서포트벡터머신 (SVM) 실습자료

PIC
실습

박석희 교수님

실습조교 이승문
ch273404@naver.com

1/16 (목)

PUSAN NATIONAL UNIVERSITY

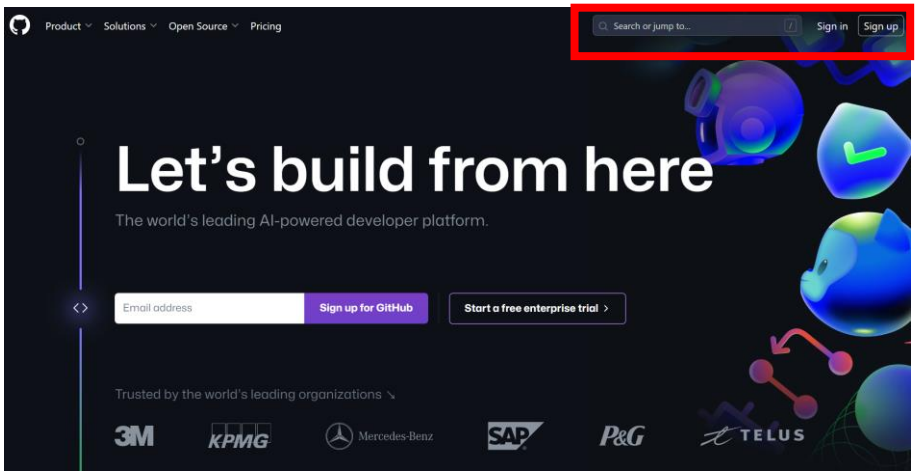
부산대학교 기계공학부
PUSAN NATIONAL UNIVERSITY
SCHOOL OF MECHANICAL ENGINEERING

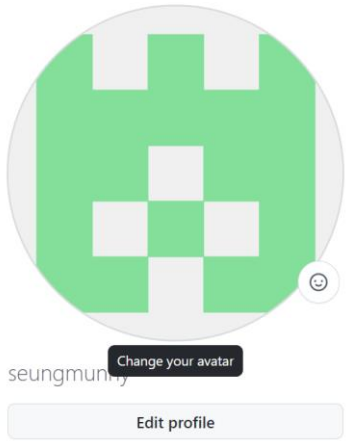
목차

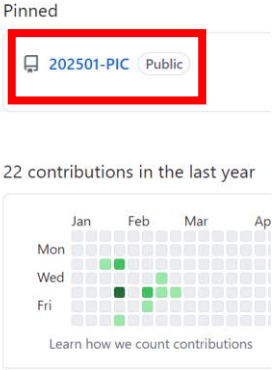
코드 및 데이터 다운로드

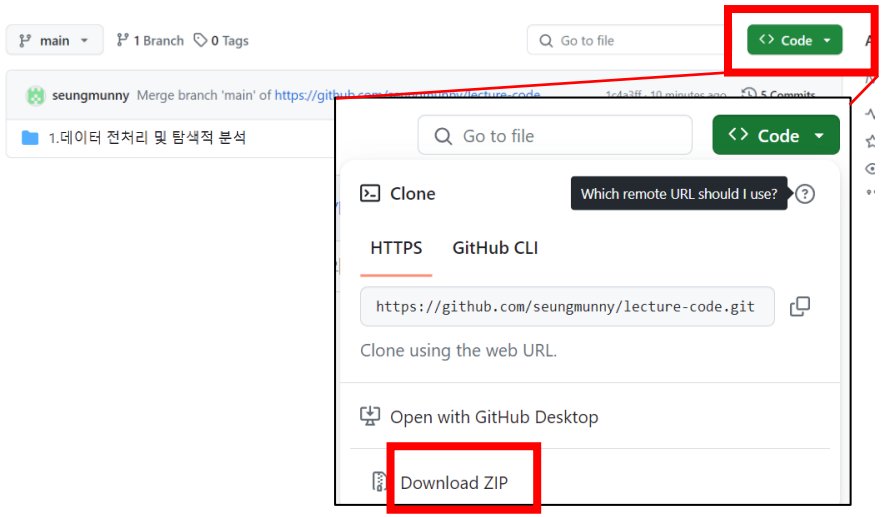
깃허브 접속

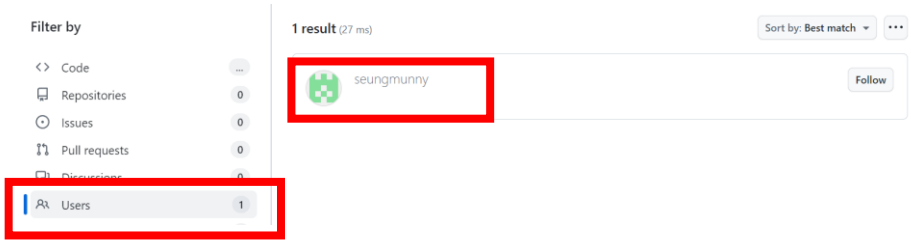
Seungmunny











Linear SVM

https://scikit-learn.org/stable/

❖ 라이브러리 불러오기 및 데이터 생성

#0. 라이브러리 설치

```
!pip install scikit-learn
!pip install matplotlib
!pip install pandas
```

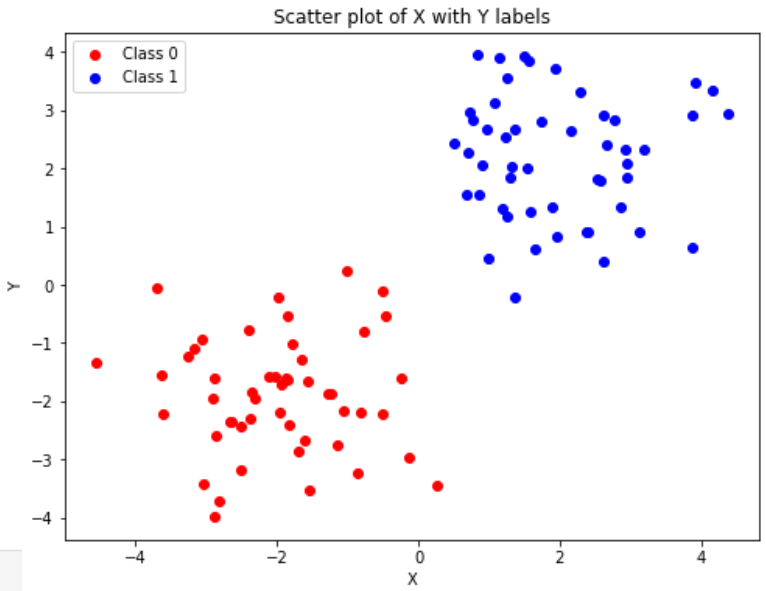
#1. 라이브러리 import

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.inspection import DecisionBoundaryDisplay
```

#2. 데이터 생성

```
# X, Y 데이터 생성
np.random.seed(0)
sample_num=50
X = np.r_[ [-2,-2]+np.random.randn(sample_num, 2) , [2,2]+ np.random.randn(sample_num, 2)]
Y = np.array([0] * sample_num + [1] * sample_num)
#X, Y = make_blobs(n_samples=40, centers=2, random_state=6)
```

- 데이터 시각화



- random.seed는 난수 생성기의 초기 상태를 설정하는 값으로, 시드(seed)를 설정하면 매번 동일한 난수 값을 생성하도록 만듦.

- np.random.randn (20, 2)은 평균이 0이고 표준 편차가 1인 20개의 2차원 데이터 포인트를 생성

- np.r_은 두 군집을 한 배열로 결합

Linear SVM

❖ 모델 학습 및 시각화

```
# 모델 생성 및 시각화
model = svm.SVC()
model.fit(X, Y)
plot_SVM(model)
```

```
def __init__(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0,
shrinking=True, probability=False, tol=0.001, cache_size=200,
class_weight=None, verbose=False, max_iter=-1,
decision_function_shape='ovr', break_ties=False, random_state=None)
```

탭에서 열기 소스 보기

C-Support Vector Classification.
The implementation is based on libsvm. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. For large datasets consider using `~sklearn.svm.LinearSVC` or `~sklearn.linear_model.SGDClassifier` instead, possibly after a `~sklearn.kernel_approximation.Nystroem` transformer or

