


머신러닝 모델 실습

<https://scikit-learn.org/stable/tutorial/index.html>

[Install](#) [User Guide](#) [API](#) [Examples](#) [Community](#) [More ▾](#)

[Prev](#) [Up](#) [Next](#)

scikit-learn 1.3.0
[Other versions](#)

Please [cite us](#) if you use the software.

Welcome to scikit-learn
scikit-learn Tutorials
An introduction to machine learning with scikit-learn
A tutorial on statistical-learning for scientific data processing
Working With Text Data
Choosing the right estimator
External Resources, Videos and Talks
Getting Started
[User Guide](#)
[Glossary of Common Terms and API Elements](#)
[Examples](#)
[API Reference](#)
[Developer's Guide](#)

scikit-learn Tutorials

An introduction to machine learning with scikit-learn

- Machine learning: the problem setting
- Loading an example dataset
- Learning and predicting
- Conventions

A tutorial on statistical-learning for scientific data processing

- Statistical learning: the setting and the estimator object in scikit-learn
- Supervised learning: predicting an output variable from high-dimensional observations
- Model selection: choosing estimators and their parameters
- Unsupervised learning: seeking representations of the data
- Putting it all together

학습(learn) 절차

- 데이터를 준비한다
- 모델(model)을 선정한다
- 문제에 맞는 objective function을 선정한다.

sklearn 제공 데이터셋

- `from sklearn import datasets`
- `iris = datasets.load_iris()` # 분꽃 데이터셋
- `digits = datasets.load_digits()` # 숫자 데이터셋
- `diabetes = datasets.load_diabetes()` # 당뇨 데이터셋
- `bcancer = datasets.load_breast_cancer()` # 폐암
- `wine = datasets.load_wine()` # 와인

분꽃(iris) 분류문제

iris setosa



petal

sepal

iris versicolor



petal

sepal

iris virginica



petal

sepal

- 3가지 종류의 분꽃을 특징 데이터를 통해서 구분

train_test_split

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(  
iris['data'], iris['target'], test_size = 0.25, random_state=0)
```

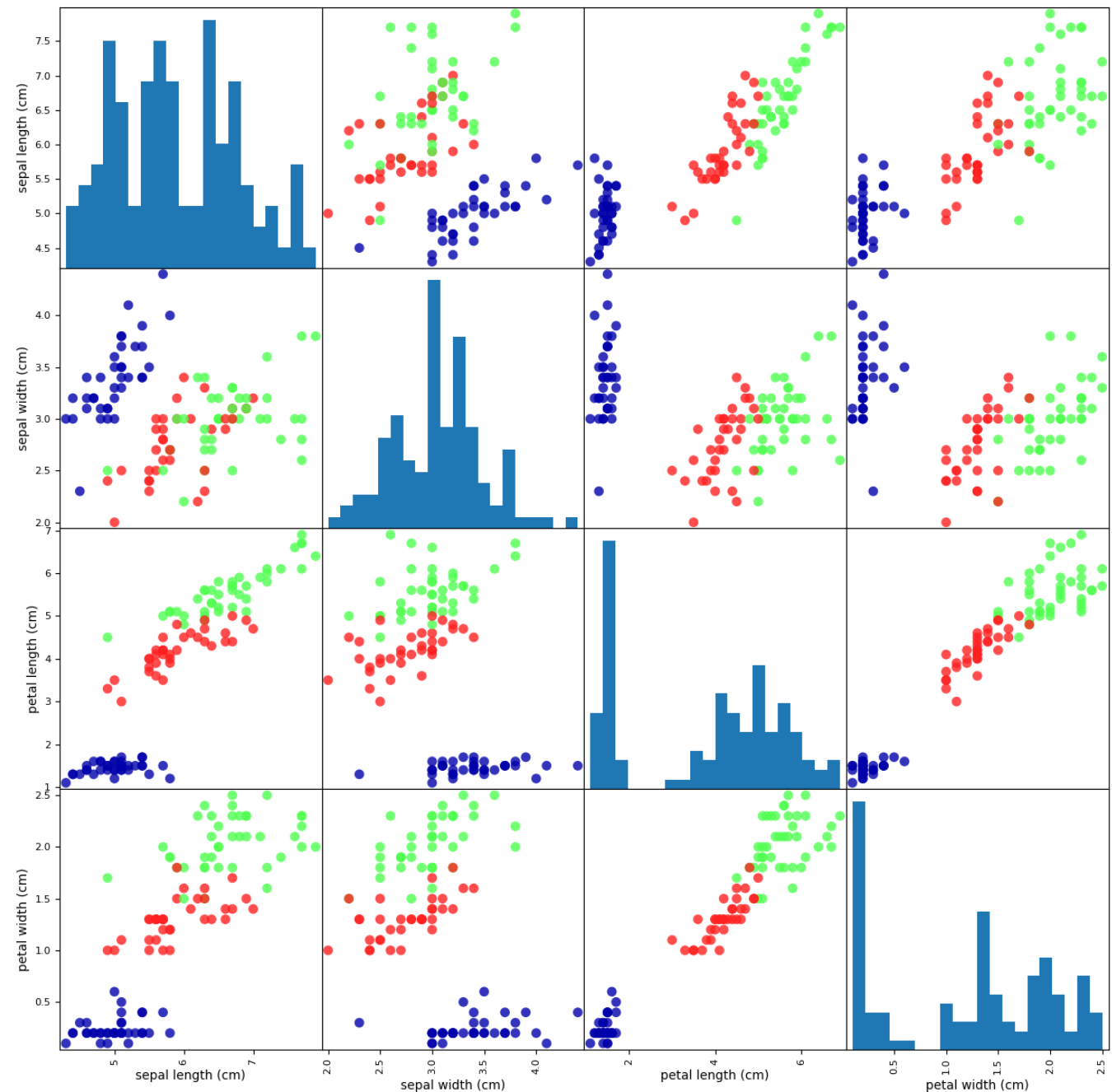
- 데이터셋을 학습(train) 데이터셋과 테스트(test) 데이터셋으로 나누어 줌
- 일반적으로 train 75%, test 25% 비율로 split
- test_size : test의 비율을 변경 (e.g. 0.3)
- random_state를 지정하면, 동일한 결과를 리턴

[Q] train, validation, test로 나눌려면?

scatter_matrix

