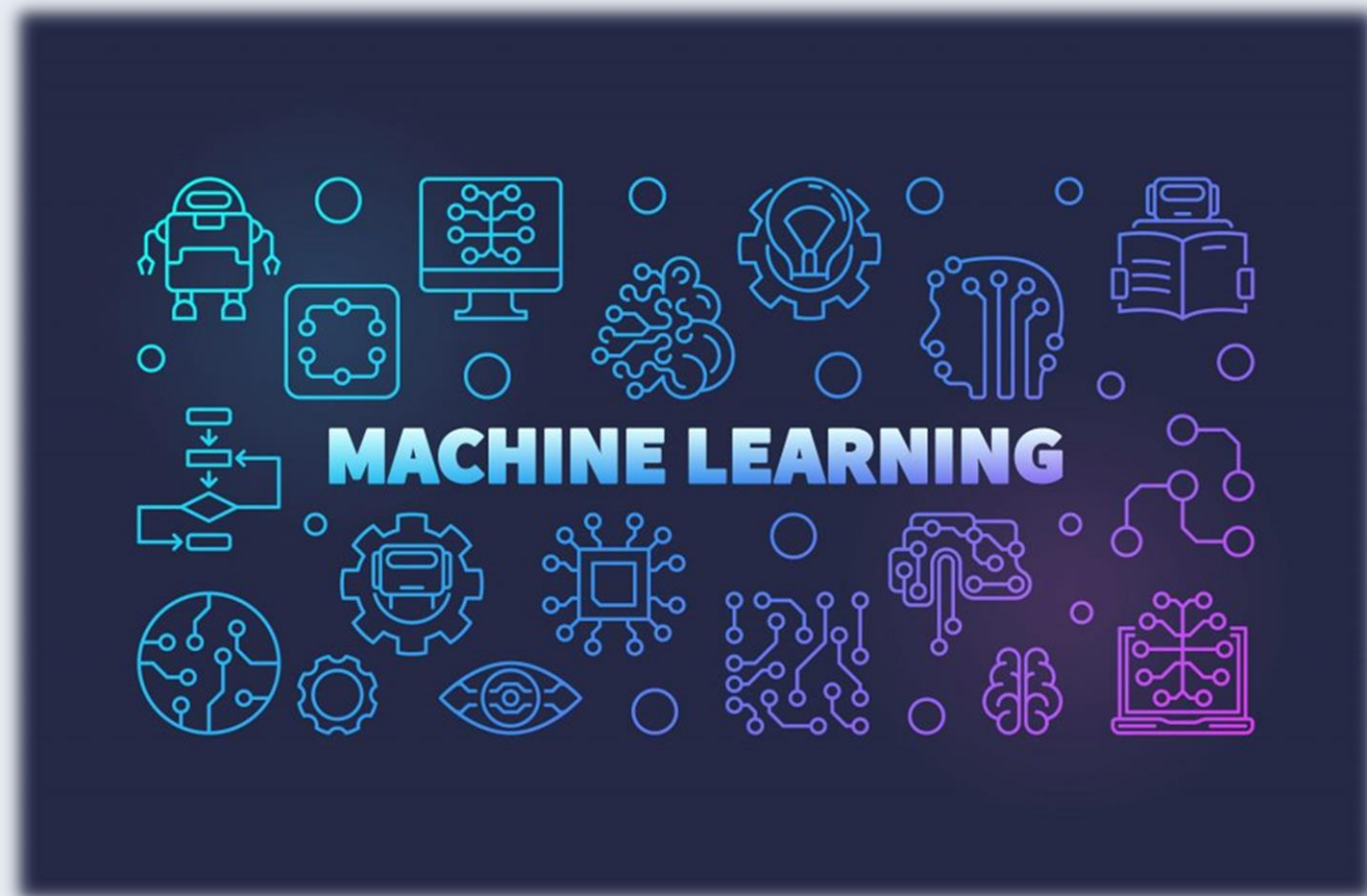


머신러닝 주요 모델



AI(Artificial Intelligent)

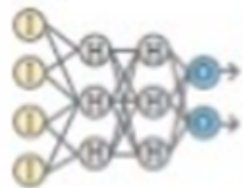
- 머신러닝은 알고리즘 위주로 발전하여 Linear Regression, Logistic Regression, Decision Tree 등 수많은 알고리즘들이 고안 되었으며, 더 높은 성능을 낼 수 있는 알고리즘이 연구 주제였습니다.
- 딥러닝에서는 단일 알고리즘에 대한 연구보다, 어떤 구조의 모델을 만들어야 더 학습이 잘 될 것인에 관심을 가지고, DNN, CNN, RNN, GAN 등 복잡한 구조의 신경망이 고안 되었습니다.
- 그리고, 모델의 크기를 증가시키면 더 좋은 성능에 나오는 것을 발견하여, 계속 모델들이 대형화되는 추세입니다.

인공지능 사람의 지적능력(추론, 인지)을 구현하고 모방하는 모든 기술

머신러닝 명시적인 프로그래밍 없이 학습하는 기술



선형회귀
로지스틱회귀
K-최근접 이웃
결정트리
랜덤포레스트
서포트 벡터 머신
클러스터링
차원축소



딥러닝

인공신경망 이용해 데이터에서 패턴을 찾아내는 기술

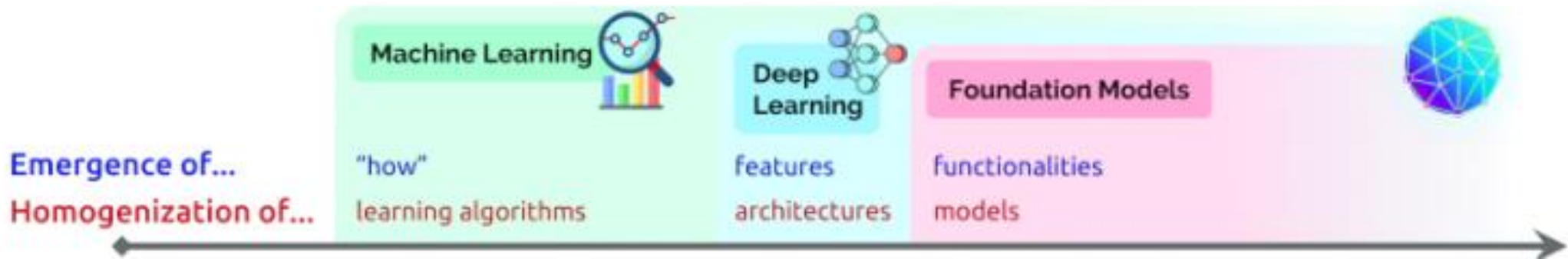
심층신경망(DNN)
합성곱신경망(CNN)
순환 신경망(RNN)
생성적 적대 신경망(GAN)
강화학습(RL)

파운데이션 모델

BERT
GPT

AI 발전방향

- 초기 머신러닝 시절에는 어떤 알고리즘을 사용할지, 어떤 특성(feature)을 사용할지를 사용자가 일일이 지정해주어야 했지만, 이제는 데이터로부터 잠재특성(latent feature)을 추출하여 학습할 수 있습니다.
- AI는 Emergence와 Homogenization을 증가시키는 방향으로 발전하고 있습니다.
- Emergence(창발성)은 AI모델이 스스로 어떠한 문제를 해결하기 위한 지식을 도출하는 능력을 의미합니다.
- Homogenization(균질화)는 AI 모델이 점차 일반화된 지식을 도출해 낼 수 있게 됨에 따라, 하나의 뛰어난 모델이 적용될 수 있는 범위가 점차 확대되고, 더욱 보편적이고 범용적으로 활용되는 현상을 의미합니다.