```
def plot_SVM(model):
    # 모델 학습
    # 그래프 생성
    plt.figure(figsize=(10, 8))
    ax = plt.gca()  # 현재의 축 가져오기

    # 결정 경계 시각화 (DecisionBoundaryDisplay 이용)
    DecisionBoundaryDisplay.from_estimator(
        model,
        X,
        plot_method="contour",
        colors="k",
        levels=[-1, 0, 1],  # -1: 마진 하단, 0: 결정 경계, 1: 마진 상단
        linestyles=["--", "-", "--"],
        ax=ax
    )

    # 각 결정 경계와 마진에 대한 선을 레이블로 표시 (범례 추가용)
    decision_boundary_line = plt.Line2D([], [], color='black', linestyle='-', la
    margin_lines = plt.Line2D([], [], color='black', linestyle='--', label='Marg

    # 데이터 포인트 시각화 (Y값에 따라 색상 지정)
    plt.scatter(X[:, 0], X[:, 1], c=Y, cmap='bwr', edgecolors="k")

    # 서포트 벡터 강조
    plt.scatter(
        model.support_vectors_[:, 0],
        model.support_vectors_[:, 1],
        s=100,
        facecolors="none",
        edgecolors="k",
```

Parameters:

C : float, optional (default=1.0)

Penalty parameter C of the error term.

kernel : string, optional (default='rbf')

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to precompute the kernel matrix.

degree : int, optional (default=3)

Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma : float, optional (default=0.0)

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. If gamma is 0.0 then 1/n_features will be used instead.

coef0 : float, optional (default=0.0)

Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

probability: boolean, optional (default=False) :

Whether to enable probability estimates. This must be enabled prior to calling fit, and will slow down that method.

shrinking: boolean, optional (default=True) :

Whether to use the shrinking heuristic.

tol : float, optional (default=1e-3)

Tolerance for stopping criterion.

cache_size : float, optional

Specify the size of the kernel cache (in MB)

class_weight : {dict, 'auto'}, optional

Set the parameter C of class i to class_weight[i]*C for SVC. If not given, all classes are supposed to have weight one. The 'auto' mode uses the values of y to automatically adjust weights inversely proportional to class frequencies.

verbose : bool, default: False

Enable verbose output. Note that this setting takes advantage of a per-process runtime setting in libsvm that, if enabled, may not work properly in a multithreaded context.

max_iter : int, optional (default=-1)

Hard limit on iterations within solver, or -1 for no limit.

random_state : int seed, RandomState instance, or None (default)

The seed of the pseudo random number generator to use when shuffling the data for probability estimation.

https://scikit-learn.org/0.15/modules/generated/sklearn.svm.SVC.html

Linear SVM

실습

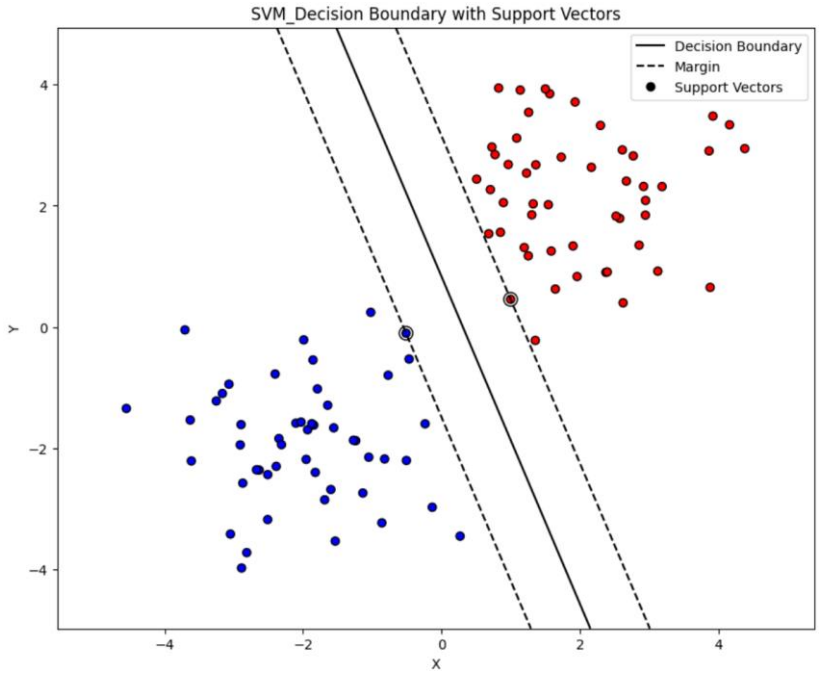
❖ SVM model hyperparameter

- **Slack variable : C (cost)**

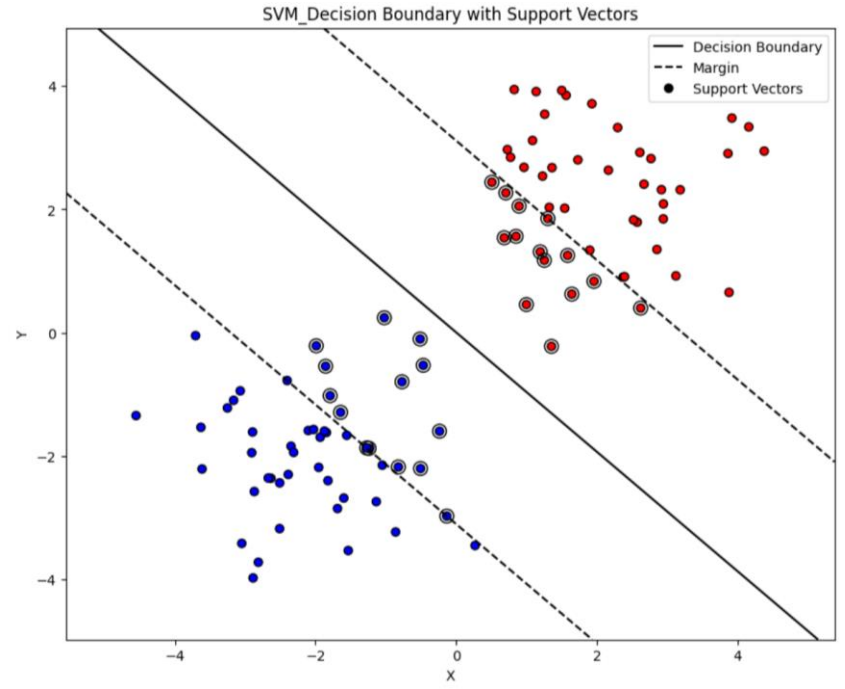
Objective function

$$\min_{w} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \varepsilon i$$

```
# 모델 생성 및 시각화
C = 1
model = svm.SVC(kernel="linear", C=C)
model.fit(X, Y)
plot_SVM(model)
```




