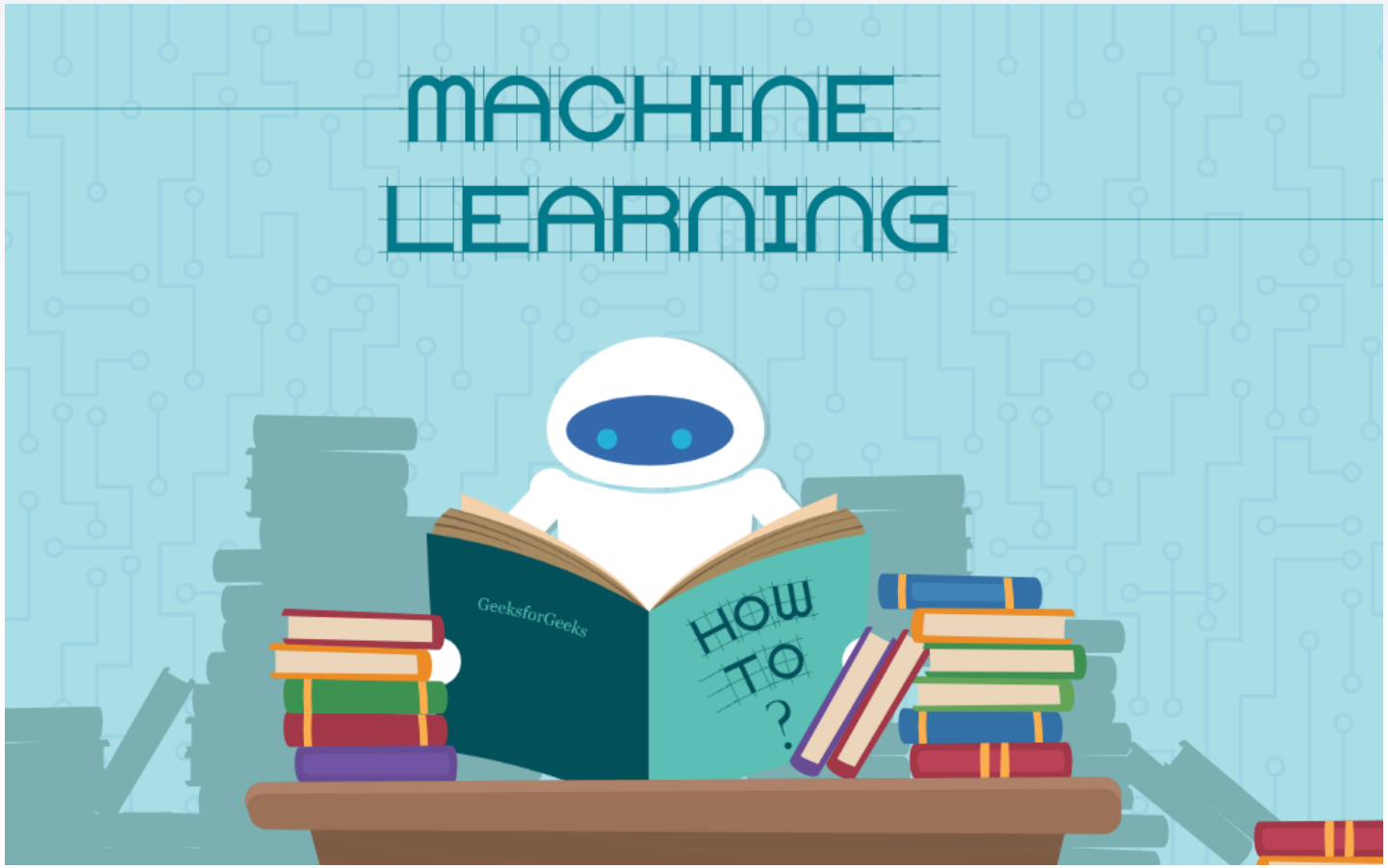
머신러닝 이론과 주요모델



kt 박 경 규

데이터

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

머신러닝

머신러닝을 실현하기 위한 방법론이 여러개가 있지만, 그 중 지도학습(Supervised Learning)과 비지도학습(Un-Supervised Learning)이 대표적입니다.

■ 지도학습(supervised learning)

- 학습시 정답을 알려 주면서 진행하는 학습으로, 학습시 데이터와 레이블(정답)이 함께 제공됩니다.
- **레이블(Label)** = 정답, 실제값, 타깃, 클래스, y
- 예측된 값 = 예측값, 분류값, $\hat{y}$ (y hat)
- 데이터마다 레이블을 달기 위해 많은 시간을 투자해야 합니다.
- 지도학습 **모델**에는 **분류모델**(이진분류, 다중분류)와 **예측(회귀)모델**(주가예측 등)이 있습니다.

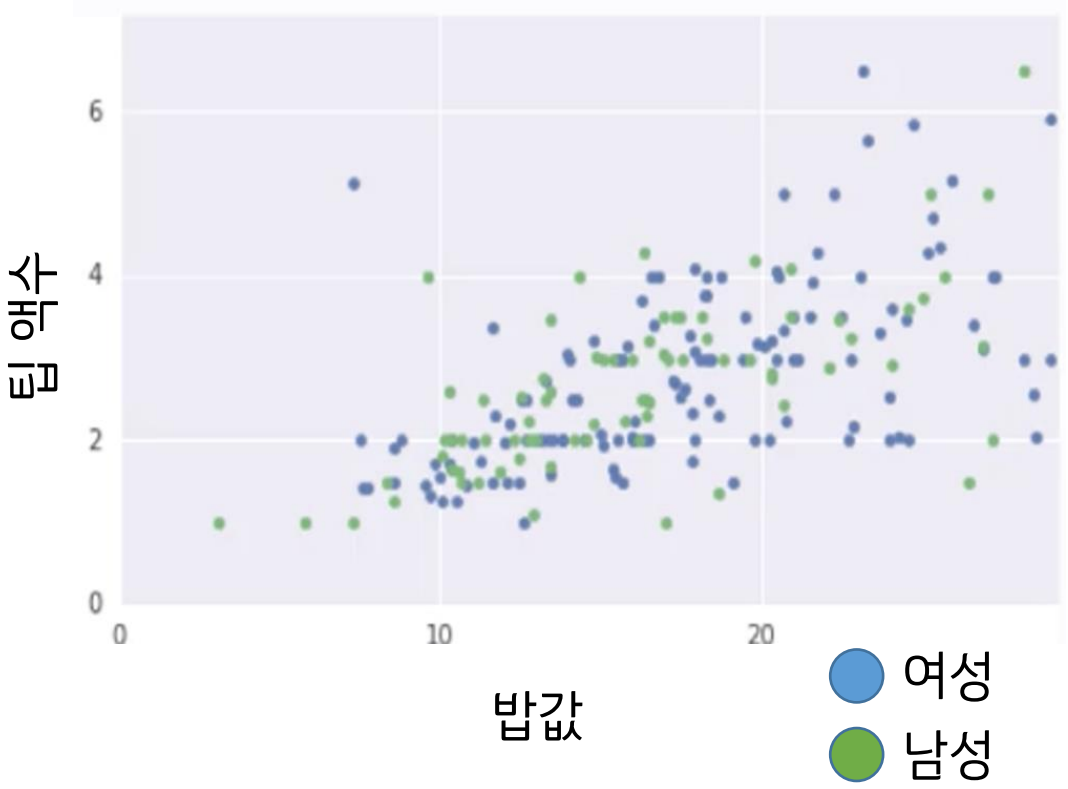
■ 비지도학습(unsupervised learning)

- 레이블(정답) 없이 진행되는 학습으로, 데이터 자체에서 패턴을 찾아내야 할 때 사용합니다.
- 비지도학습의 대표적인 예는 군집화(clustering)와 차원축소가 있습니다.

