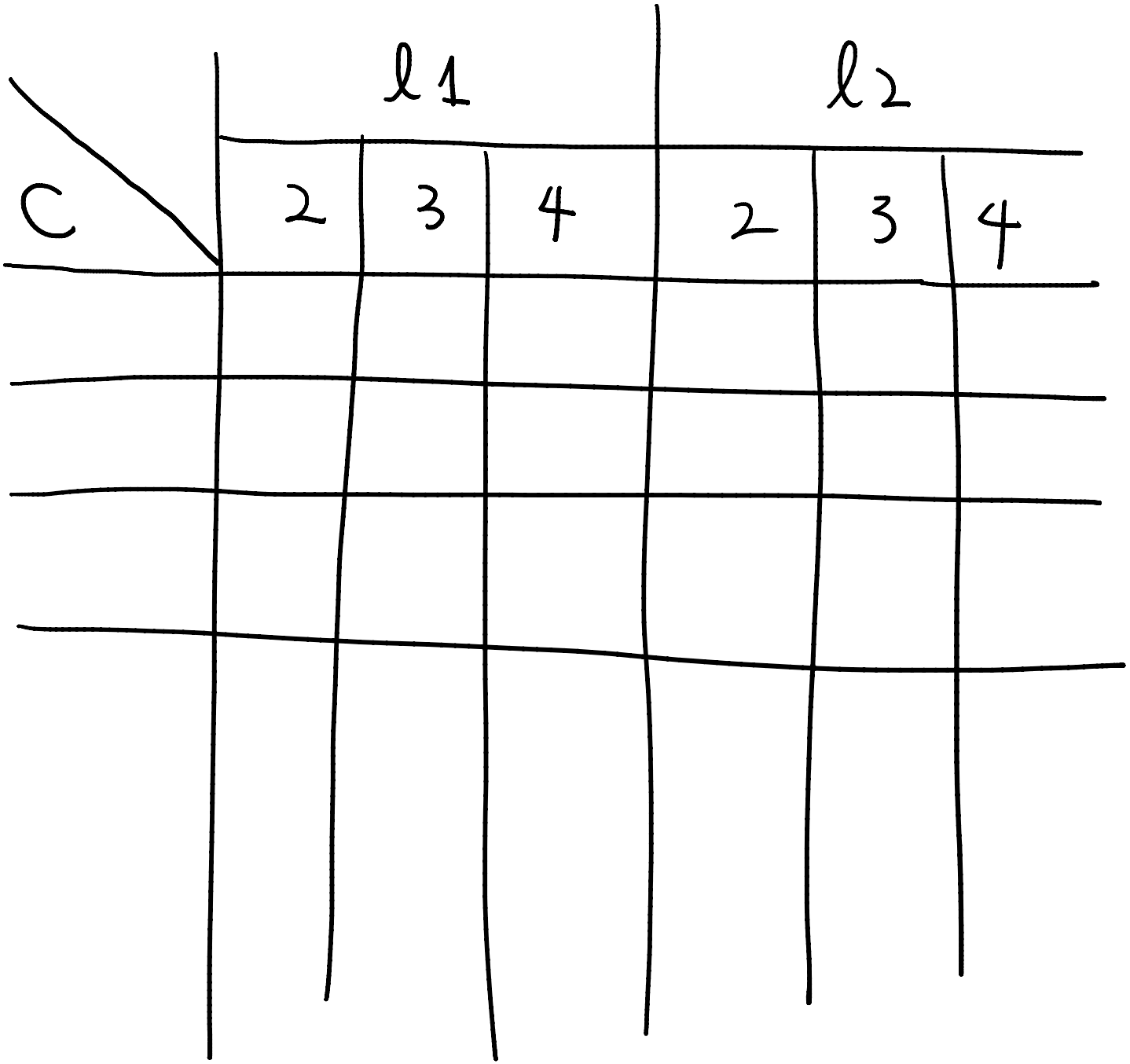
In [202]:
```
LogisticRegression()
```

Out[202]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_interc
ept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_j
obs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=
0.0001,
          verbose=0, warm_start=False)
```

l1 l2

In [198]:
```
param_grid = [
1)  {'model__C': params},
2)  {'pca__n_components': [2,3,4], 'model__C': params},
3)  {
        'pca__n_components': [2,3,4],
        'model__C': params,
        'model__penalty': ['l1', 'l2']}
]
```

In [ ]:


l1
l2
C
2
3
4
2
3
4

jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [206]:
```
gs = GridSearchCV(
    estimator=pipe,
    param_grid=param_grid,
    scoring='accuracy',
    cv=10
)
```

In [207]:
```
gs.fit(X_train, y_train)
```

...

In [208]:
```
gs.best_score_