```
# 모델 생성 및 시각화
C = 0.01
model = svm.SVC(kernel="linear", C=C)
model.fit(X, Y)
plot_SVM(model)
```



- C는 SVM 모델이 오류를 허용하는 정도를 결정하는 변수
- C가 커지면 학습오류를 허용하지 않고(하드마진) 마진이 줄어들어 오류가 적지만 과적합이 발생할 수 있음
- C가 작아지면 오류를 허용해(소프트마진) 마진이 크고 과소적합이 발생할 수 있음

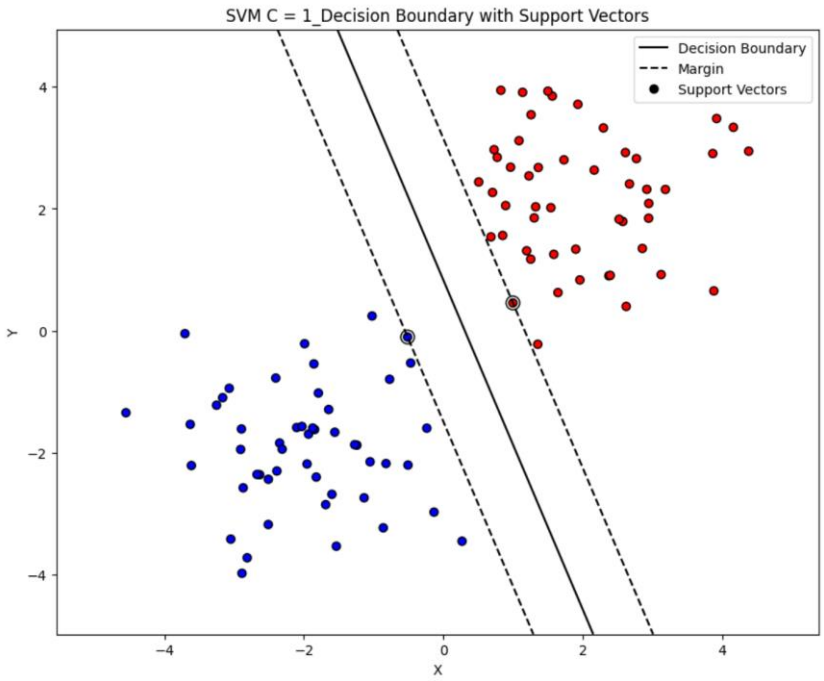
Linear SVM

❖ 데이터 변경

✓ 원 데이터

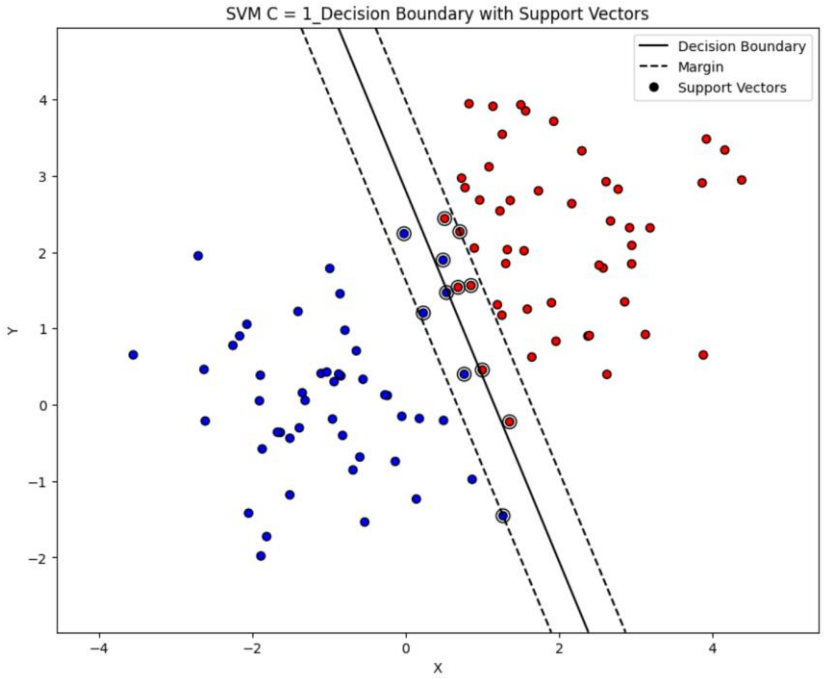
#2. 데이터 생성

```
# X, Y 데이터 생성
np.random.seed(0)
sample_num=50
X = np.r_[ [-2,-2]+np.random.randn(sample_num, 2) , [2,2]+ np.random.randn(sample_num, 2)]
Y = np.array([0] * sample_num + [1] * sample_num)
#X, Y = make_blobs(n_samples=40, centers=2, random_state=6)
```



✓ 중심점이 가까워져 선형분류가 까다로운 데이터

```
# X, Y 데이터 생성
np.random.seed(0)
sample_num=50
X = np.r_[ [-1 ,0]+np.random.randn(sample_num, 2) , [2,2]+ np.random.randn(sample_num, 2)]
Y = np.array([0] * sample_num + [1] * sample_num)
#X, Y = make_blobs(n_samples=40, centers=2, random_state=6)
```



Nonlinear SVM : polynomial kernel

실습

https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py

❖ SVM model hyperparameter

▪ Polynomial kernel

$$K(\mathbf{x}_1, \mathbf{x}_2) = (\gamma \cdot \mathbf{x}_1^\top \mathbf{x}_2 + r)^d$$

where d is the degree (`degree`) of the polynomial, γ (`gamma`) controls the influence of each individual training sample on the decision boundary and r is the bias term (`coef0`) that shifts the data up or down. Here, we use the default value for the degree of the polynomial in the kernel function (`degree=3`). When

Parameters

C : float, default=1.0
Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty. For an intuitive visualization of the effects of scaling the regularization parameter C, see `sphx_glr_auto_examples_svm_plot_svm_scale_c.py`.

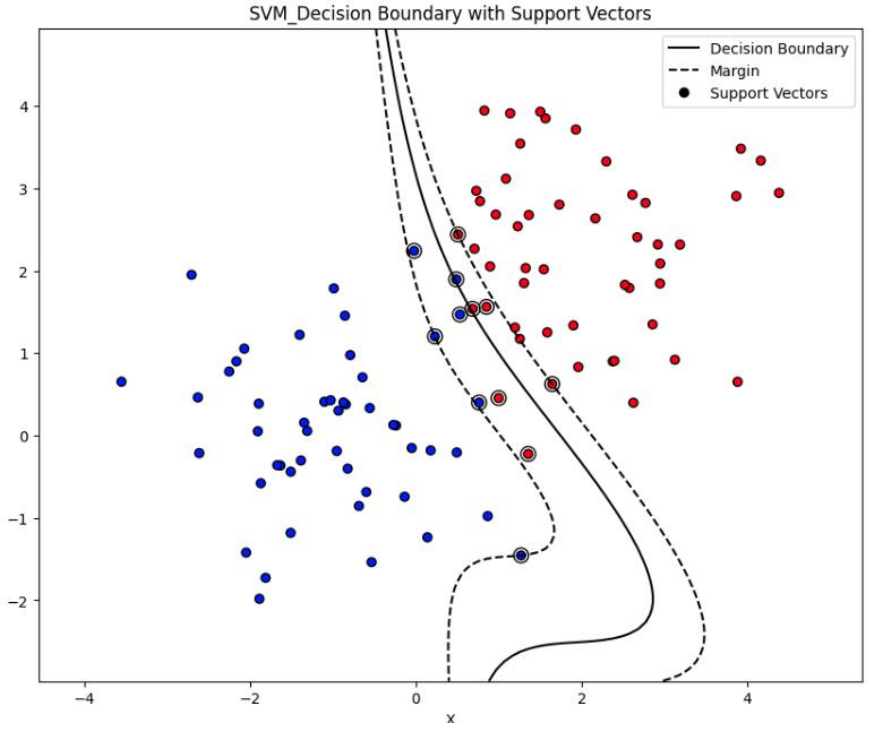
kernel : {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'
Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`. For an intuitive visualization of different kernel types see `sphx_glr_auto_examples_svm_plot_svm_kernels.py`.

degree : int, default=3
Degree of the polynomial kernel function ('poly'). Must be non-negative. Ignored by all other kernels.

gamma : {'scale', 'auto'} or float, default='scale'
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.
- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features
- if float, must be non-negative.

coef0 : float, default=0.0
Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