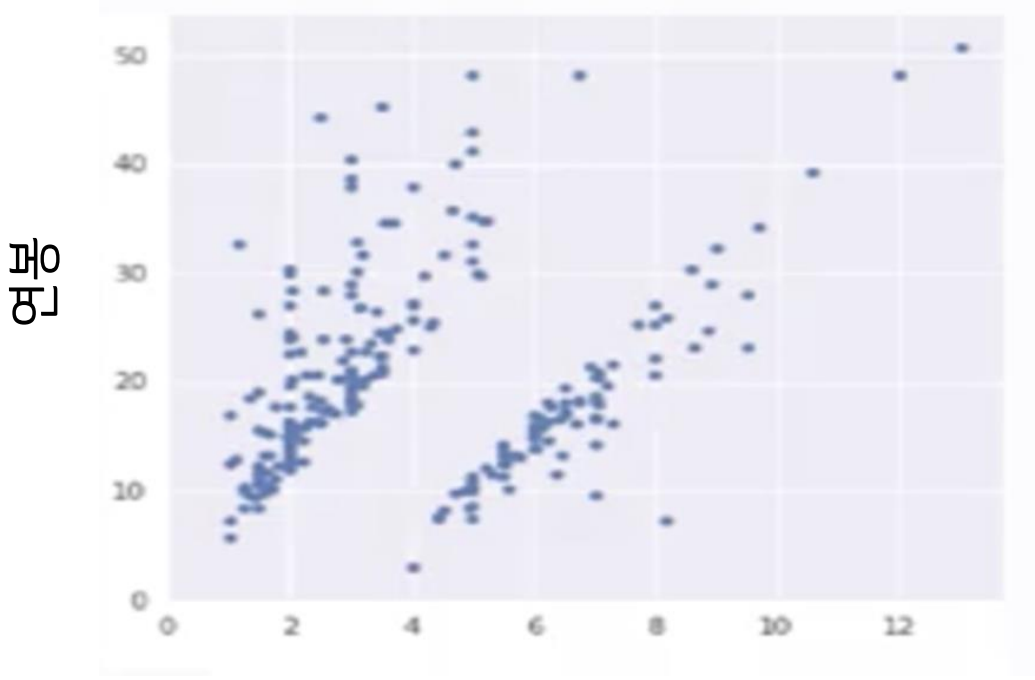
지도학습 vs 비지도학습

지도학습에서는 데이터가 레이블링 되어 있으며, 축적된 데이터로부터 클래스를 분류 또는 미래의 값을 예측합니다. 비지도학습에선 데이터가 레이블링 되어 있지 않습니다.

성별에 따른 팁 액수의 크기



패스트트랙(Fast-track) 인가?



지도학습 모델 종류

모델은 데이터들의 패턴을 대표할 수 있는 함수로, 입력을 독립변수 출력을 종속변수라고 합니다.

■ 모델이란?

- 데이터들의 패턴을 대표할 수 있는 함수, 예) $f(x) = ax + b$
- 함수의 입력은 독립변수이고 출력은 종속변수로, 독립변수들에 의해 출력값이 정해집니다.

■ 분류 모델(Classification)

- 레이블의 값들이 이산적으로 나눠질 수 있는 문제에 사용합니다.
- 예) 밥값이 많이 나오면 남성, 적게 나오면 여성

■ 예측 모델(Regression, 회귀모델)

- 레이블의 값들이 연속적인 문제에 사용합니다.
- 예) 밥값이 많이 나올수록, 팁의 크기도 계속 커진다.

지도학습 모델 종류

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

예측 모델(회귀 모델)
팁의 크기를 예측

분류 모델
손님의 성별을 예측

지도학습 데이터셋 구조

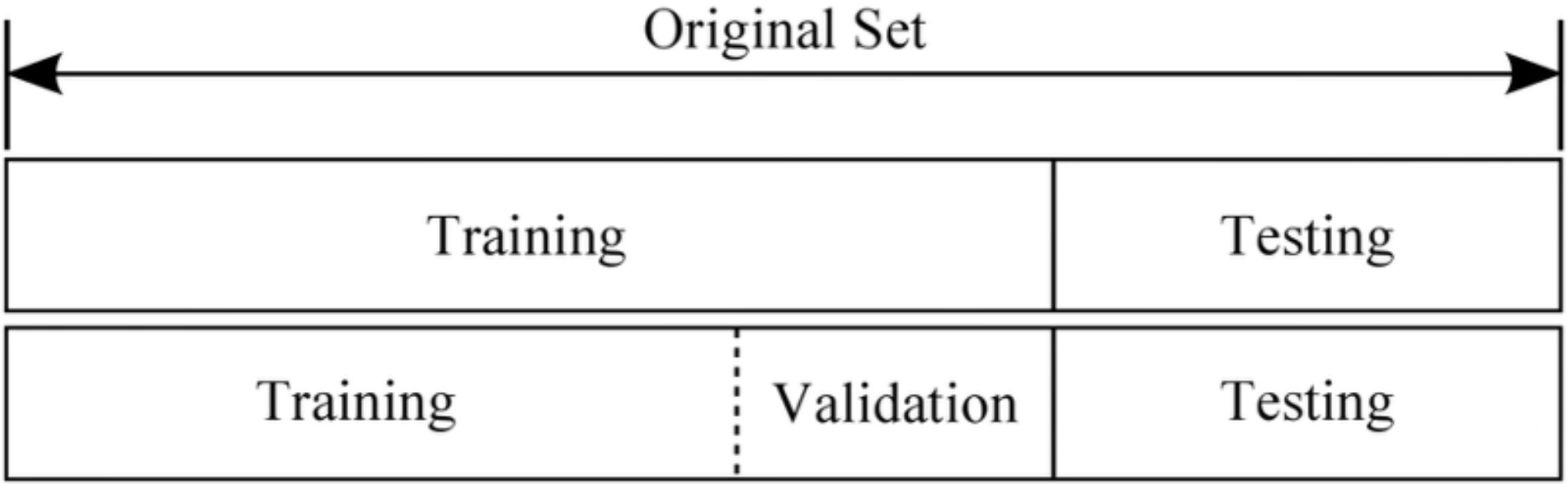
각 열(column)을 특징/속성(feature) 이라고 합니다.

각 행(row)을 예제(Example) 데이터라고 합니다.

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

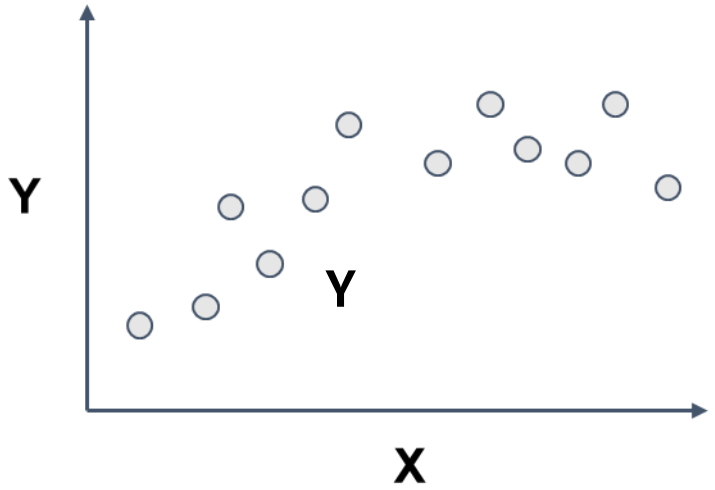
데이터 컬럼(column)중에 하나를 선택해서 레이블로 사용합니다.