```

Out[208]: 0.96733668341708545

In [210]:
```
gs.best_params_
```

Out[210]: {'model__C': 0.1, 'pca__n_components': 4}

In [211]:
```
best_model = gs.best_estimator_
```

Out[208]: 0.96733668341708545

In [210]:
```
gs.best_params_
```

Out[210]: {'model__C': 0.1, 'pca__n_components': 4}

In [211]:
```
best_model = gs.best_estimator_
```

In [212]:
```
best_model.score(X_test, y_test)
```

Out[212]: 0.9707602339181286

In [ ]:

jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [229]:
```
confmat = confusion_matrix(y_true=y_test, y_pred=y_pred)
```

In [235]:
```
confmat = DataFrame(confmat, index=['P', 'N'], columns=['P', 'N'])
confmat.index.name = '실제'
confmat.columns.name = '예측'
confmat
```

Out[235]:

실제 \ 예측	P	N
P	103	4
N	1	63

P P P N 4 5

N N N P 1 0

5 5

Acc. 96% 97%

In [ ]:

$$PRE = \frac{TP}{TP + FP} \downarrow$$

예측 / 실제	P	N
P	103 (TP)	4 (FN)
N	1 (FP)	63 (TN)

$$REC = TPR \downarrow$$

$$F1 = 2\frac{PRE \times REC}{PRE + REC} \downarrow$$

$$TPR = \frac{TP}{TP + FN} = \frac{103}{103 + 4}$$

"True Pos. Rate"

$$FPR = \frac{FP}{FP + TN} = \frac{1}{1 + 63}$$

jupyter Day 3

Logout

Notebook saved | Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

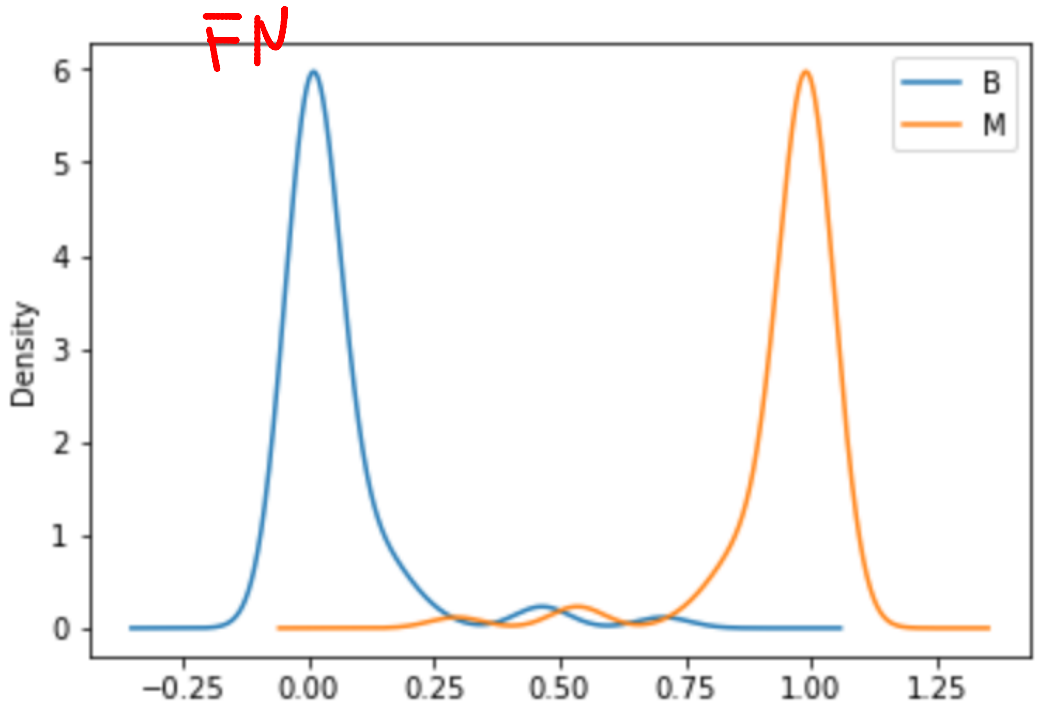
Code

In [253]:
```
probas.loc['M'].plot(kind='kde')
```

Out[253]: <matplotlib.axes._subplots.AxesSubplot at 0x11914208>

M: '악성'

FN



In [ ]:

jupyter Day 3

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

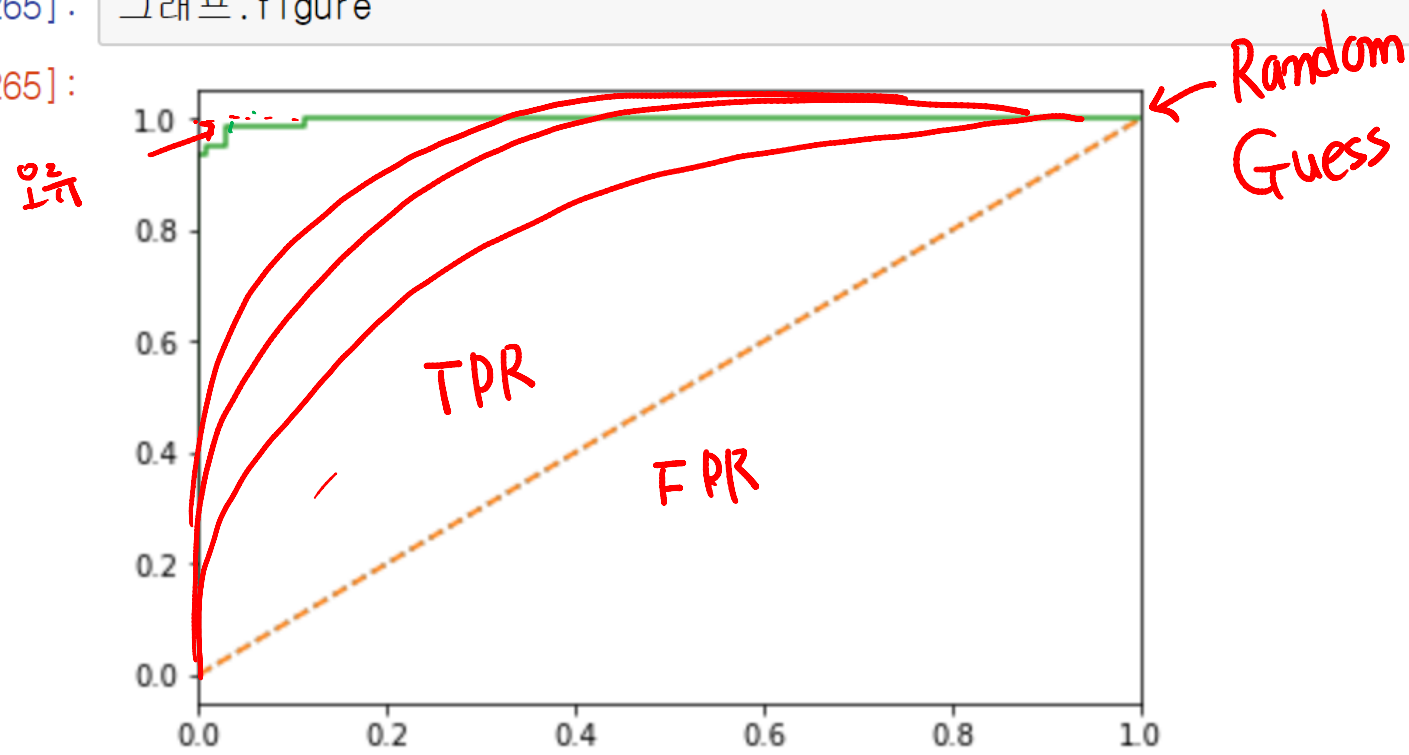
Code

In [264]:
```
Series(tpr, index=fpr).plot(ax=그래프)
```

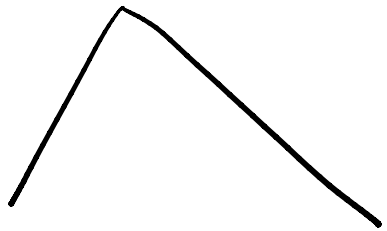
Out[264]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1167b1d0>
```