```
# 모델 생성 및 시각화
C = 10
model = svm.SVC(kernel="poly",degree=3,C=C)
model.fit(X, Y)
plot_SVM(model)
```



- 큰 gamma 값은 데이터 포인트 간 거리에 매우 민감하게 반응. 가까운 데이터 포인트들만이 강한 유사성을 갖도록 학습

Nonlinear SVM : RBF kernel

실습

https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py

❖ SVM model hyperparameter

▪ RBF kernel

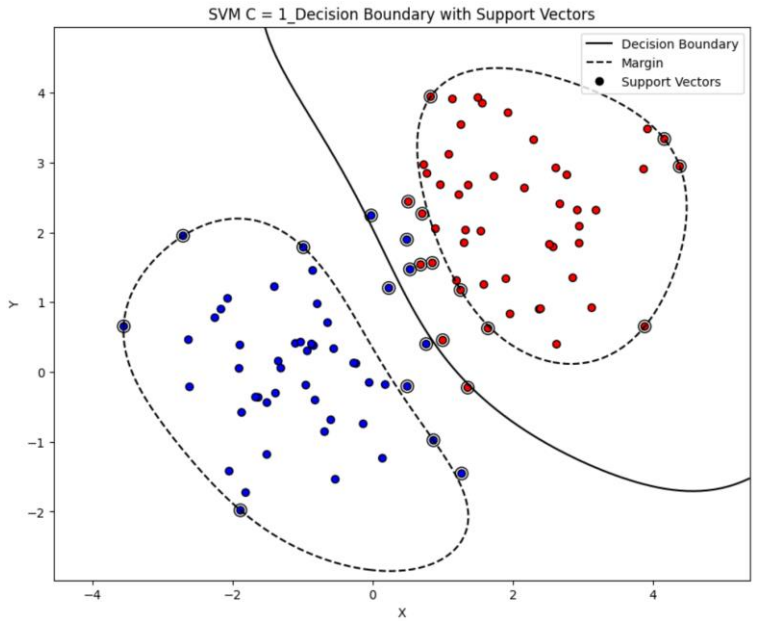
$$K(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\gamma \cdot \|\mathbf{x}_1 - \mathbf{x}_2\|^2\right)$$

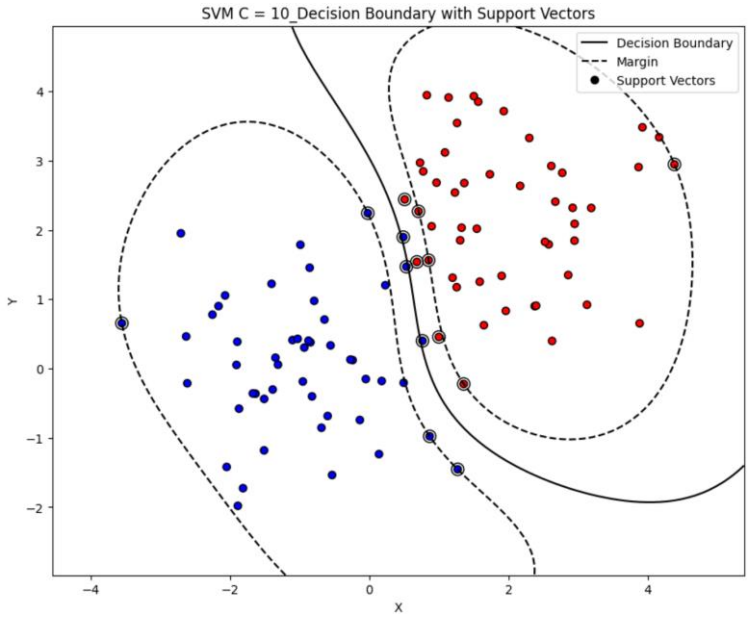
where γ (`gamma`) controls the influence of each individual training sample on the decision boundary.

```
# 모델 생성 및 시각화
C = 1
model = svm.SVC(kernel="rbf", C=C)
model.fit(X, Y)
plot_SVM(model)
```

gamma : {'scale', 'auto'} or float, default='scale'
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features
- if float, must be non-negative.





- Gamma 값이 클 수록 결정경계가 구부러지는 정도가 커져 오버피팅 위험

Nonlinear SVM : Sigmoid kernel

실습

❖ SVM model hyperparameter

▪ Sigmoid kernel

https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py

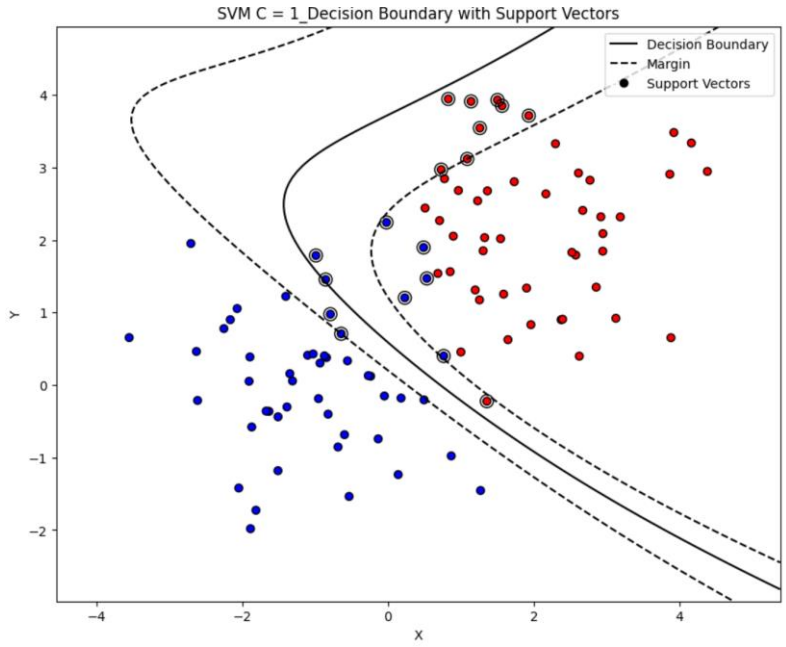
$$K(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\gamma \cdot \mathbf{x}_1^\top \mathbf{x}_2 + r)$$

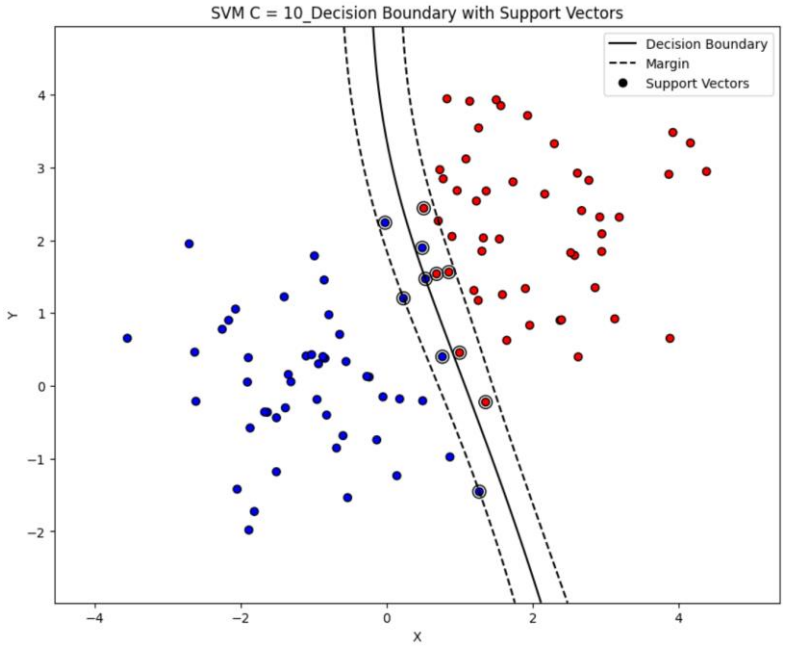
where the kernel coefficient γ (`gamma`) controls the influence of each individual training sample on the decision boundary and r is the bias term (`coef0`) that shifts the data up or down.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features
- if float, must be non-negative.

coef0 : float, default=0.0
Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