데이터셋 분리

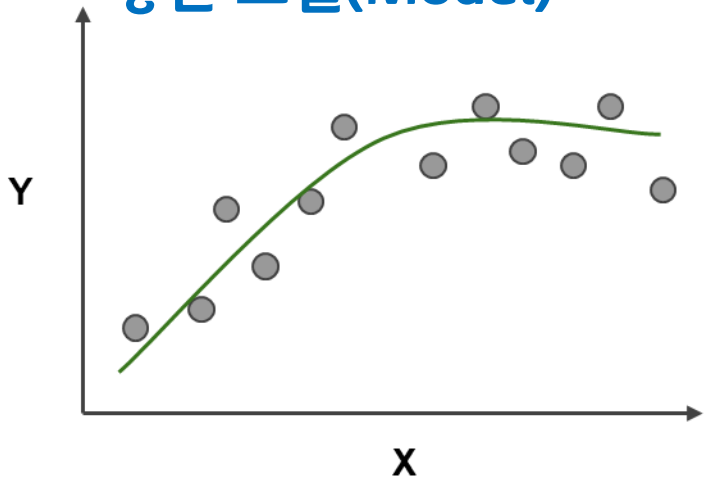

Original Set
Training
Testing
Training
Validation
Testing

모델 선택

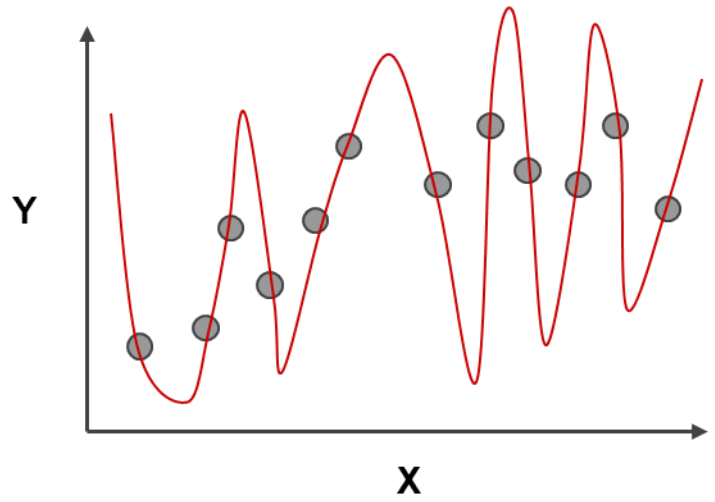
데이터



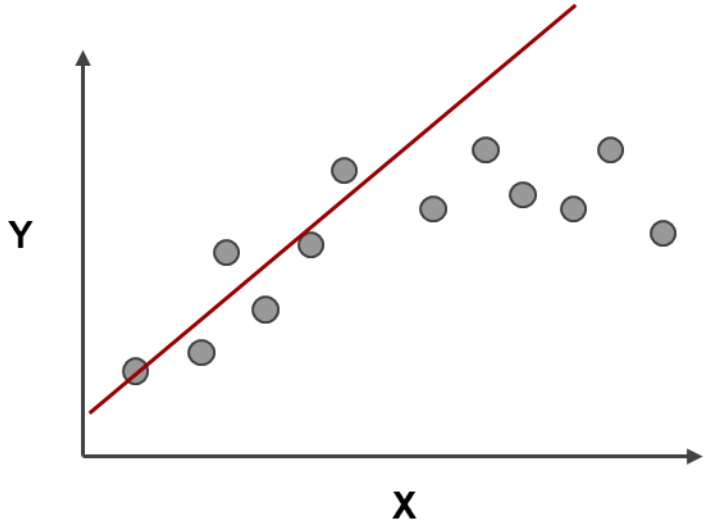
좋은 모델(Model)



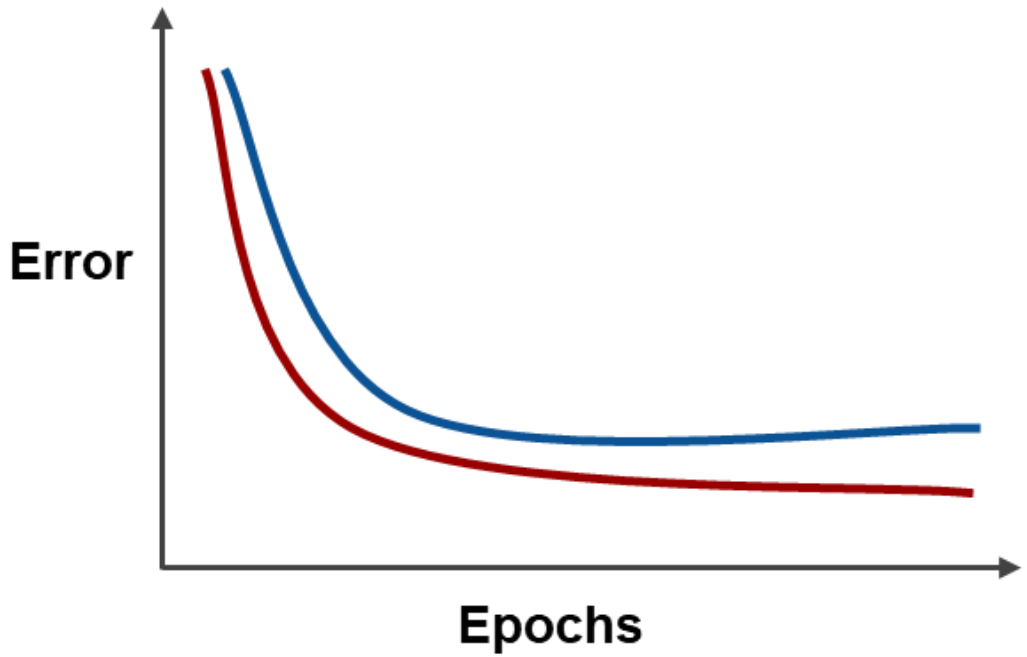
과적합(Overfitting)

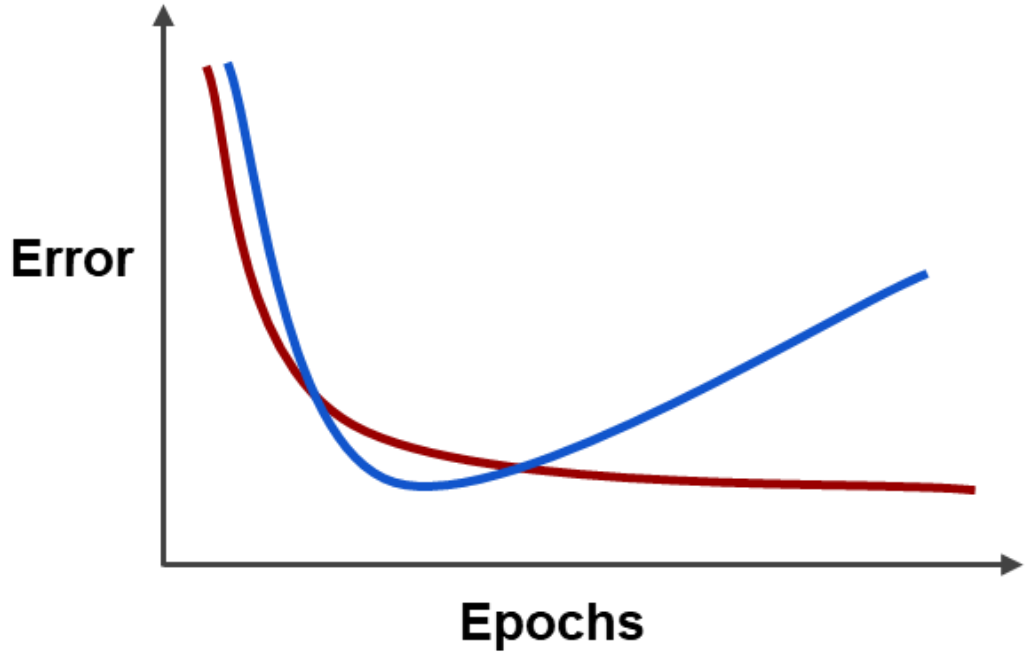


과소적합(Underfitting)



모델 성능 평가





training set test set

오차 행렬(Confusion Matrix)

모델의 대략적인 성능확인과 모델의 성능을 오차행렬을 기반으로 수치로 표현할 수 있습니다.

True Positive(TP) : True를 True로 예측 (정답)

True Negative(TN) : False를 False로 예측 (정답)

False Positive(FP) : False를 True로 예측 (오답)

False Negative(FN) : True를 False로 예측 (오답)

		예측값	
		0	1
실제값	0	TN	FP
	1	FN	TP

성능지표

성능지표는 학습이 끝난 후, 모델을 평가하는 용도로 사용됩니다.

■ 정밀도(Precision)

모델이 True라고 분류한 것 중에서 실제 True인 것의 비율
날씨 예측 모델이 맑다로 예측했는데, 실제 날씨가 맑았는지는 나타낸 지표입니다.

$$(Precision) = \frac{TP}{TP + FP}$$

■ 재현율(Recall)

실제 True인 것 중에서 모델이 True라고 예측한 것의 비율
실제 날씨가 맑은 날 중에서 모델이 맑다고 예측한 비율을 나타낸 지표입니다.

$$(Recall) = \frac{TP}{TP + FN}$$

■ 정확도(Accuracy)

$$(Accuracy) = \frac{TP + TN}{TP + FN + FP + TN}$$

가장 직관적으로 모델의 성능을 나타낼 수 있는 평가 지표
한달에 맑은 날이 28일이고 비가 오는 날이 2일인 경우,
비가 오는 것을 예측하는 성능은 매우 낮을 수 밖에 없으므로 이를 보완할 지표가 필요합니다.