머신러닝

머신러닝은 컴퓨터 알고리즘이 데이터를 학습하여 입력과 출력간의 관계를 찾는 과정입니다.
학습할 때 데이터에 정답 레이블이 있는지 여부에 따라 지도학습과 비지도학습으로 구분을 합니다.

■ 지도학습(supervised learning)

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

지도 학습은 정답이 있는 데이터를 활용해 데이터를 학습시키는 것입니다.

입력 값(X data)이 주어지면 입력값에 대한 Label(Y data)를 주어 학습시키며, 분류모델과 회귀모델이 있습니다.

■ 모델

- 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 나뉘질 수 있는 문제에 사용합니다.
- 예) 스팸 메일 분류, 품종 분류

Classification



■ 회귀 모델(Regression)

- 레이블의 값들이 연속적인 문제에 사용합니다.
- 예) 날씨 예측, 주가 예측

Regression



지도학습 데이터셋 구조

각 열(column)을 특징/속성(feature) 이라고 합니다.
데이터 컬럼(column)중에 하나를 선택해서 레이블로 사용합니다.

	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

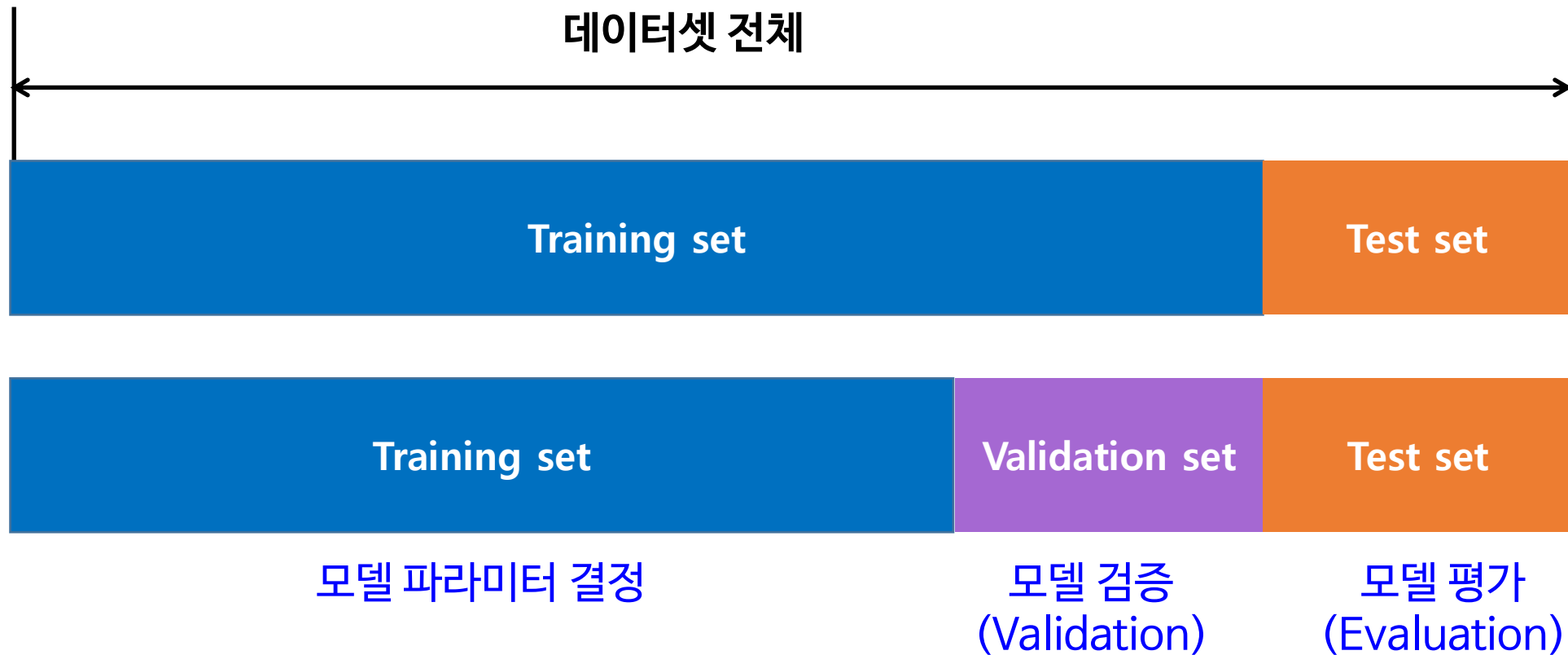
각 행(row)을
예제 (Example)
데이터라고
합니다.