```
pandas.plotting.scatter_matrix(iris, c=y_train,  
figsize=(15, 15), marker='o', hist_kws={'bins': 20}, s=60, alpha=.8,  
cmap=cm3)
```

- 데이터셋의 특징들을 각각 선택하여, 2차원으로 가시화
- 데이터가 간단히 어떻게 되어 있는지 확인하기 좋음
- `c : color`



모델(model) 선정

- `model = LogisticRegression(C = 1.0, random_state=0)`
- `model = svm.SVC(C=1)`
- `model = DecisionTreeClassifier(criterion = 'entropy', max_depth = 5)`
- `model = RandomForestClassifier(n_estimators = 500, criterion = 'entropy', n_jobs = 2)`
- 이후에 `model.fit(X_train, y_train)`

svm.SVC (Support Vector Classifier)

```
svm.SVC(kernel='rbf', C = 10.0, gamma=0.1)
```

- **C**는 틀린 데이터에 대한 가중치로 **C=1**처럼 작으면 틀린 데이터에 대해서 영향을 거의 받지 않고, **C=1000**처럼 적용하면 틀린 데이터가 가능한 없도록 민감하게 분류함
- **gamma**는 가우시안 커널 폭의 역수로 하나의 훈련 샘플이 미치는 영향의 범위를 설정하는 파라미터로 모델의 복잡도를 결정 (**gamma**가 적을수록 모델의 복잡도를 낮음)

데이터 포인트 사이의 거리는 가우시안 커널에 의해 계산됩니다.