■ F1 점수(F1-score)

정밀도와 재현율의 조화평균입니다.

$$(F1\text{-}score) = 2 \times \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}}$$

오차 행렬(Confusion Matrix)

```
y_true  = [1, 0, 1, 1, 0, 1]
y_pred =  [0, 0, 1, 1, 0, 0]
```

		예측값	
		0	1
실제값	0	2	0
	1	2	2

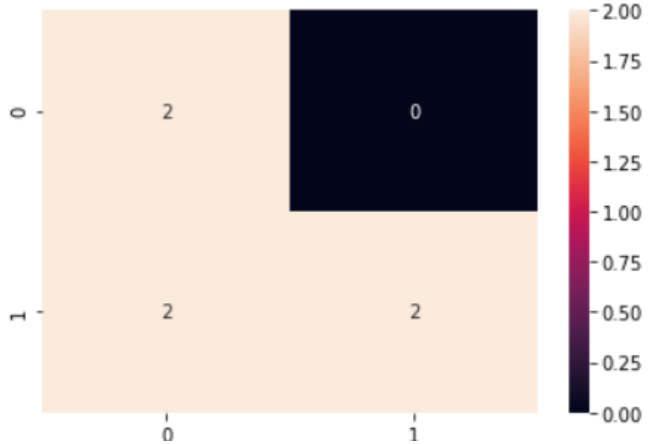
오차 행렬(Confusion Matrix)

```
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score

y_true = [1, 0, 1, 1, 0, 1]
y_pred = [0, 0, 1, 1, 0, 0]
cm = confusion_matrix(y_true, y_pred)
cm
[Out] array([[2, 0],
             [2, 2]], dtype=int64)

sns.heatmap(cm, annot=True)

precision_score(y_true, y_pred)
[Out] 1.0

recall_score(y_true, y_pred)
[Out] 0.5
```

		예측값	
		0	1
실제값	0	2	0
	1	2	2

정밀도: 2/2 =1

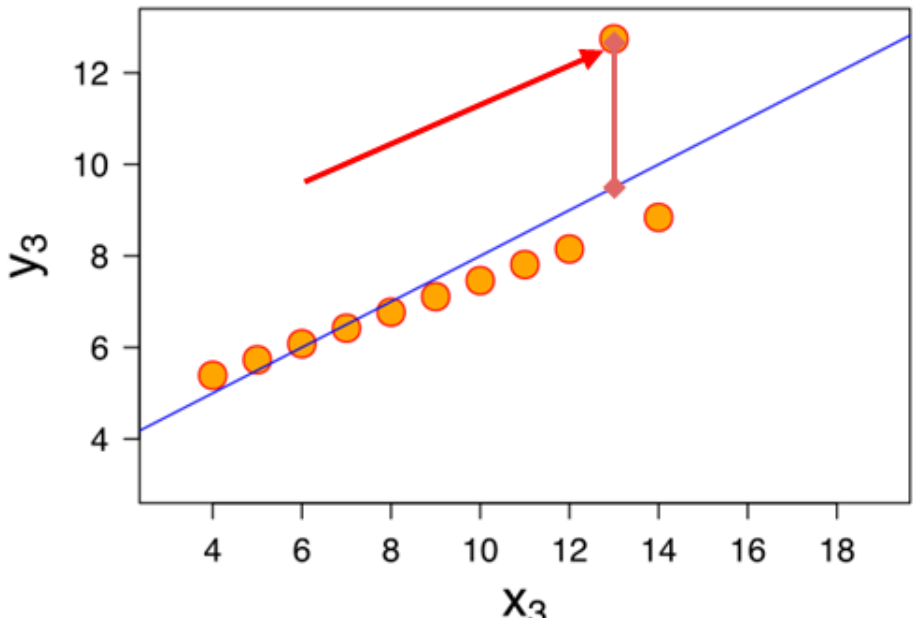
재현율 : 2/4 = 0.5



손실함수

Loss Function은 모델의 손실을 측정하는 함수로 학습을 진행하면서 지속적으로 측정이 됩니다.
예측 모델의 손실측정함수는 MSE, 분류 모델의 손실측정함수는 Cross Entropy가 사용됩니다.

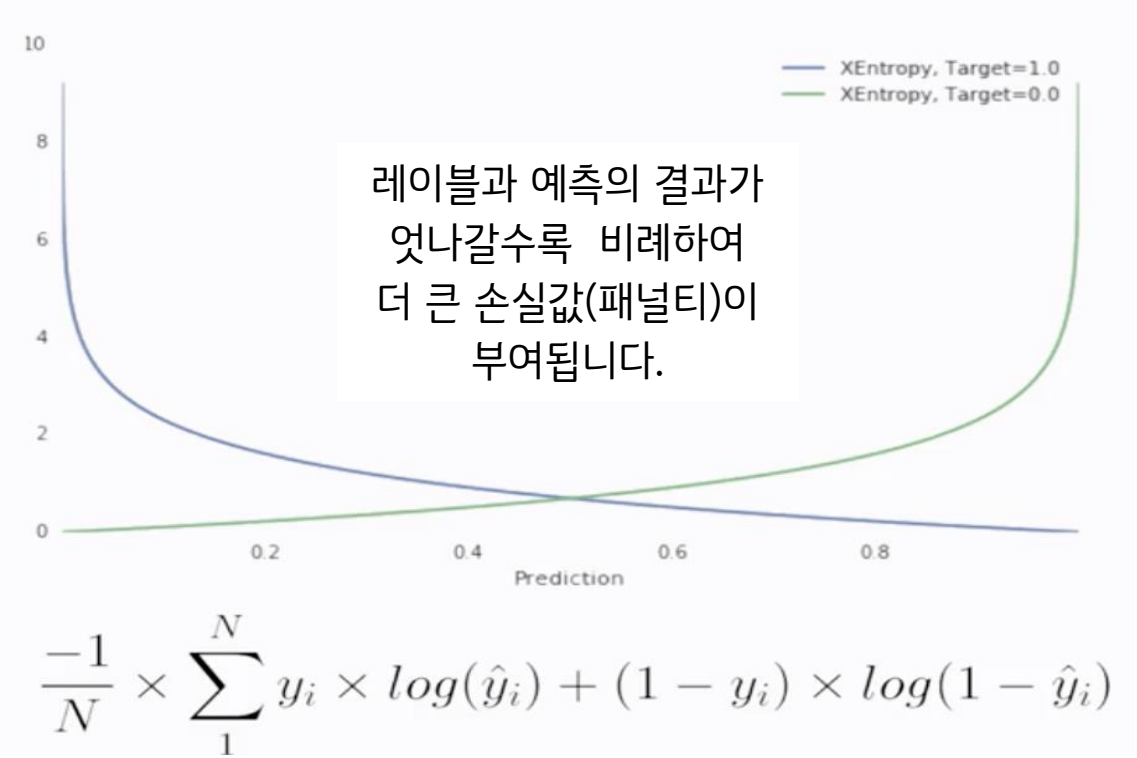
■ MSE(Mean Squared Error)



$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

레이블 값 (실제값) y_hat (모델이 예측한 값)