회귀 모델
팁 액수(숫자)를 예측

분류 모델
손님의 성별(카테고리)을 예측

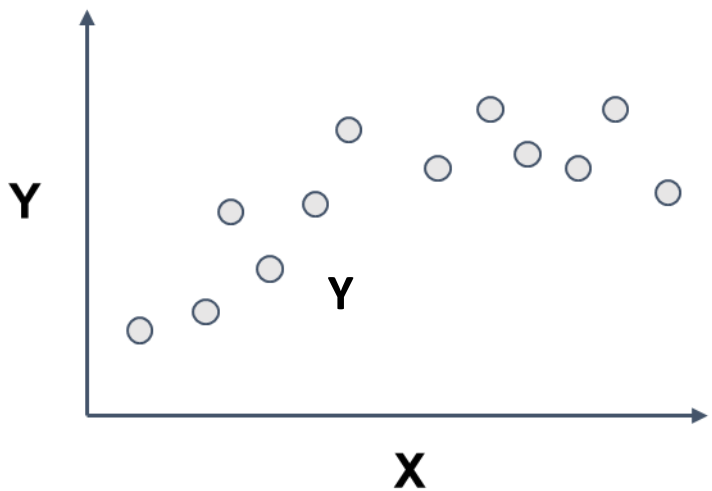


데이터셋 분리

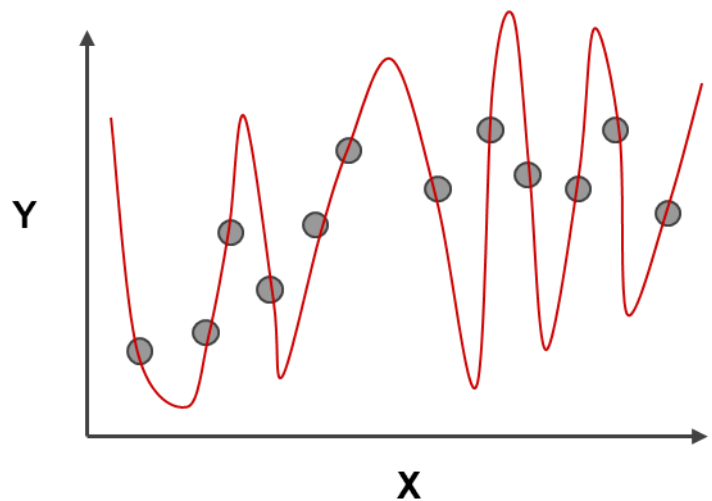


모델 선택

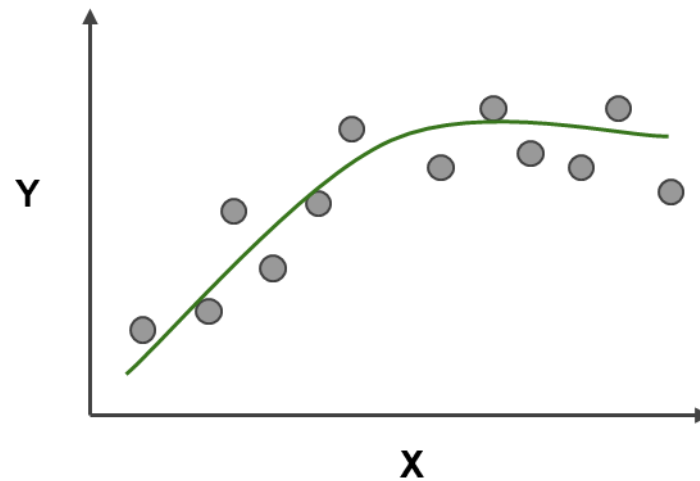
데이터



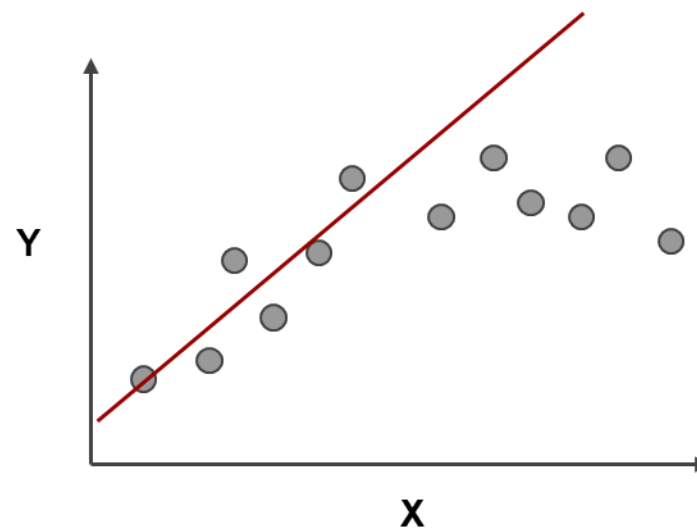
과적합(Overfitting)



좋은 모델(Model)

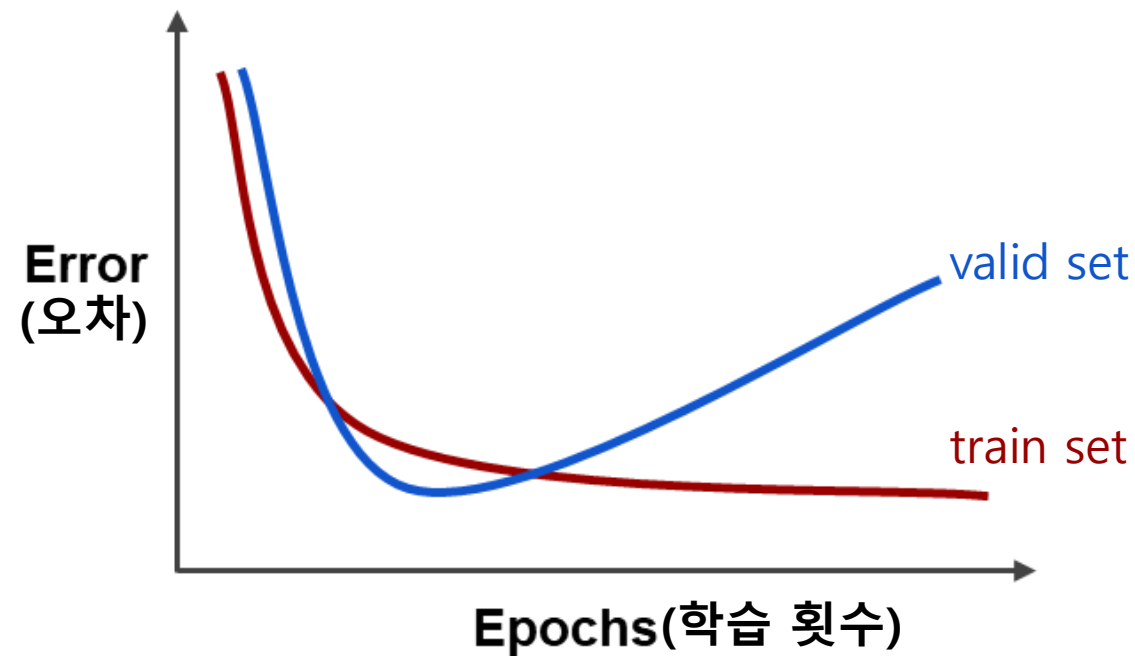
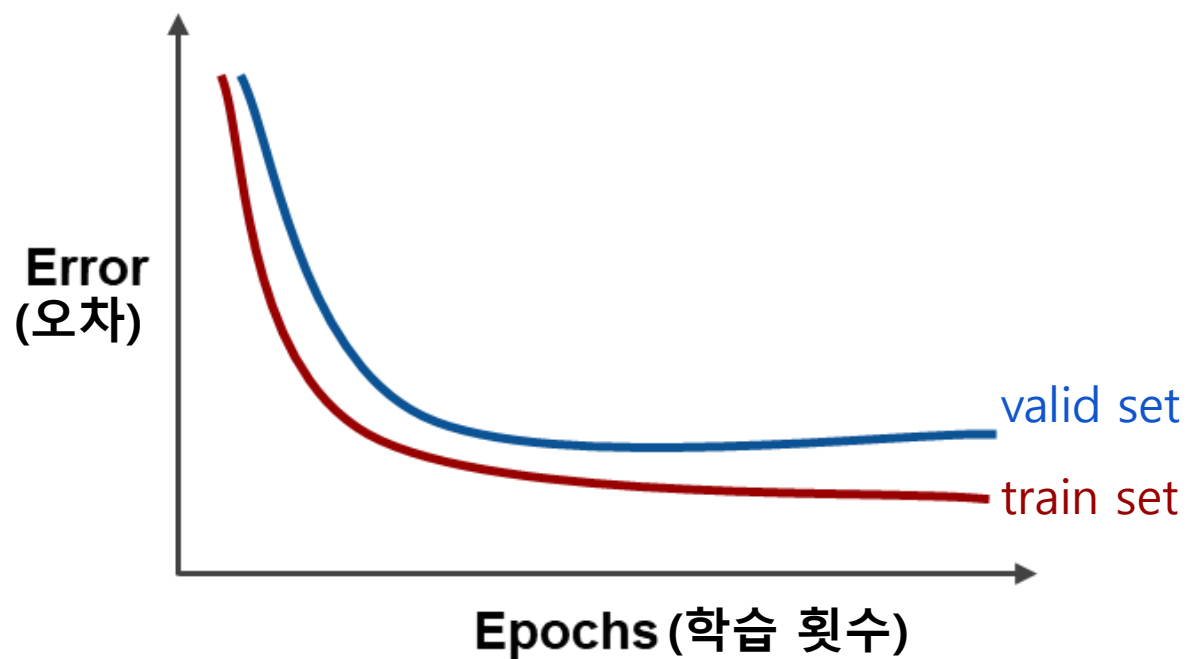