$$k_{rbf}(x_1, x_2) = \exp(-\gamma \|x_1 - x_2\|^2)$$

여기에서 x_1 과 x_2 는 데이터 포인트이며 $\|x_1 - x_2\|$ 는 유클리디안 거리이고 γ 감마, gamma는 가우시안 커널의 폭을 제어하는 매개변수입니다.

● 클래스 0
 ▲ 클래스 1
 ● 클래스 0 서포트 벡터
 ▲ 클래스 1 서포트 벡터

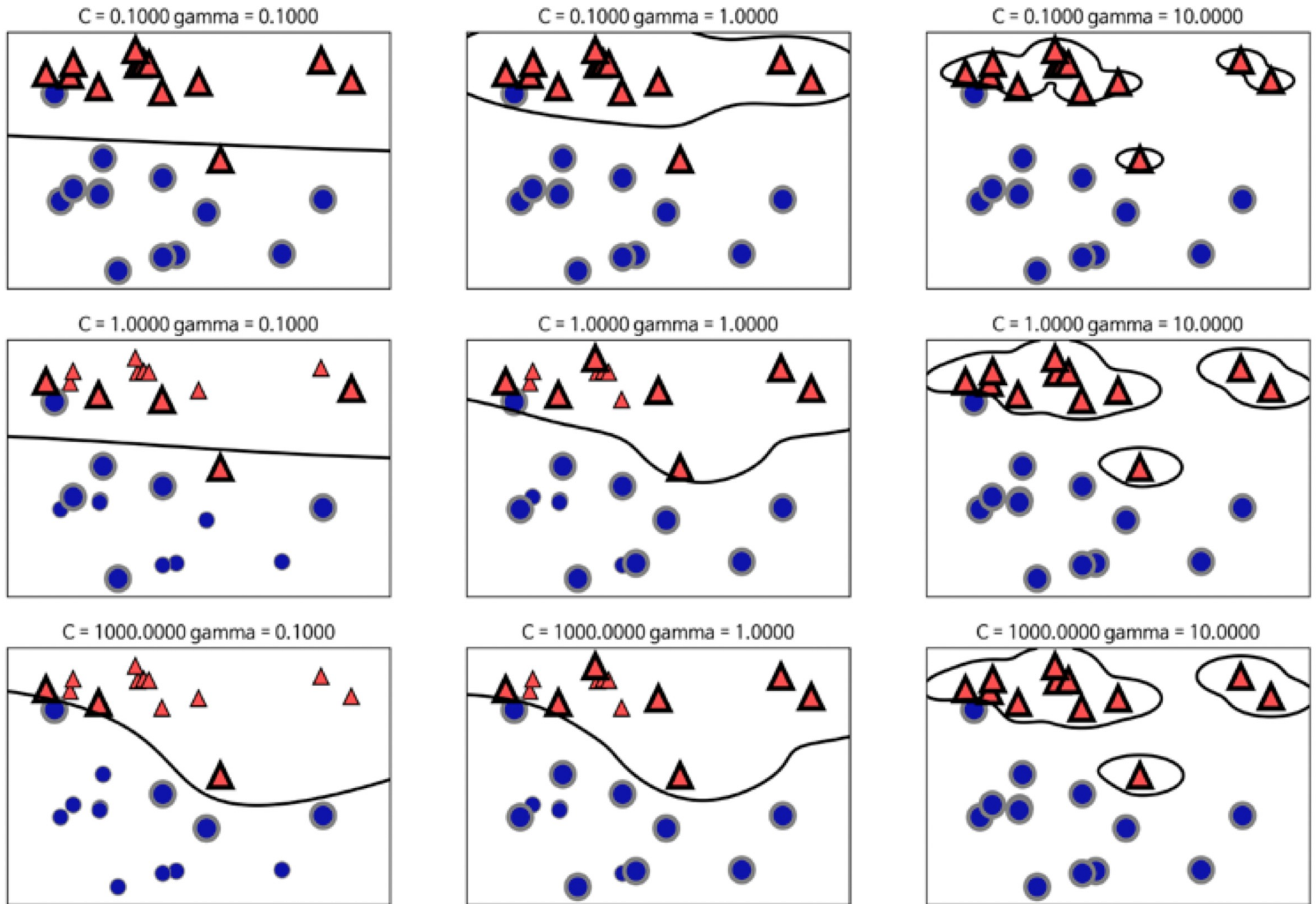


그림 2-42 C 와 γ 매개변수 설정에 따른 결정 경계와 서포트 벡터

```
from sklearn.metrics
import classification_report
```

- 테스트 결과를 confusion matrix 또는 그 결과 형태로 출력

```
from sklearn.metrics import classification_report
```

```
y_pred = model.predict(X_test)
print( metrics.confusion_matrix(y_test, y_pred) )
print( classification_report(y_test, y_pred) )
```

```
[>] [[13  0  0]
      [ 0 15  1]
      [ 0  0  9]]
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	1.00	0.94	0.97	16
2	0.90	1.00	0.95	9
accuracy			0.97	38
macro avg	0.97	0.98	0.97	38
weighted avg	0.98	0.97	0.97	38

```
from sklearn.model_selection
import GridSearchCV
```