■ Cross Entropy

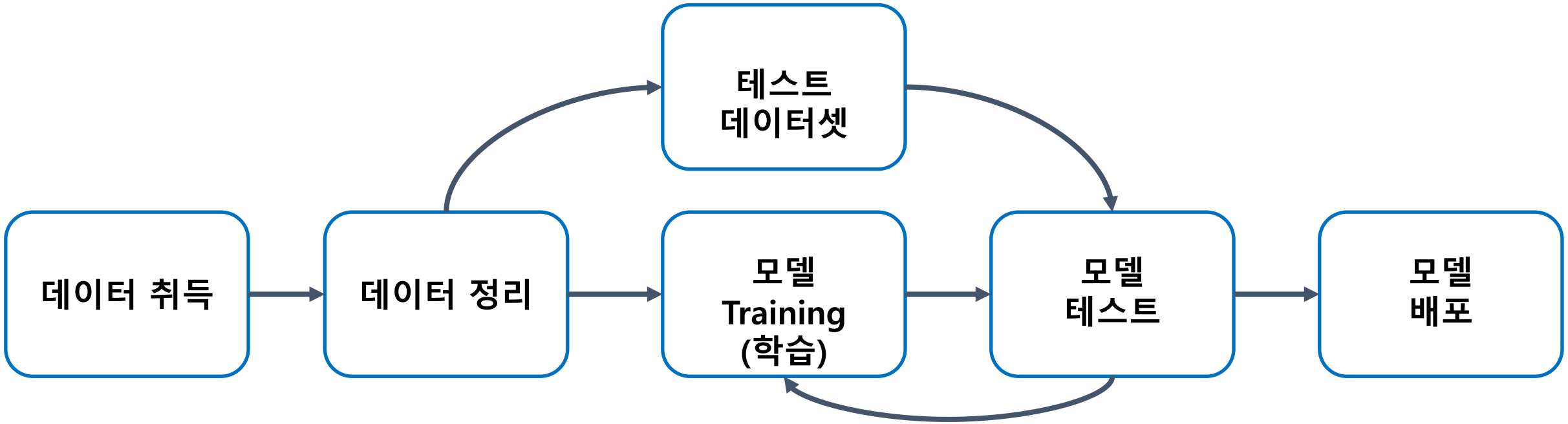


레이블과 예측의 결과가 엇나갈수록 비례하여 더 큰 손실값(패널티)이 부여됩니다.

$$\frac{-1}{N} \times \sum_{1}^{N} y_i \times log(\hat{y}_i) + (1 - y_i) \times log(1 - \hat{y}_i)$$

※ 크로스 엔트로피 참고자료
https://youtu.be/r3iRRQ2ViQM
https://youtu.be/Jt5BS71uVfI

머신러닝 프로세스



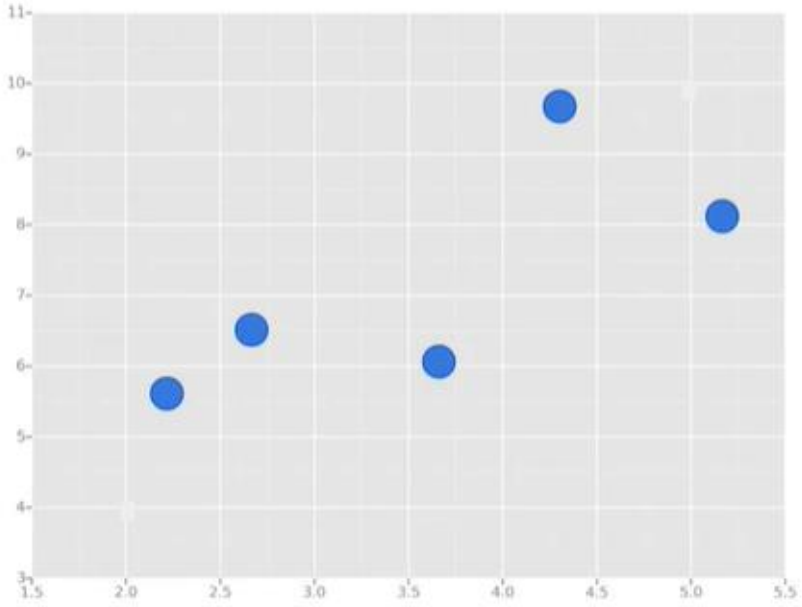
사이킷런(Scikit-learn)

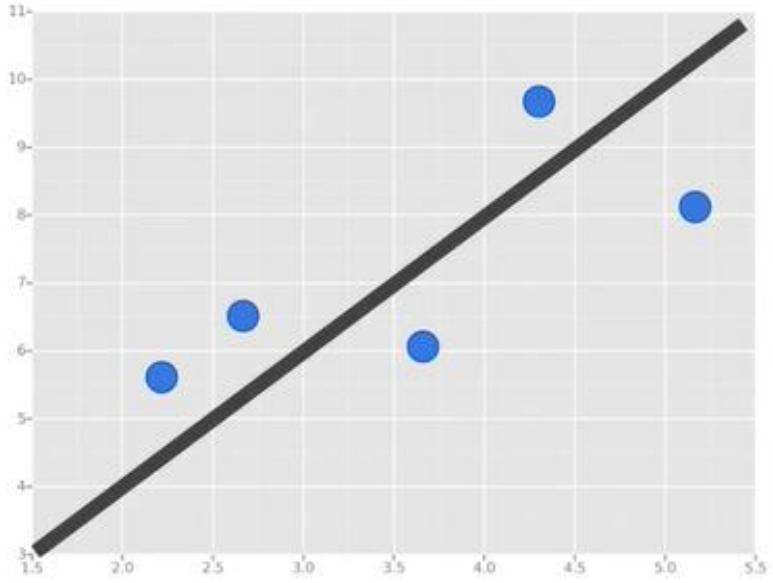
가장 인기있는 머신러닝 패키지이며, 많은 머신러닝 알고리즘이 내장되어 있습니다.

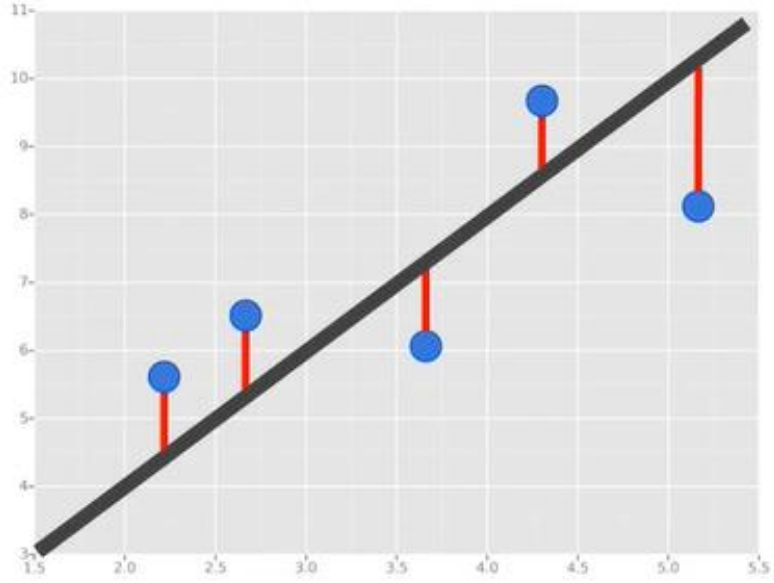
```
from sklearn.family import Model
```