과소적합(Underfitting)



모델 성능 측정

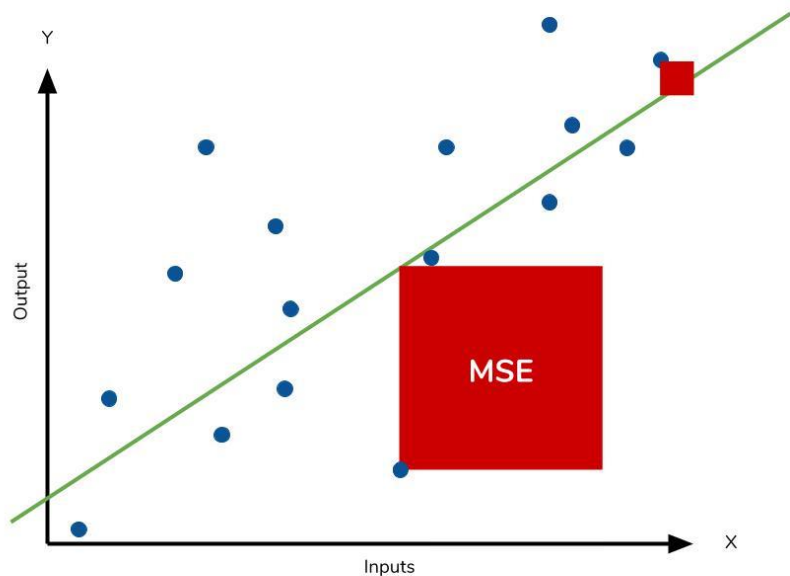
모델학습을 진행하면서 손실(Loss, Error, 오차)을 지속적으로 측정합니다.



회귀모델 손실함수(Loss Function)

회귀모델(Regression)에서는 주로 평균제곱오차(MSE)를 손실함수로 사용합니다.

■ MSE(Mean Squared Error)



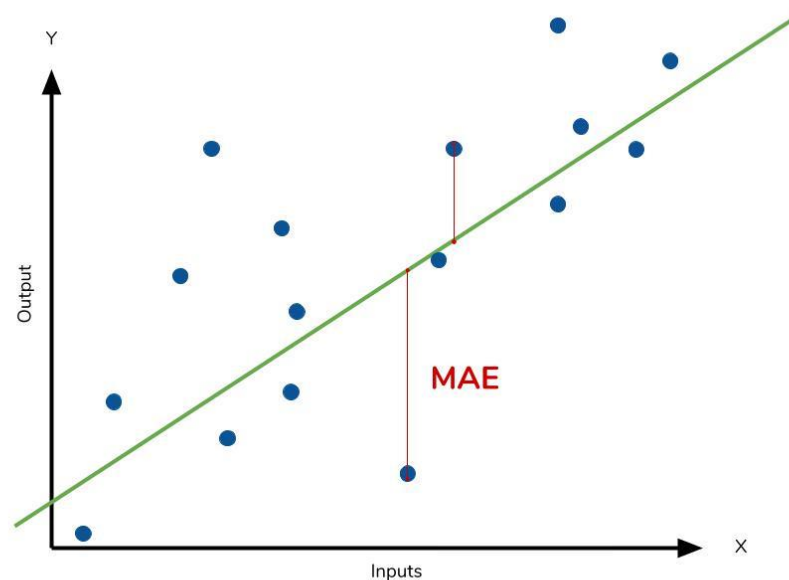
$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

레이블 값
(실제값)

y_hat

(모델이 예측한 값)

■ MAE(Mean Absolute Error)



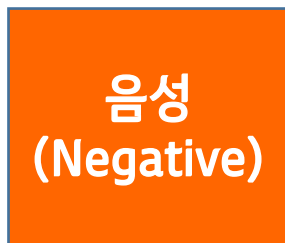
$$\frac{1}{n} \sum_{i=1}^n (|y_i - \hat{y}_i|)$$

분류모델 손실함수(Loss Function)

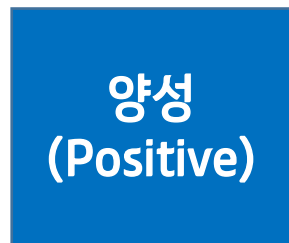
이진분류는 Binary Cross Entropy 를 다중분류는 Categorical Cross Entropy 손실함수를 사용합니다.

■ 이진분류(Binary Classification)