- 다양한 모델 파라미터를 테스트하고자 할 때 사용

```
# Grid Search
print("Performing grid search ... ")

# Parameter Grid
param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001, 0.00001, 10]}

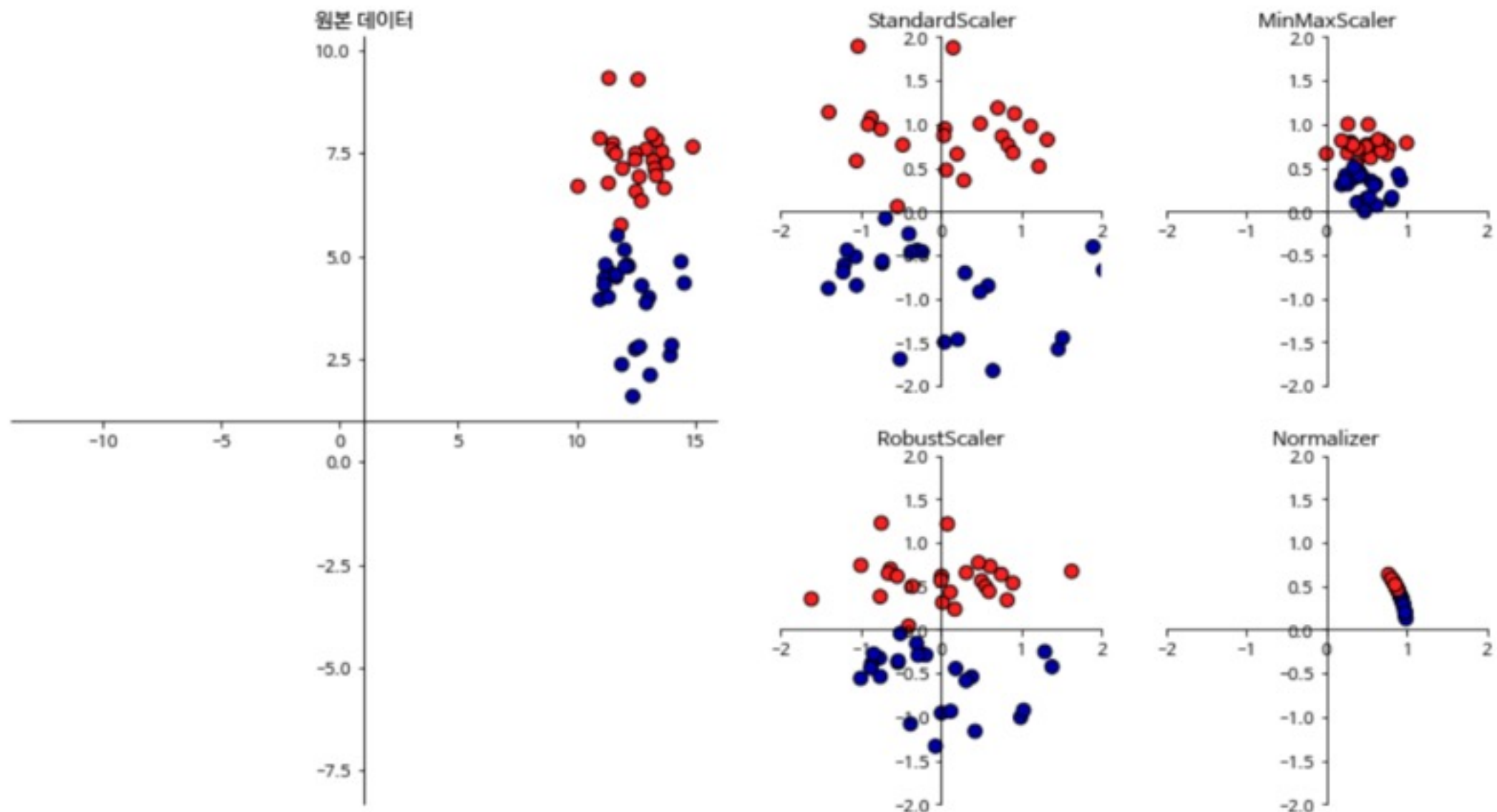
# Make grid search classifier
clf_grid = GridSearchCV(svm.SVC(), param_grid, verbose=1)

# Train the classifier
clf_grid.fit(X_train, y_train)

# clf = grid.best_estimator_()
print("Best Parameters:\n", clf_grid.best_params_)
print("Best Estimators:\n", clf_grid.best_estimator_)
```

from sklearn.preprocessing

- 분석시에 각 특징변수들의 스케일이 다른 경우, 이를 정규화 해줄 필요가 있음
- StandardScaler, MinMaxScaler, RobustScaler, Normalizer



```
from sklearn.preprocessing  
import StandardScaler
```

```
sc = StandardScaler() # 표준화  
sc.fit(x_test)  
sc.fit(x_train) # 표준화 학습 완료
```

```
x_train = sc.transform(x_train)  
x_test = sc.transform(x_test)
```

```
# 표준화 값을 원래 값으로 복귀  
inver_x_train = sc.inverse_transform(x_train)
```

예제 코드

01-sklearn-introduction.ipynb

02-sklearn_sample_examples.ipynb

03-SVM.ipynb