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
model.predict(X_test)
```

선형 회귀(Linear Regression)







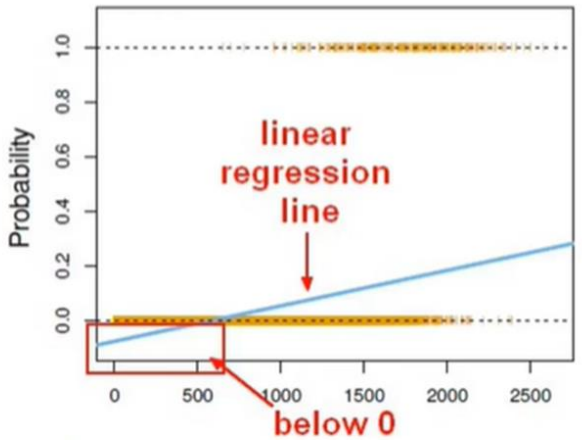
선형 회귀(Linear Regression)

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression


X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

로지스틱 회귀(Logistic Regression)

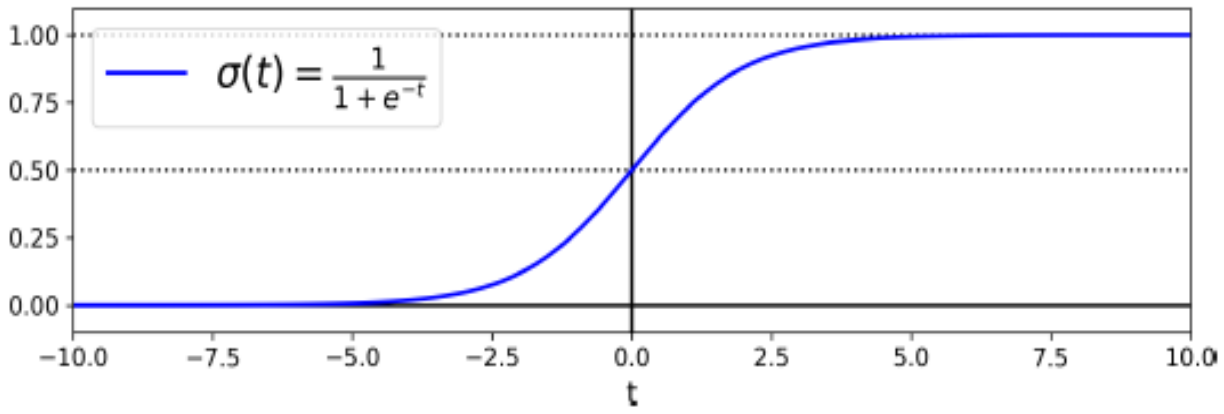
- 이진 분류 규칙은 0과 1의 두 클래스를 갖는 것으로, 일반 선형 회귀 모델을 이진분류에 사용할 수 없습니다.
- 대신 선형 회귀를 로지스틱 회귀 곡선으로 변환 할 수 있으며, 로지스틱 회귀 곡선은 0과 1 사이에서만 이동할 수 있으므로 분류에 사용할 수 있습니다.
- 로지스틱 회귀는 선형 회귀처럼 바로 결과를 출력하지 않고 로지스틱(logistic)을 출력합니다.
- 로지스틱 회귀는 샘플이 특정 데이터에 속할 확률을 추정(이진분류)하는 데 사용됩니다.
- 추정 확률이 50%가 넘으면 모델은 그 샘플이 해당 클래스에 속한다고 예측합니다.

일반 선형 모델



로지스틱 함수

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



로지스틱 확률모델

$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma\left(\mathbf{x}^T\boldsymbol{\theta}\right)$$

$$\hat{y} = \begin{cases} 0 \text{ if } \hat{p} < 0.5 \\ 1 \text{ if } \hat{p} \geq 0.5 \end{cases}$$

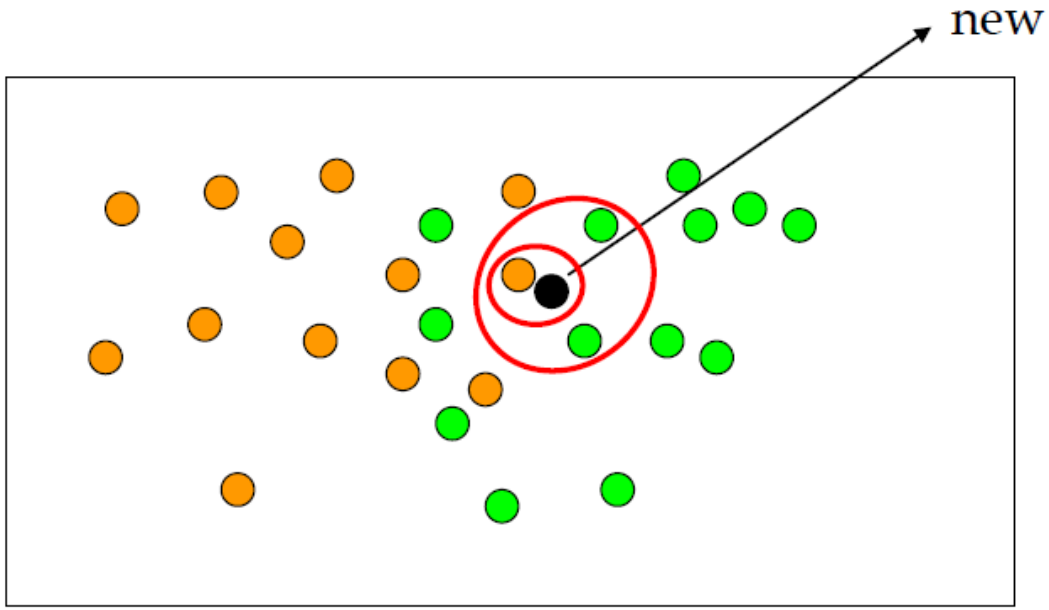
로지스틱 회귀(Logistic Regression)

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# 데이터셋: https://www.kaggle.com/c/titanic
train = pd.read_csv(aidu_framework.config.data_dir + '/train.csv')



X_train, X_test, y_train, y_test = train_test_split(
                train.drop('Survived',axis=1),
                train['Survived'], test_size=0.30, random_state=42)
model =  LogisticRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
print(classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
0	0.79	0.91	0.85	163
1	0.82	0.62	0.71	104
accuracy			0.80	267
macro avg	0.81	0.77	0.78	267
weighted avg	0.80	0.80	0.80	267

K-최근접 이웃(K-Nearest Neighbor)

- KNN은 새로운 데이터가 주어졌을 때 기존 데이터 가운데 가장 가까운 k개 이웃의 정보로 새로운 데이터를 예측하는 방법론입니다. 아래 그림처럼 검은색 점의 범주 정보는 주변 이웃들을 가지고 추론해낼 수 있습니다.
- 만약 k가 1이라면 오렌지색, k가 3이라면 녹색으로 분류(classification)하는 것입니다.
- 만약, 회귀(regression) 문제라면 이웃들 종속변수(y)의 평균이 예측값이 됩니다.



출처 : https://ratsgo.github.io/machine%20learning/2017/04/17/KNN/

K-최근접 이웃(K-Nearest Neighbor)

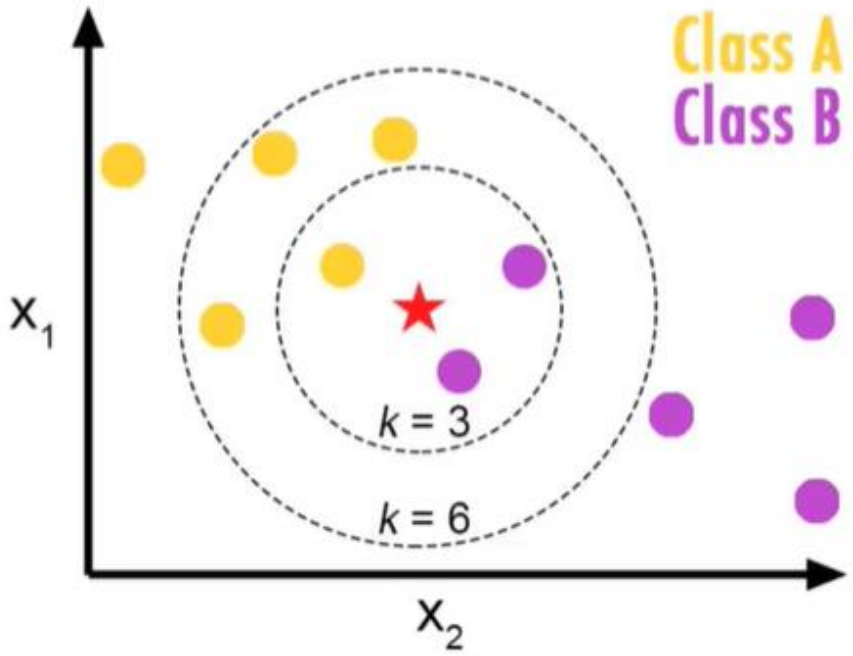
- 알고리즘이 간단하며 큰 데이터셋과 고차원 데이터에 적합하지 않은 단점이 있습니다.

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier(n_neighbors=3)
```

```
knn.fit(X_train,y_train)
```

```
pred = knn.predict(X_test)
```