0



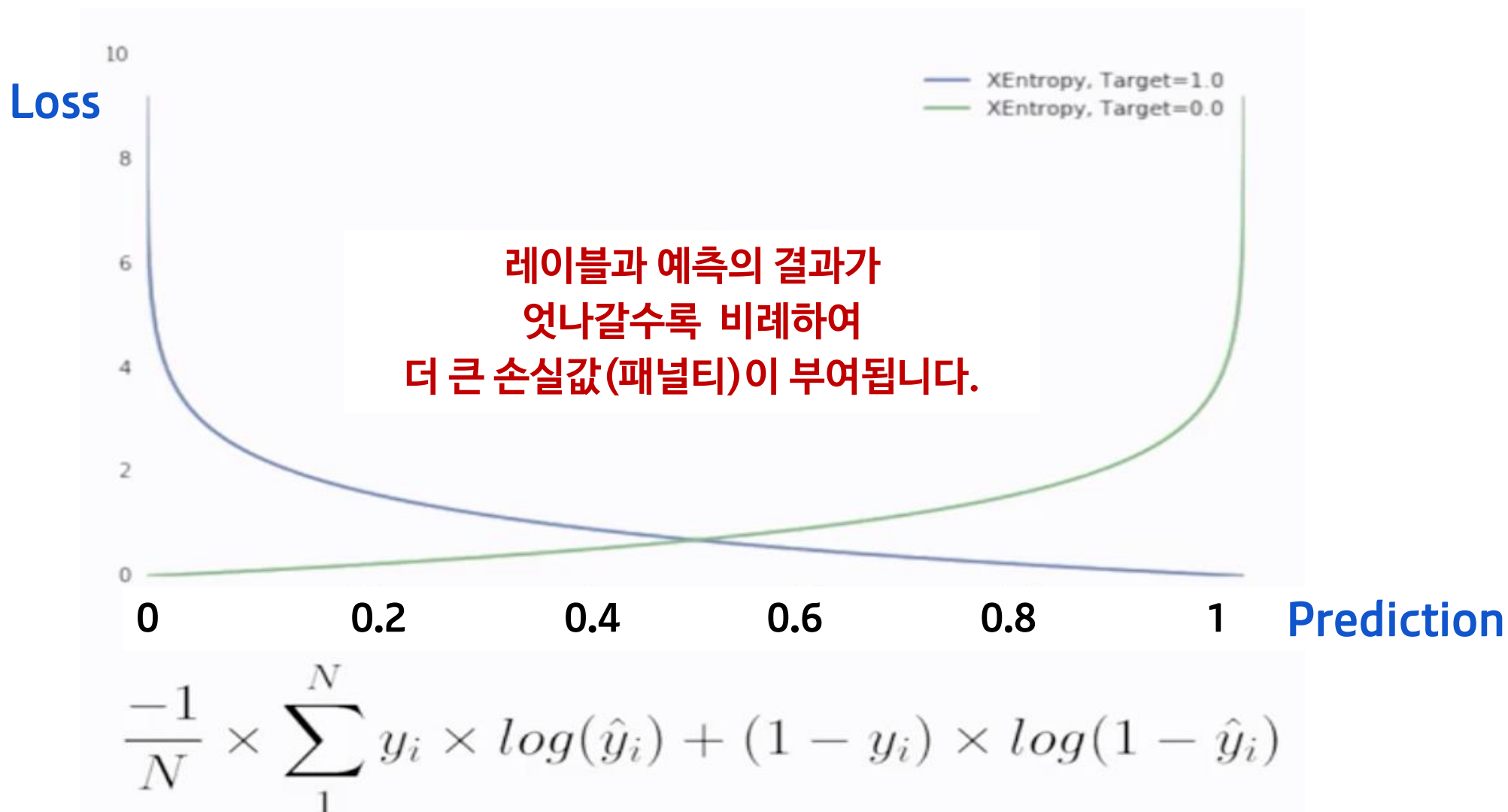
1



■ 다중분류(Multi-Class Classification)

C = 3	Samples		
  			
	Labels (t)		
	[0 0 1]	[1 0 0]	[0 1 0]

분류모델 손실함수(Loss Function)



분류모델 성능측정 - 오차행렬 (Confusion Matrix)

분류모델 성능평가에 사용하며 대략적인 성능확인과 모델의 성능을 오차행렬을 기반으로 수치로 표현할 수 있습니다.

Positive = True = 양성 = 1

Negative = False = 음성 = 0

분류결과	실제정답	
	Positive	Negative
Positive	True Positive	False Positive
Negative	False Negative	True Negative

■ 예측이 맞은 경우

TP(True Positive) : Positive로 예측했고, 실제값이 Positive

TN(True Negative) : Negative로 예측했고 실제값이 Negative

■ 예측이 틀린 경우

FP(False Positive) : Positive로 예측했지만 실제값은 Negative

FN(False Negative) : Negative로 예측했지만 실제값은 Positive

분류모델 성능측정 - 평가지표

■ 정확도(Accuracy)

가장 직관적으로 모델의 성능을 나타낼 수 있는 기본적인 평가 지표

불균형한 데이터셋(예 : 양성 99개, 음성1개)에서는 제대로 평가가 안 되는 약점이 있습니다.

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

■ 정밀도(Precision)

예측을 Positive로 한 대상중에 예측값과 실제값이 Positive로 일치한 데이터의 비율

Positive 예측 성능을 더욱 정밀하게 측정하기 위한 평가 지표로 **양성 예측도**라고도 불립니다.

정밀도를 높이기 위해서는 FP(Negative를 Positive로 예측하는 경우)를 낮추어야 합니다.

정밀도가 중요한 경우 : Negative를 Positive로 예측하면 큰 문제/비용이 발생시(예 : 스팸 메일 분류 모델, 현장 출동 분류 모델)

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

■ 재현율(Recall, 회수율)

실제값이 Positive인 대상중에 예측값과 실제값이 Positive로 일치한 데이터의 비율

민감도(Sensitivity) 또는 TPR(True Positive Rate)라고도 불린다.

재현율을 높이기 위해서는 FN(Positive를 Negative로 예측하는 경우)를 낮추어야 합니다.

재현율이 중요한 경우 : Positive를 Negative로 예측하면 큰 문제가 발생하는 경우(예 : 암 판단 모델, 금융 사기 적발 모델)

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

■ F1 점수(F1 score)

정밀도와 재현율은 trade-off 관계가 있습니다.

F1 score는 정밀도와 재현율의 조화평균입니다.

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

분류모델 성능측정 - 평가지표

The screenshot shows a web browser window with a ChatGPT chat interface. The browser's address bar shows the URL: `chat.openai.com/c/76890a50-2c3e-4b6a-b4a4-065f520616ba`. The chat window has a sidebar on the left with a 'New chat' button and a list of previous chats, including one titled '에너지 데이터 분석'. The main chat area shows a user's question in a red-bordered box: '머신러닝 분류모델의 성능측정 지표인 정밀도와 재현율의 차이점에 대해 알려줘.' Below this, the AI's response is displayed, starting with the OpenAI logo and explaining that Precision and Recall are important metrics for evaluating classification models. The response is structured with numbered points and bullet points. At the bottom of the chat area, there is a 'Send a message' input field and a 'Regenerate' button. The footer of the page includes a disclaimer: 'Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT August 3 Version'.

에너지 데이터 분석

chat.openai.com/c/76890a50-2c3e-4b6a-b4a4-065f520616ba

+ New chat

Today

에너지 데이터 분석

머신러닝 분류모델의 성능측정 지표인 정밀도와 재현율의 차이점에 대해 알려줘.

머신러닝 분류모델의 성능을 평가하는 지표로서 정밀도(Precision)와 재현율(Recall)은 중요한 개념입니다. 하지만 이 두 지표는 서로 다른 측면을 측정합니다. 정밀도와 재현율의 주요 차이점을 살펴보겠습니다:

- 정밀도 (Precision):**
 - 정밀도는 모델이 예측한 양성 클래스 중에서 실제로 양성인 경우의 비율을 측정합니다.
 - True Positives를 True Positives와 False Positives의 합으로 나눈 값으로 계산됩니다.
 - 공식: $\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$

정밀도의 장점:

- 거짓 양성(Type I 오류)의 비용이 높을 때나 거짓 양성을 최소화하고자 할 때 유용합니다.
- 양성 예측의 정확도에 대한 정보를 제공합니다.

정밀도의 단점:

- 거짓 음성(실제 양성인데 예측이 음성인 경우)을 고려하지 않아 모델 성능을 완전히 이해하는 데에 부족할 수 있습니다.