In [265]:
```
그래프.figure
```

Out[265]:



지도학습



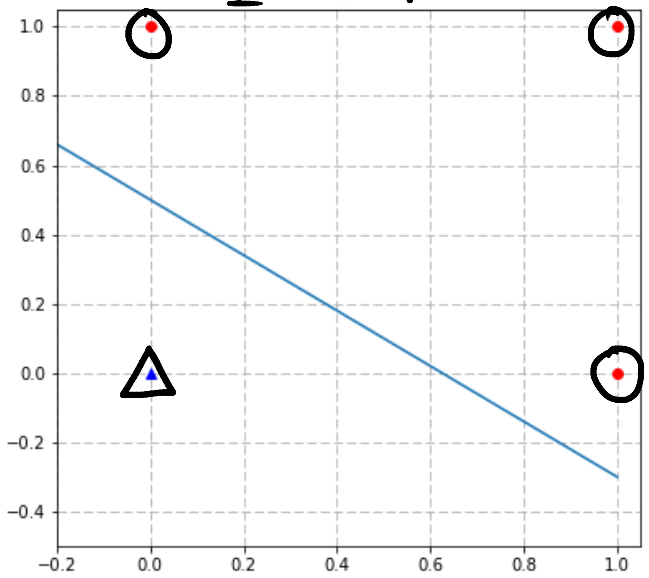
회귀(추세)　　분류

$X_n \rightarrow X_{n+1}$

1969 선형 모델의 한계

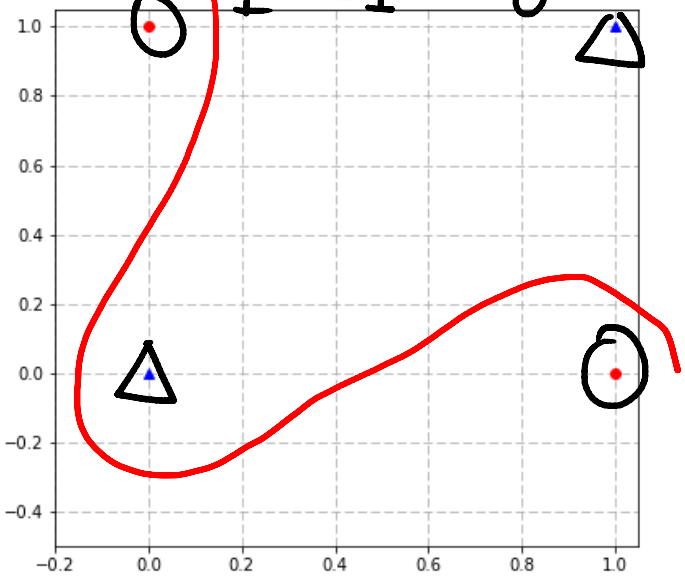
OR

0	0	0
0	1	1
1	0	1
1	1	1



XOR

0	0	0
0	1	1
1	0	1
1	1	0



다층 퍼셉트론

		NAND	OR	XOR
x1	x2	s1	s2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0