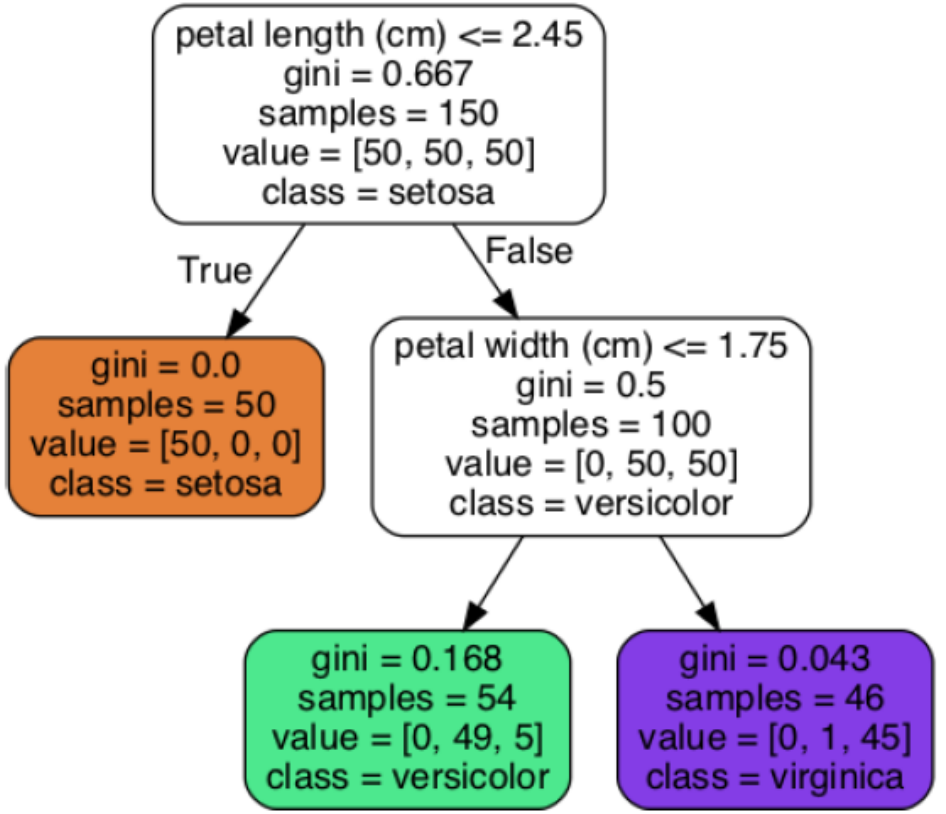


결정 트리(Decision Tree)

- 분류와 회귀 작업 이 가능한 다재다능한 머신러닝 알고리즘입니다.
- 복잡한 데이터셋도 학습할 수 있으며, 강력한 머신러닝 알고리즘인 랜덤 포레스트의 기본 구성 요소입니다.

■ 붓꽃(Iris)

■ 붓꽃 분류 결정트리

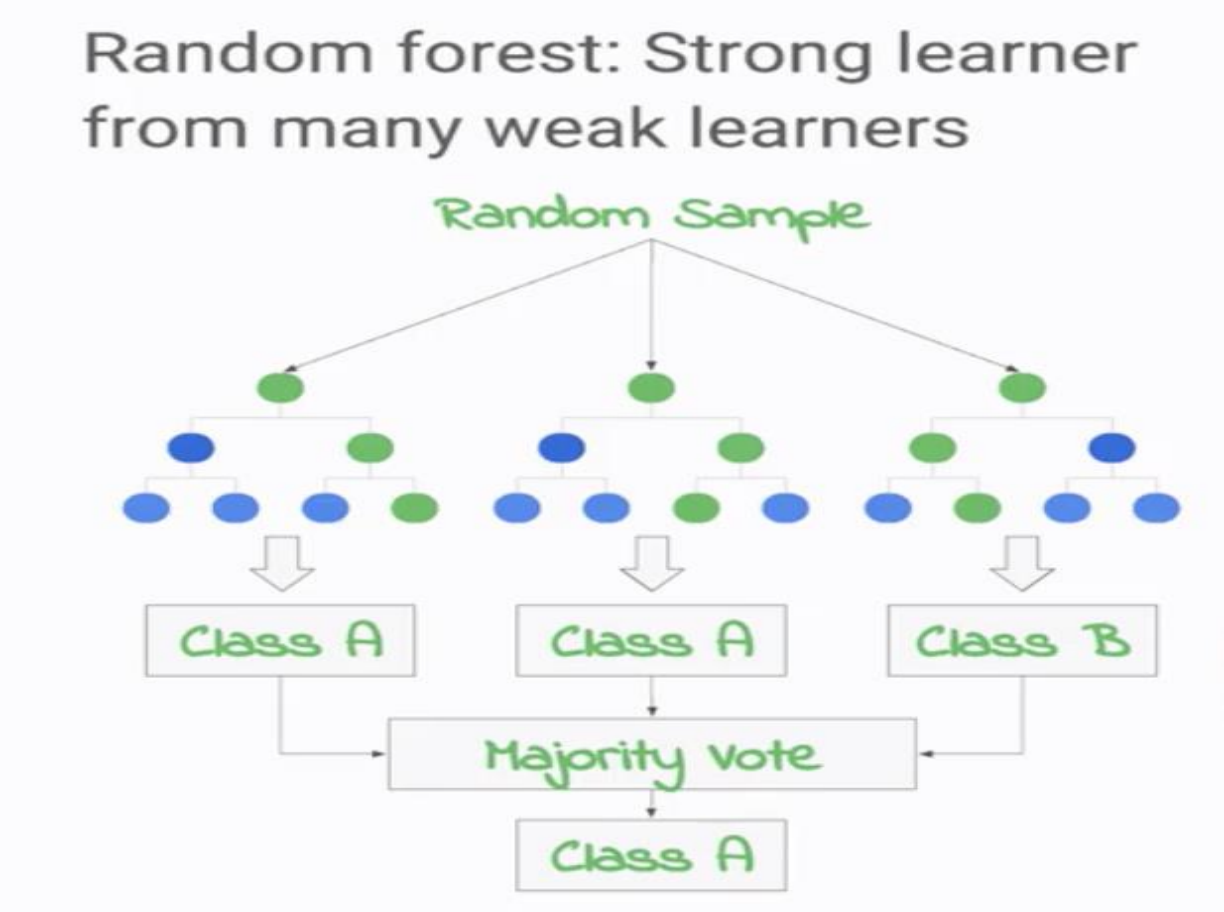


결정 트리(Decision Tree)

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier

iris = load_iris()
X = iris.data[:, 2:] # petal length and width
y = iris.target

dtree_clf = DecisionTreeClassifier(max_depth=2)
dtree_clf.fit(X, y)

dtree_clf.predict_proba([[5, 1.5]])
[Out] array([[0. , 0.90740741, 0.09259259]])

dtree_clf.predict([[5, 1.5]])
[Out] array([1])
```

랜덤 포레스트(Random Forest)

- 일련의 예측기(분류, 회귀모델)로 부터 예측을 수집하면 가장 좋은 모델 하나보다 더 좋은 예측을 얻을 수 있습니다.
- 일련의 예측기를 앙상블이라고 부르며, 결정 트리의 앙상블을 랜덤 포레스트라고 합니다.
- 훈련 세트로 부터 무작위로 각기 다른 서브셋을 만들어 일련의 결정 트리 분류기를 훈련시킬 수 있습니다.



랜덤 포레스트(Random Forest)

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

iris = datasets.load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
X_train, X_test, y_train, y_test = train_test_split(
      df[iris.feature_names], iris.target, test_size=0.25,
      stratify=iris.target, random_state=42)
rf = RandomForestClassifier(n_estimators=50,
                            max_depth=20,
                            random_state=42)
rf.fit(X_train, y_train)
predicted = rf.predict(X_test)
accuracy = accuracy_score(y_test, predicted)
```