- 재현율 (Recall):**
 - 재현율은 실제 양성 클래스 중에서 모델이 얼마나 많은 양성을 예측했는지 측정합니다.

Send a message

Regenerate

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. [ChatGPT August 3 Version](#)

분류모델 성능측정 - 평가지표

분류결과(y_pred) = [0, 0, 1, 1, 0, 0]

실제정답(y_true) = [1, 0, 1, 1, 0, 1]

		실제값	
		1	0
예측값	1	TP 2	FP 0
	0	FN 2	TN 2

$$\text{Precision} = \frac{TP}{TP + FP}$$
$$= \frac{2}{2+0} = \frac{2}{2} = 1$$

$$\text{Recall} = \frac{TP}{TP + FN}$$
$$= \frac{2}{2+2} = \frac{2}{4} = 0.5$$

분류모델 성능측정

classification_performance.ipynb

```
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)
```

```
print(cm)
```

```
[Out] array([[2, 0],
            [2, 2]], dtype=int64)
```

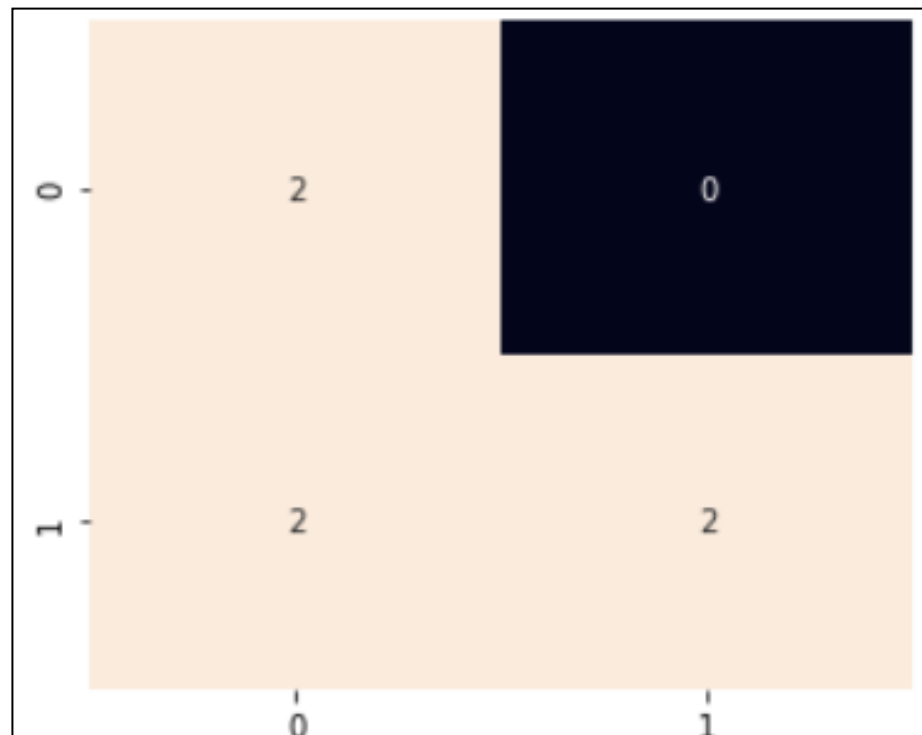
```
sns.heatmap(cm, annot=True)
```

```
precision_score(y_true, y_pred)
```

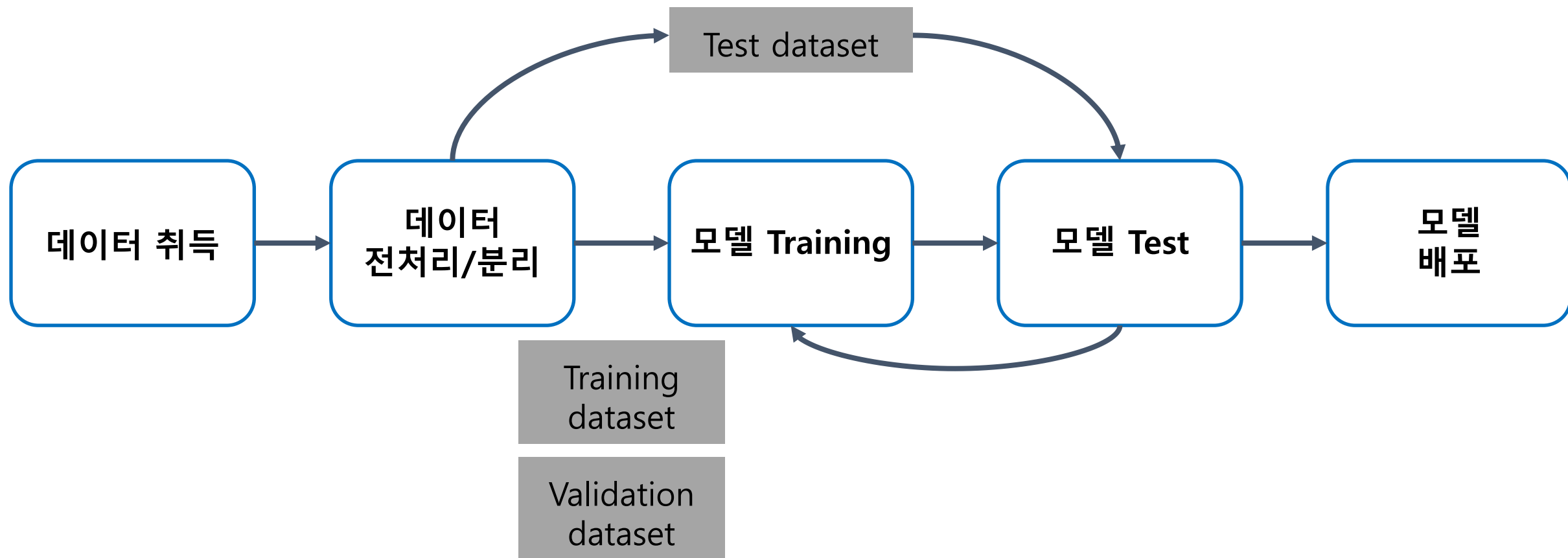
```
[Out] 1.0
```

```
recall_score(y_true, y_pred)
```

```
[Out] 0.5
```



머신러닝 프로세스



사이킷런 (Scikit-learn)

가장 인기있는 머신러닝 패키지이며, 많은 머신러닝 알고리즘이 내장되어 있습니다.

 [Install](#) [User Guide](#) [API](#)

<https://scikit-learn.org>

scikit-learn

Machine Learning in Python

[Getting Started](#) [Release Highlights for 0.23](#) [GitHub](#)

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...