```
# 모델 생성 및 시각화
C = 1
model = svm.SVC(kernel="sigmoid", C=C)
model.fit(X, Y)
plot_SVM(model, C)
```





SVM model hyperparameter

❖ SVM hyperparameter tuning

https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py

Hyperparameter tuning plot

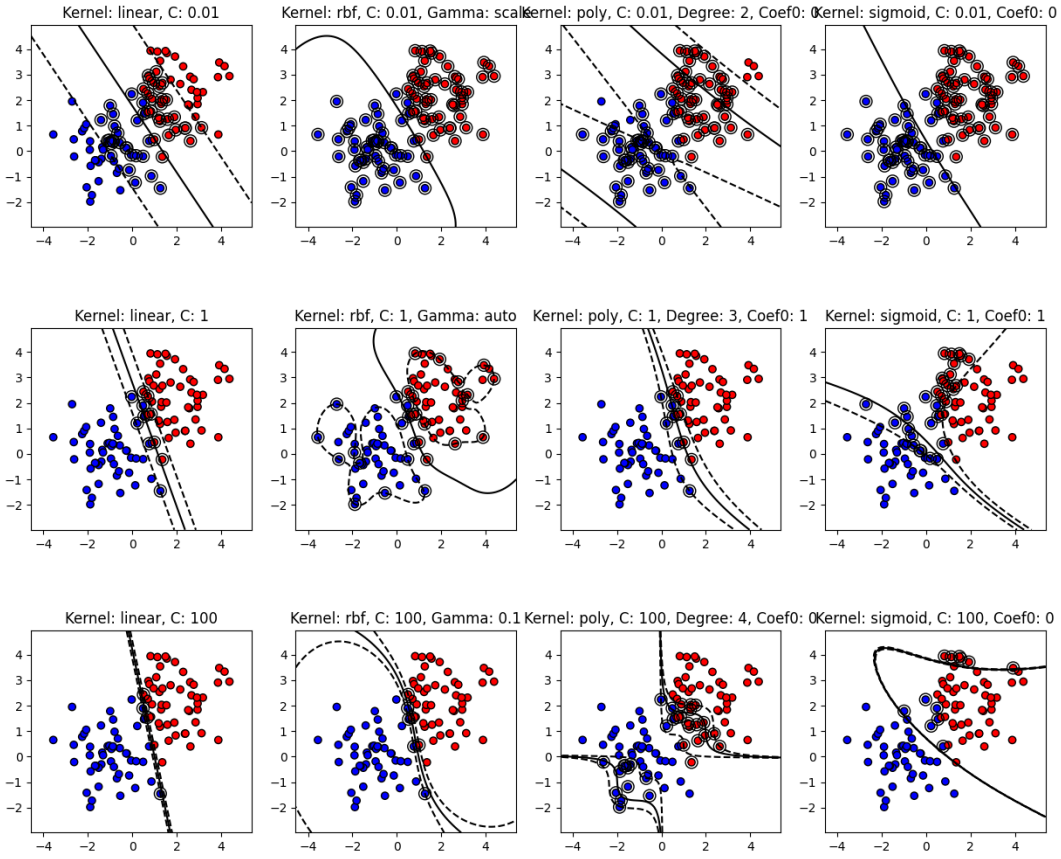
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.inspection import DecisionBoundaryDisplay

# 데이터 생성
np.random.seed(0)
sample_num = 50
X = np.r_[ [-1, 0] + np.random.randn(sample_num, 2), [2, 2] +
Y = np.array([0] * sample_num + [1] * sample_num)

# 하이퍼파라미터 설정
C_values = [0.01, 1, 100]  # C 값들
kernels = ['linear', 'rbf', 'poly', 'sigmoid']  # 다양한 커널
gammas = ['scale', 'auto', 0.1]  # gamma 값들
degrees = [2, 3, 4]  # 다항식 커널을 위한 degree
coef0s = [0, 1]  # 다항식과 시그모이드 커널을 위한 coef0

# 모델과 경계선을 시각화하는 함수
def plot_HTSVM(ax, model, X, Y, title):
    model.fit(X, Y)

    # 결정 경계 시각화
    DecisionBoundaryDisplay.from_estimator(
```



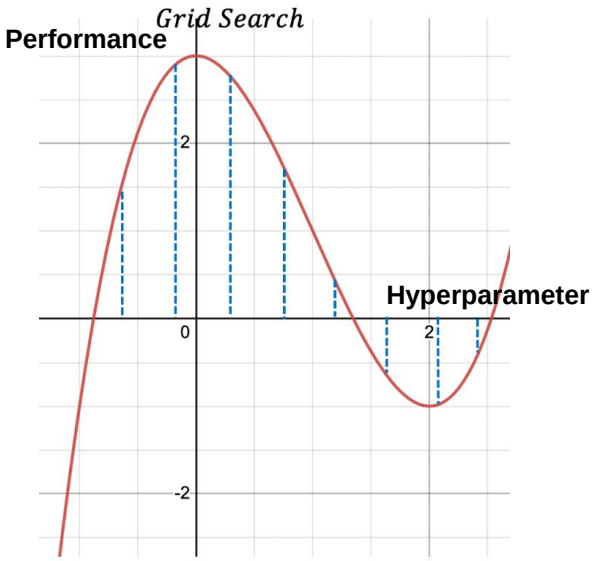
- 데이터의 형태 및 복잡도에 따라 hyperparameter를 적절히 선택하는 것이 필요함

SVM model hyperparameter

https://heytech.tistory.com/389

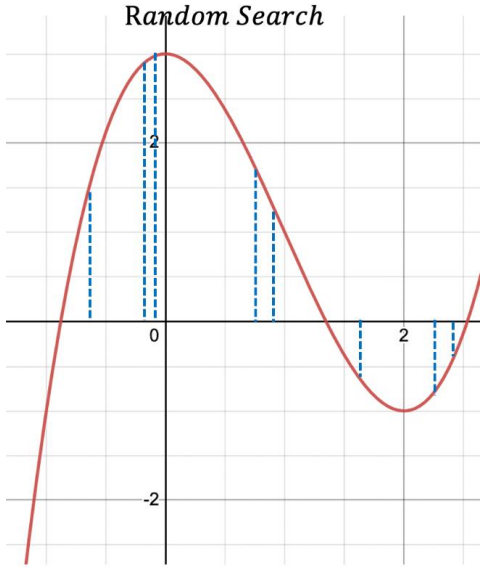
❖ 하이퍼파라미터 최적화 기법

1. Grid Search



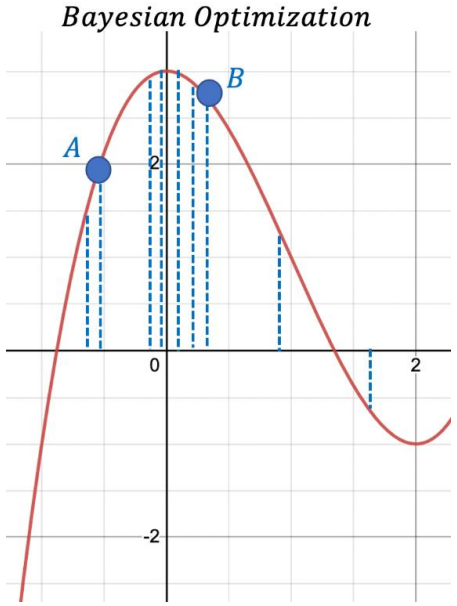
하이퍼파라미터를 일정한 간격으로 변경하며 최적의 파라미터를 찾아가는 기법

2. Random Search



랜덤 서치(Random Search)는 임의의 하이퍼파라미터를 선정하는 과정을 통해 최적의 해를 찾아가는 기법

3. Bayesian Optimization



베이지안 최적화(Bayesian Optimization)는 최적의 해 근처의 하이퍼파라미터를 위주로 탐색하는 작업과 임의의 새로운 하이퍼파라미터를 탐색하는 과정을 반복하여 최적의 해를 탐색하는 기법

SVM Kaggle 예제

https://www.kaggle.com/code/rajeevnair676/svm-hyperparameter-tuning

❖ 데이터셋 확인

- **Heart failure prediction dataset**

SVM-Hyperparameter Tuning

Python · Heart Failure Prediction Dataset

Notebook Input Output Logs Comments (24)

Run
206.0s

Version 5 of 5

SVM

Heart Disease Prediction Using SVM

Show hidden code

/kaggle/input/heart-failure-prediction/heart.csv

Heart Failure Prediction Dataset

11 clinical features for predicting heart disease events.