shift+tab : 함수 설명 보기

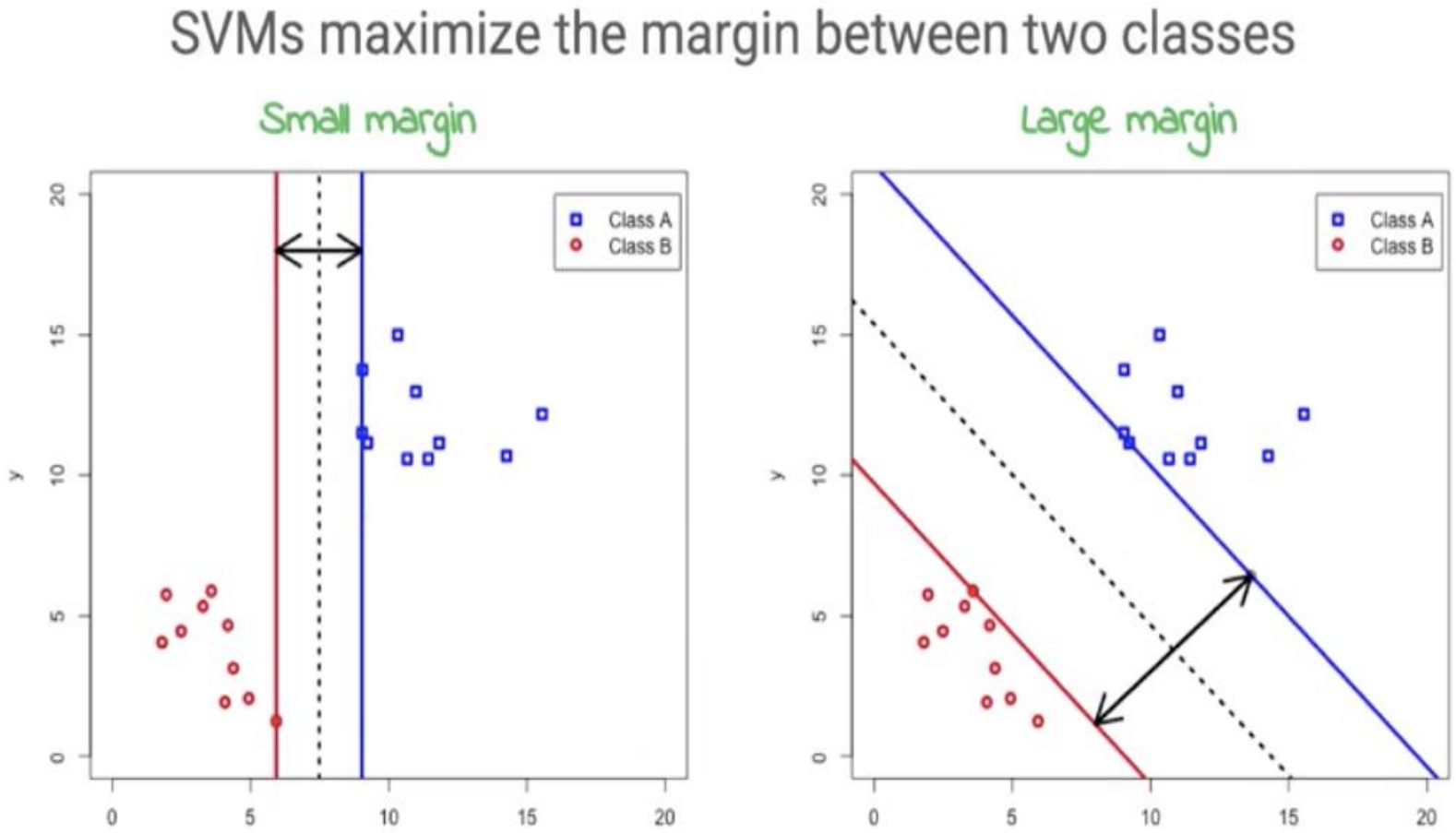
```
Init signature:
RandomForestClassifier(
    n_estimators=100,
    *,
    criterion='gini',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features='auto',
```

랜덤 포레스트(Random Forest)

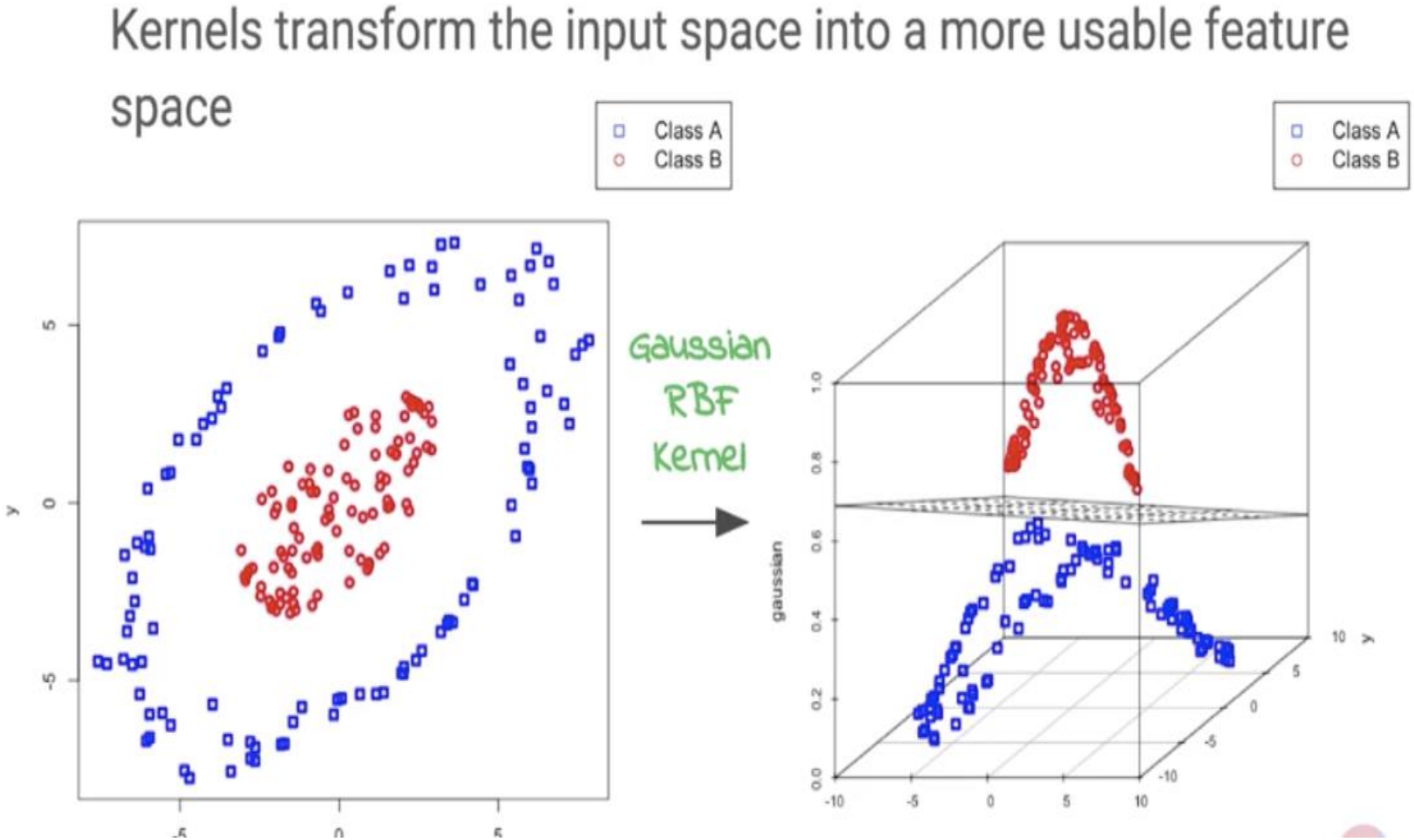
- n_estimators : 모델에서 사용할 트리 갯수(학습시 생성할 트리 갯수)
- criterion : 분할 품질을 측정하는 기능 (default : gini)
- max_depth : 트리의 최대 깊이
- min_samples_split : 내부 노드를 분할하는데 필요한 최소 샘플 수 (default : 2)
- min_samples_leaf : 리프 노드에 있어야 할 최소 샘플 수 (default : 1)
- min_weight_fraction_leaf : min_sample_leaf와 같지만 가중치가 부여된 샘플 수에서의 비율
- max_features : 각 노드에서 분할에 사용할 특징의 최대 수
- max_leaf_nodes : 리프 노드의 최대수
- min_impurity_decrease : 최소 불순도
- min_impurity_split : 나무 성장을 멈추기 위한 임계치
- bootstrap : 부트스트랩(중복허용 샘플링) 사용 여부
- oob_score : 일반화 정확도를 줄이기 위해 밖의 샘플 사용 여부
- n_jobs :적합성과 예측성을 위해 병렬로 실행할 작업 수
- random_state : 난수 seed 설정
- verbose : 실행 과정 출력 여부

서포트 벡터 머신(SVM)

- 강력하고 선형이나 비선형 분류, 회귀, 이상치 탐색에도 사용할 수 있는 다목적 머신러닝 모델입니다.
- SVM은 특히 복잡한 분류 모델에 잘 들어 맞으며 작거나 중간 크기의 데이터셋에 적합합니다.



서포트 벡터 머신(SVM)


Kernels transform the input space into a more usable feature space
Class A
Class B
Gaussian
RBF
Kernel
Class A
Class B
y
gaussian

실습

https://github.com/kgpark88/ems/blob/master/ml_summary.ipynb

감사합니다.