Clustering

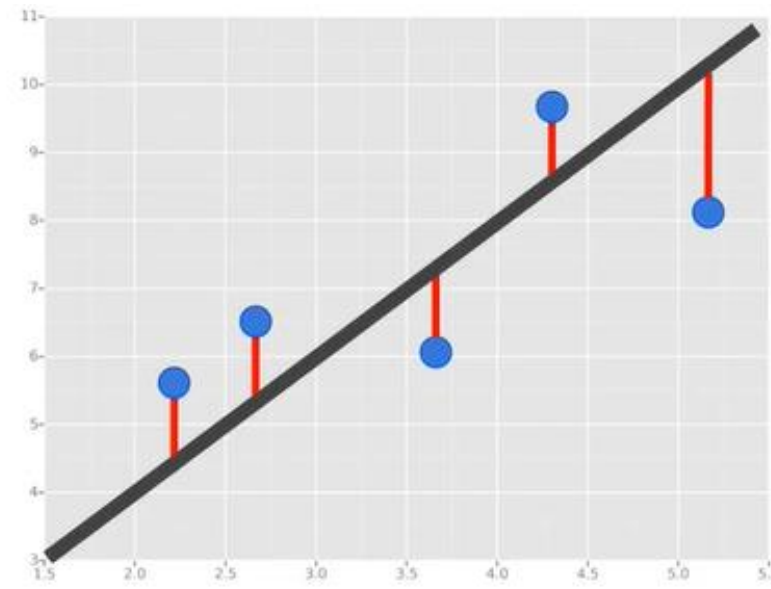
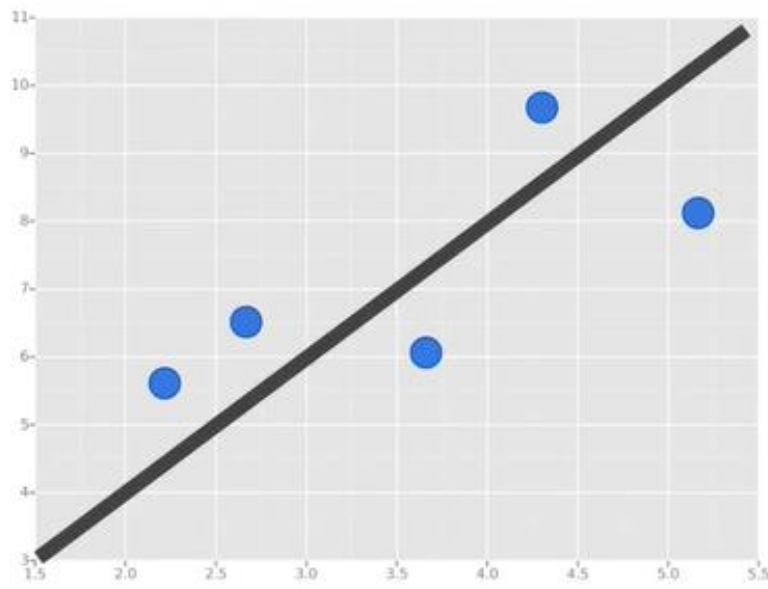
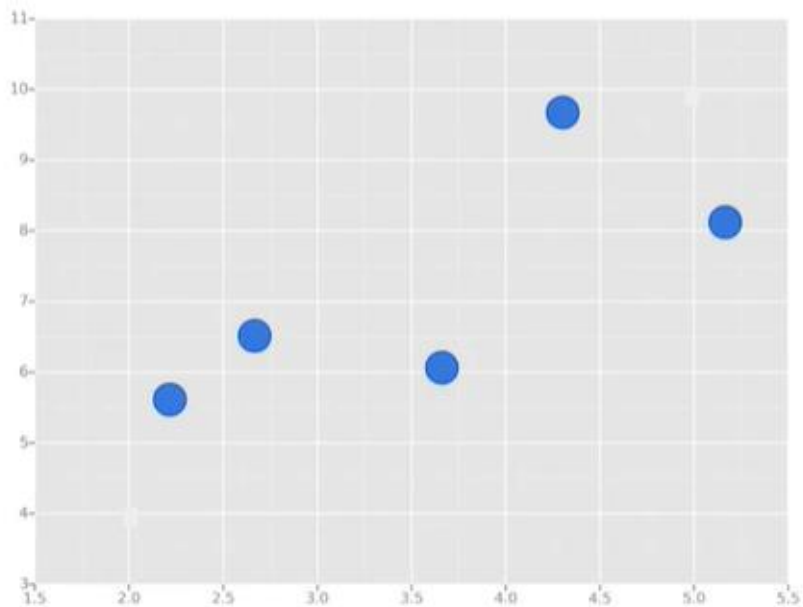
Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



선형 회귀 (Linear Regression)



선형 회귀 (Linear Regression)



LinearRegression.ipynb

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

X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape(-1,1)
y = np.array([13, 25, 34, 47, 59, 62, 79, 88, 90, 100])
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)
```

ElasticNet

1. 릿지(Ridge) 회귀

- 릿지 회귀는 L2 정규화를 사용하여 모델의 복잡성을 줄이고 과적합을 방지합니다.
- L2 정규화는 모델 계수의 제곱 합을 비용 함수에 추가하여 계수 값을 작게 만드는 역할을 합니다.

2. 라쏘(Lasso) 회귀

- 라쏘 회귀는 L1 정규화를 사용하여 모델의 변수 개수를 줄이고 중요한 변수를 선택하는 데 도움을 줍니다.
- L1 정규화는 모델 계수의 절댓값 합을 비용 함수에 추가하여 일부 계수 값을 0으로 만들어 변수를 제거합니다.

3. 엘라스틱넷 회귀

- 엘라스틱넷은 릿지 회귀의 L2 정규화와 라쏘 회귀의 L1 정규화를 모두 사용하여 모델의 과적합을 방지하고 중요한 변수를 선택하는 데 도움을 줍니다.
- 엘라스틱넷 클래스의 주요 생성 파라미터는 α 와 $l1_ratio$ 입니다.
- 엘라스틱넷의 규제는 $a \cdot L1 + b \cdot L2$ 로 정의될 수 있으며, 이 때 a 는 L1규제의 α 값, b 는 L2규제의 α 값입니다.
- ElasticNet클래스의 α 파라미터 값은 $a+b$ 입니다.
- ElasticNet클래스의 $l1_ratio$ 파라미터 값은 $a/(a+b)$ 입니다.
- $l1_ratio$ 가 0이면 a 가 0이므로 L2규제와 동일하고, $l1_ratio$ 가 1이면 b 가 0이므로 L1규제와 동일합니다.

분류 (Classification)



Classification.ipynb

■ setosa



■ versicolor



■ virginica