Age	Sex	ChestPainType	RestingBP	Cholesterol
age	sex	chest pain type	resting blood pressure	serum cholesterol
28 – 77	M 79% F 21%	ASY 54% NAP 22% Other (219) 24%	0 – 200	0 – 603

FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak
fasting blood sugar	resting electrocardiogram results	maximum heart rate achieved	exercise induced angina	oldpeak = ST
0 – 1	Normal 60% LVH 20% Other (178) 19%	60 – 202	true 371 40% false 547 60%	-2.6 – 6.2

ST_Slope	HeartDisease
the slope of the peak exercise ST segment	target
Flat 50% Up 43% Other (63) 7%	0 – 1

Data : 918개
Feature : 11 (Categorical 5개, Numerical 6개)

SVM Kaggle 예제 : Heart Disease Prediction Using SVM

실습

https://www.kaggle.com/code/rajeevnair676/svm-hyperparameter-tuning

❖ 데이터 전처리

- 라이브러리 및 데이터 import

```
# 라이브러리 및 데이터 import

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
import pandas as pd

#Loading the CSV file
df = pd.read_csv('./data/heart.csv')
df.head()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

SVM은 범주형 데이터 처리가 불가능함.
따라서 범주형 데이터를 숫자로 변환하는 것이 필요

```
#Preparing the dataset for training
df=pd.get_dummies(df) # 원-핫 인코딩 : 범주형 데이터를 숫자로 변환하는 방법
X = df.drop('HeartDisease', axis=1)
y = df['HeartDisease']
df.head()
```

SVM Kaggle 예제 : Heart Disease Prediction Using SVM

https://cori.tistory.com/163

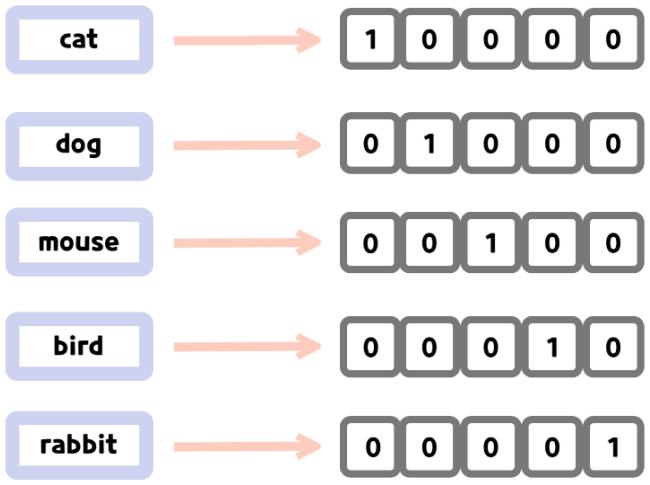
❖ 데이터 전처리

▪ 범주형 데이터처리(One hot encoding)

- N개의 클래스를 N차원의 One-Hot 벡터로 표현되도록 변환하는 기법
(고유값들을 피처로 만들고 정답에 해당하는 열은 1, 나머지는 0으로 표현)

- 숫자의 차이가 모델에 영향을 미치는 선형 계열 모델에서 범주형 데이터로 변환할 때 사용

Example



One-hot encoding 적용 전

```
#Loading the CSV file
df = pd.read_csv('/content/heart.csv')
df.head()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat

One-hot encoding 적용 후

```
#Preparing the dataset for training
df=pd.get_dummies(df) # 원-핫 인코딩 : 범주형 데이터를 숫자로 변환하는 방법
X = df.drop('HeartDisease', axis=1)
y = df['HeartDisease']
df.head()
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease	Sex_F	Sex_M	ChestPainType_ASY	...	ChestPainType_NAP	ChestPainType_TA
0	40	140	289	0	172	0.0	0	False	True	False	...	False	False
1	49	160	180	0	156	1.0	1	True	False	False	...	True	False
2	37	130	283	0	98	0.0	0	False	True	False	...	False	False
3	48	138	214	0	108	1.5	1	True	False	True	...	False	False

(False는 0, True는 1)

SVM Kaggle 예제 : Heart Disease Prediction Using SVM

❖ 모델 학습 및 평가

https://velog.io/@hyeongjun/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D-%EC%82%AC%EC%9D%B4%ED%82%B7%EB%9F%B0sklearn-%EA%B8%B0%EC%B4%88

https://white-joy.tistory.com/9

https://velog.io/@crescent702/%EB%B2%88%EC%97%AD-Evaluation-Metrics-for-Machine-Learning-Models

```
#Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
```

```
#Fitting the model on SVC with default parameters
model_svc = SVC()
model_svc.fit(X_train,y_train)
```

SVC

SVC()

```
#Evaluationg the trained model
pred_svc = model_svc.predict(X_test)
accuracy_score(y_test,pred_svc)
```

0.75

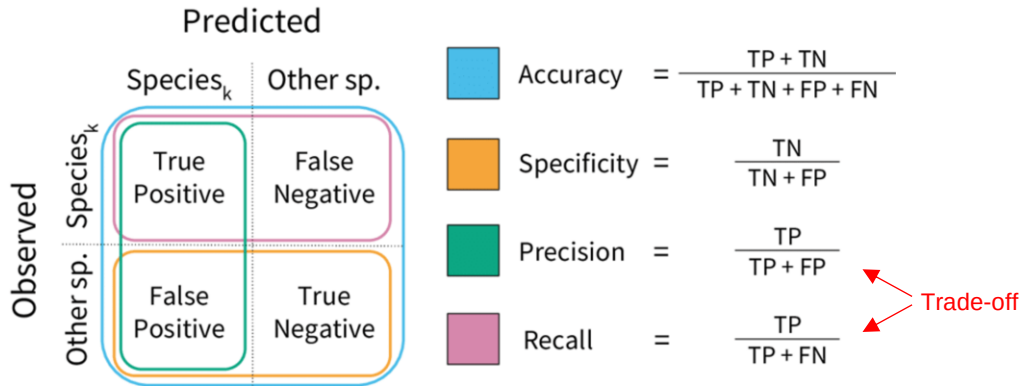
```
#Building classification report
print(classification_report(y_test,pred_svc))
```

	precision	recall	f1-score	support
0	0.73	0.70	0.72	125
1	0.76	0.79	0.78	151
accuracy			0.75	276
macro avg	0.75	0.75	0.75	276
weighted avg	0.75	0.75	0.75	276

Default hyperparameters

C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None

Evaluation metrics for classification



Accuracy (정확도): 모델이 전체 문제 중에서 정답을 맞춘 비율
Precision (정밀도): 모델이 positive 예측 중 실제로 정답이 positive인 비율
Recall (재현율): 정답이 positive인 것들 중에서 모델이 positive라고 예측한 비율
Specificity (특이도): 정답이 negative인 것들 중 모델이 negative라고 예측한 비율
F1-score : precision과 recall의 조화평균

SVM Kaggle 예제 : Heart Disease Prediction Using SVM

https://velog.io/@jiazzang/%EB%8D%B0%EC%9D%B4%ED%84%B0-%EC%A0%84%EC%B2%98%EB%A6%AC-%EB%8D%B0%EC%9D%B4%ED%84%B0-%EC%8A%A4%EC%BC%80%EC%9D%BC%EB%A7%81StandardScaler-MinMaxScaler-Robust

❖ 데이터 전처리 후 성능 비교

▪ 데이터 스케일링 (Data scaling)

- 데이터 스케일링(Data Scaling)이란 서로 다른 변수의 값 범위를 일정한 수준으로 맞추는 작업.

- SVM은 **결정 경계**를 학습하는 과정에서 **서포트 벡터**와 **마진**을 계산하는데, 여기서 **거리**를 중요하게 사용.
- 선형 모델(로지스틱 회귀, 선형 회귀 등)은 **가중치(weight)**와 특성 값의 내적을 계산하여 예측을 수행

- 예를들어, 하나의 특성 범위가 [0, 1]인데, 다른 특성의 범위가 [0, 1000]이라면, 후자의 특성이 훨씬 더 큰 영향을 미치게 됨.
- 따라서, SVM과 선형 모델에서는 특성 간 스케일 차이가 큰 경우, 모델 성능이 저하될 수 있음.

1. 표준화(Standardization): **변수 각각의 평균을 0, 분산을 1로 만들어주는 스케일링 기법**

$$z = \frac{x_i - mean(x)}{stdev(x)}$$

2. 정규화(Normalization): 일반적으로 서로 다른 변수의 크기를 통일하기 위해 크기를 변환

$$\frac{x_i - \min(x)}{\max(x) - \min(x)}$$

3. 로버스트(Robust): 데이터의 중앙값 = 0, IQR = 1이 되도록 스케일링

$$\frac{x_i - median(x)}{Q3 - Q1}$$

표준화(Standardization) 적용

```
#데이터 표준화
```

```
#Scaling the features using pipeline
pipeline = Pipeline([
            ('std_scaler',StandardScaler()),
                    ])
scaled_X_train = pipeline.fit_transform(X_train)
scaled_X_test = pipeline.transform(X_test)

#Fitting the model on SVC with default parameters
model_svc = SVC()
model_svc.fit(scaled_X_train,y_train)
#Calculating predictions, and accuracy score
pred_svc = model_svc.predict(scaled_X_test)
accuracy_score(y_test,pred_svc)
```

```
0.8840579710144928
```

(스케일링 적용전 : 0.75)

SVM Kaggle 예제 : Heart Disease Prediction Using SVM

https://velog.io/@hyeongjun/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D-%EC%82%AC%EC%9D%B4%ED%82%B7%EB%9F%B0sklearn-%EA%B8%B0%EC%B4%88

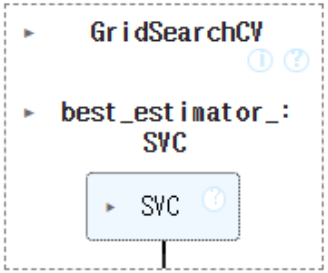
❖ 하이퍼파라미터 최적화

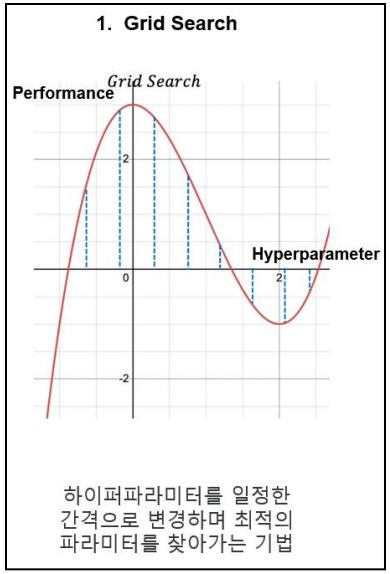
- **Hyperparamter tuning using Gridsearch**

```
svm = SVC()
# param_grid = {'C':[0.01,0.05,0.1,1,10, 100, 1000],'kernel':['linear','rbf'], 'gamma':['scale','auto'] }
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf','linear']}
grid = GridSearchCV(svm,param_grid)
```

하이퍼 파라미터 범위 설정

```
#Fitting the model
grid.fit(scaled_X_train,y_train)
```





하이퍼파라미터를 일정한 간격으로 변경하며 최적의 파라미터를 찾아가는 기법

```
#Calculating the accuracy of tuned model
grid_svc = grid.predict(scaled_X_test)
accuracy_score(y_test,grid_svc)
```

```
0.8913043478260869
```

(하이퍼 파라미터 최적화 전 : Accuracy 0.884)

```
print(grid.best_params_)
print(grid.best_estimator_.get_params())
```

최적화 된 하이퍼파라미터

```
{'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
{'C': 100, 'break_ties': False, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape'
ernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.001, 'ver
```

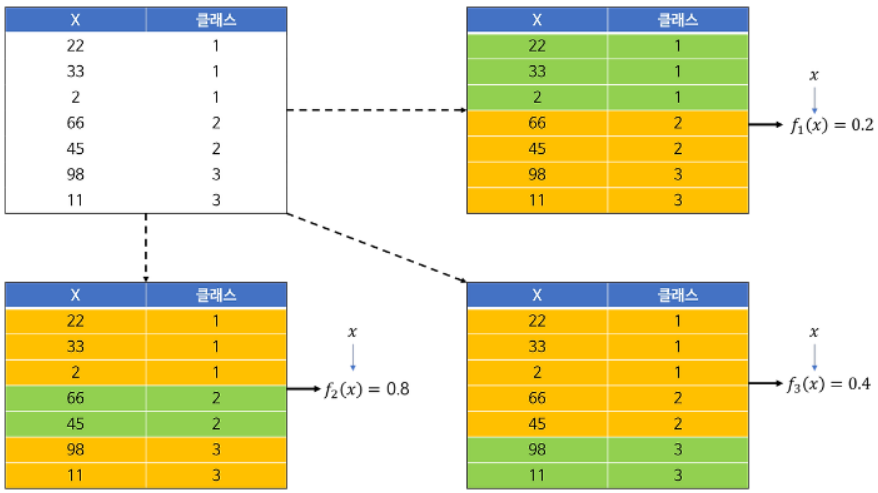
SVM 기반 다중 분류

https://www.baeldung.com/cs/svm-multiclass-classification
https://zephyrus1111.tistory.com/205

❖ Multi-classification

One-vs-Rest (OVR)

- 다중 클래스 무제를 여러 개의 이진 분류 문제로 바꿈
- Class 개수가 m 일때 m 개의 SVM을 활용



one vs rest 설명

$$\underset{k\in\{1,2,3\}}{\arg\max}\{f_1(x), f_2(x), f_3(x)\} = \underset{k\in\{1,2,3\}}{\arg\max}\{0.2, 0.8, 0.4\} = 2$$

One-vs-One (OVO)

- 다중 클래스 문제를 여러 개의 이진 분류 문제로 바꿈
- Class 개수가 m 일때 m(m-1)/2 개의 SVM을 활용

이번에도 라벨이 3개 있다고 하고 각 라벨은 1, 2, 3이다. 이 경우 아래와 같이 3(=(3x2)/2) 개의 이진 분류기를 학습하게 된다.

j	클래스 쌍
1	1, 2
2	1, 3
3	2, 3

새로운 x에 대하여 j번째 학습된 이진 분류기의 라벨이 k인 확률 $f_{jk}(x)$가 다음과 같이 계산되었다고 하자.

j	예측 확률
1	$f_{11}(x) = 0.2, f_{12}(x) = 0.8$
2	$f_{21}(x) = 0.4, f_{23}(x) = 0.6$
3	$f_{32}(x) = 0.5\ f_{33}(x) = 0.5$

이 경우 예측 확률의 최대값은 $f_{12}(x) = 0.8$이고 이에 대응하는 라벨인 2를 최종 예측 라벨로 선정하게 된다.

$$J^* = \underset{j\in\{1,\ldots,J\}}{\arg\max} f_j(x) \qquad f_{J^*}(x)\text{'s Label}$$

SVM 기반 다중 분류

실습

❖ Multi-classification

https://scikit-learn.org/stable/auto_examples/svm/plot_svm_tie_breaking.html
https://zephyrus1111.tistory.com/205

#Multi-class classification

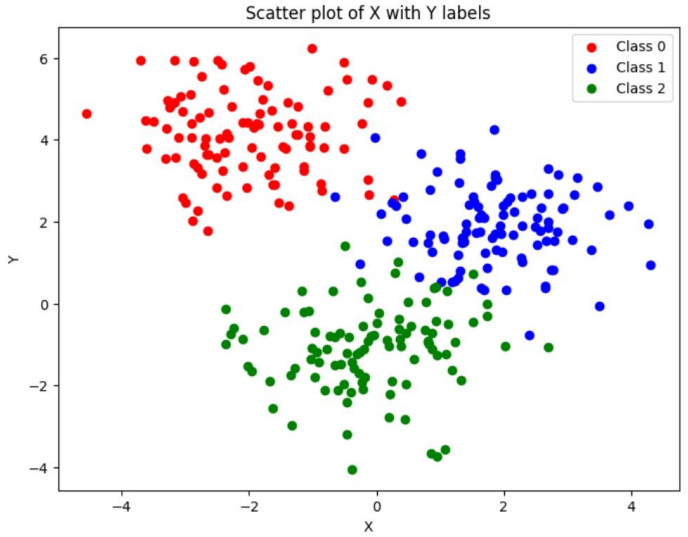
```
np.random.seed(0)
sample_num=100
X = np.r_[ [-2 ,4]+np.random.randn(sample_num, 2) , [2,2]+ np.random.randn(sample_num, 2),[0,-1]+ np.random.randn(sample_num, 2)]
Y = np.array([0] * sample_num + [1] * sample_num+ [2] * sample_num)

# 산점도 그리기
plt.figure(figsize=(8, 6))
plt.scatter(X[Y == 0, 0], X[Y == 0, 1], color='red', label='Class 0')
plt.scatter(X[Y == 1, 0], X[Y == 1, 1], color='blue', label='Class 1')
plt.scatter(X[Y == 2, 0], X[Y == 2, 1], color='green', label='Class 2')

# 그래프 제목과 축 라벨 추가
plt.title("Scatter plot of X with Y labels")
plt.xlabel("X")
plt.ylabel("Y")

# 범례 추가
plt.legend()

# 그래프 출력
plt.show()
```



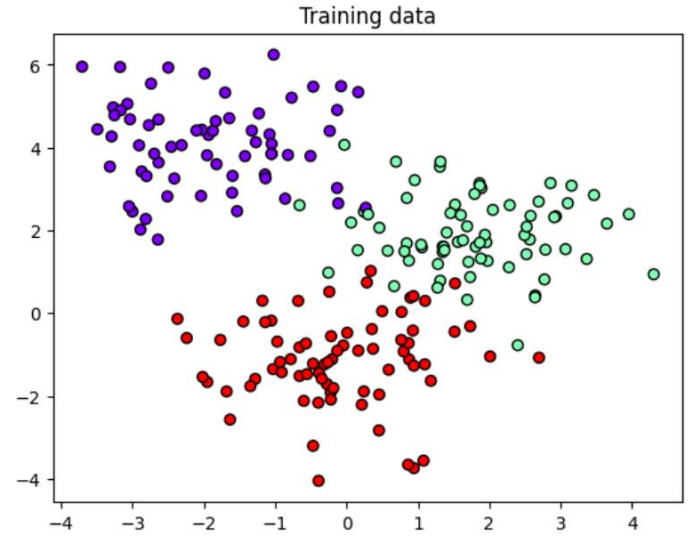

```
# 학습 및 테스트 데이터 분리
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=42)

# 선형 커널을 사용하는 SVM 모델 생성 (기본적으로 OVR 방식 사용)
model = svm.SVC(kernel="linear", C=1.0,decision_function_shape='ovr')
model.fit(X_train, y_train)

# 예측
y_pred = model.predict(X_test)

# 정확도 출력
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# 데이터 및 결정 경계 시각화
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap='rainbow', edgecolors='k')
plt.title("Training data")
plt.show()
```



https://www.baeldung.com/cs/svm-multiclass-classification

SVR (Support Vector Regression)

https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html

❖ Support vector 회귀

- SVM은 두 클래스를 구분하는 **최대 마진 분류기**를 사용하지만, 회귀에서는 데이터 포인트로부터 **일정한 허용 오차(마진)를 가진 최적의 회귀 함수**를 찾음

- SVR의 목표는 데이터 포인트들이 결정 경계와 ε-튜브 내에 최대한 많이 포함되도록 하면서, 최적의 회귀 평면(또는 곡선)을 얻는 것임

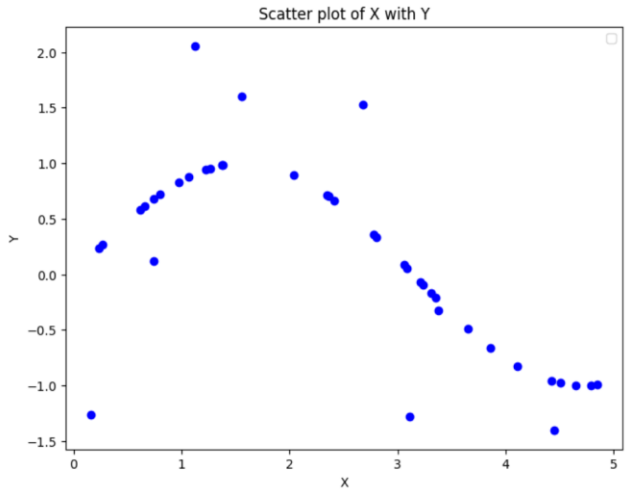
데이터 생성

```
import matplotlib.pyplot as plt
import numpy as np

from sklearn.svm import SVR
```

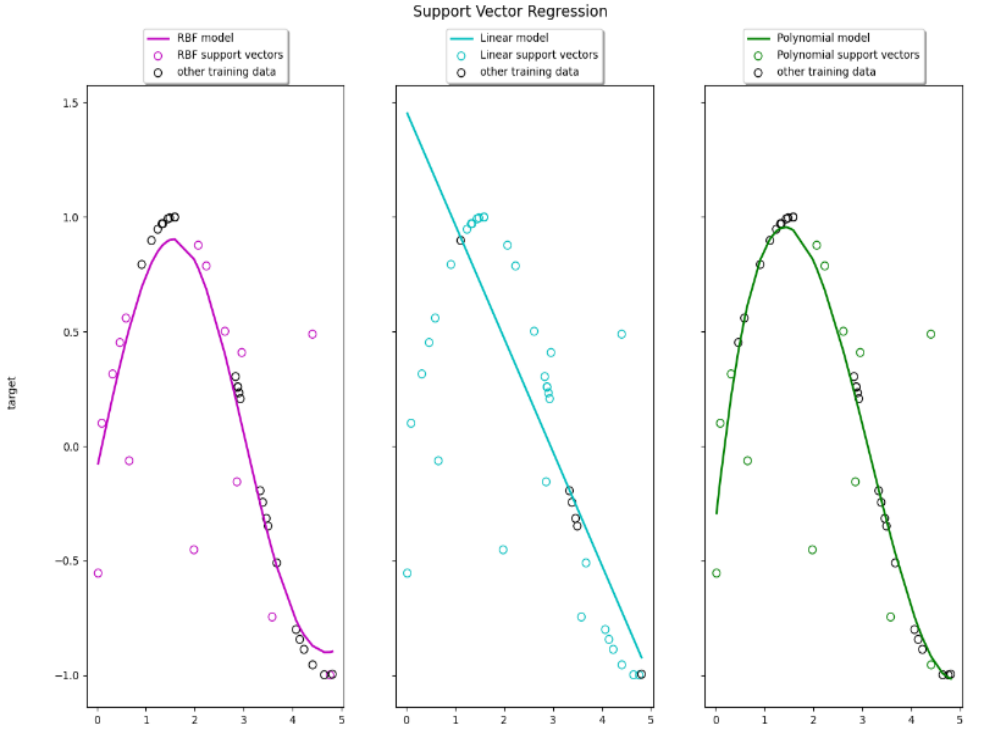
```
X = np.sort(5 * np.random.rand(40, 1), axis=0)
y = np.sin(X).ravel()

# add noise to targets
y[::5] += 3 * (0.5 - np.random.rand(8))
```



모델 생성

```
svr_rbf = SVR(kernel="rbf", C=100, gamma=0.1, epsilon=0.1)
svr_lin = SVR(kernel="linear", C=100, gamma="auto")
svr_poly = SVR(kernel="poly", C=100, gamma="auto", degree=3, epsilon=0.1, coef0=1)
```



감사합니다

질문 : PDF 파일 경고 마크 인식 task

경고 마크 인식을 위해 object detection 모델 도입 추천함
YOLO 모델이 이에 속함. 여러 버전이 있으며 상황에 따라 활용

https://bong-sik.tistory.com/30
https://ropiens.tistory.com/44
https://foss4g.tistory.com/1647
https://lynnshin.tistory.com/48
https://yobbicorgi.tistory.com/34?category=476484
https://minding-deep-learning.tistory.com/19
https://magicode.tistory.com/55