```
from sklearn import datasets
from sklearn.model_selection import train_test_split
iris = datasets.load_iris()
```

Train Test 데이터셋 분할

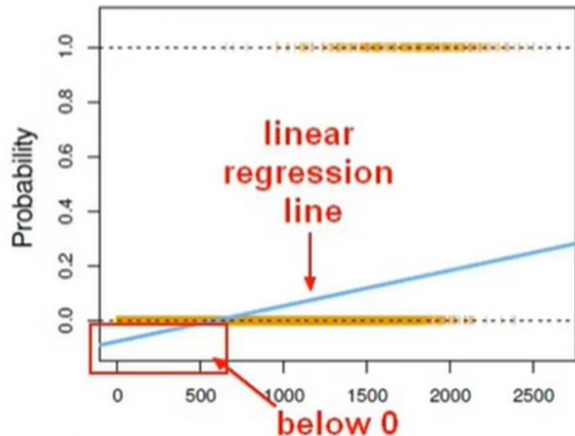
```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(  
    iris['data'],  
    iris['target'],  
    test_size=0.3,  
    shuffle=True,  
    stratify=iris.target,  
    random_state=42)
```

로지스틱 회귀(Logistic Regression)

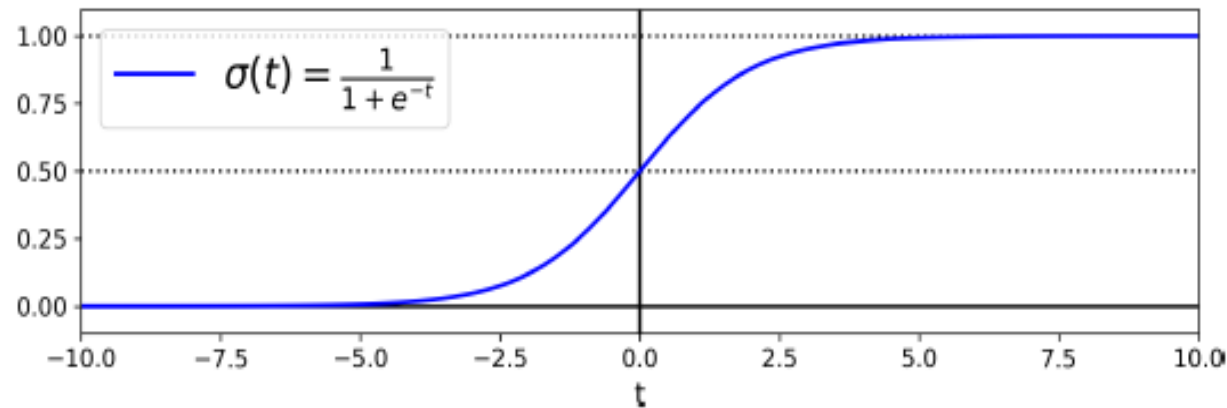
- 이진 분류 규칙은 0과 1의 두 클래스를 갖는 것으로, 일반 선형 회귀 모델을 이진분류에 사용할 수 없습니다.
- 대신 선형 회귀를 로지스틱 회귀 곡선으로 변환 할 수 있으며, 로지스틱 회귀 곡선은 0과 1 사이에서만 이동할 수 있으므로 분류에 사용할 수 있습니다.
- 로지스틱 회귀는 선형 회귀처럼 바로 결과를 출력하지 않고 로지스틱(logistic)을 출력합니다.
- 로지스틱 회귀는 샘플이 특정 데이터에 속할 확률을 추정(이진분류)하는 데 사용됩니다.
- 추정 확률이 50%가 넘으면 모델은 그 샘플이 해당 클래스에 속한다고 예측합니다.

일반 선형 모델



로지스틱 함수

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



로지스틱 확률모델

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

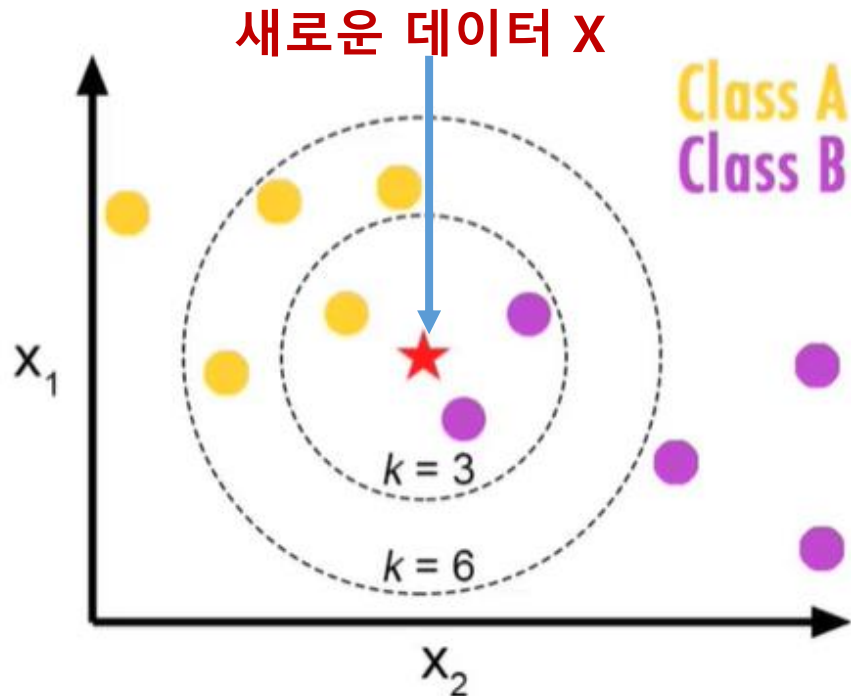
$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} \geq 0.5 \end{cases}$$

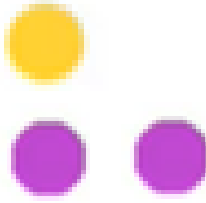
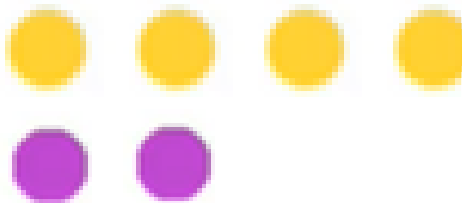
로지스틱 회귀(Logistic Regression)

```
from sklearn.linear_model import LogisticRegression
# 모델 학습
lr = LogisticRegression()
lr.fit(X_train, y_train)
# 예측
pred = lr.predict(X_test)
print(f'예측값: {pred[:10]}')
# 모델 성능 평가
accuracy = accuracy_score(y_test, pred)
print(f'Mean accuracy score: {accuracy:.4}')
# 확률값
prob = lr.predict_proba(X_test)
print(f'Probability: {prob[0]}')
```

KNN (K-Nearest Neighbor)

- KNN은 새로운 데이터가 주어졌을 때 기존 데이터 가운데 가장 가까운 k 개 이웃의 정보로 새로운 데이터를 예측하는 방법론입니다. 아래 그림처럼 검은색 점의 범주 정보는 주변 이웃들을 가지고 추론해낼 수 있습니다.
- 만약 k 값이 3이면 **Class B**, k 가 6이면 **Class A**로 분류(classification)하는 것입니다.
- 만약, 회귀(regression) 문제라면 이웃들 종속변수(y)의 평균이 예측값이 됩니다.
- 알고리즘이 간단하며 큰 데이터셋과 고차원 데이터에 적합하지 않은 단점이 있습니다.



K	이웃(Neighbor)	예측값
3		Class B
7		Class A

KNN (K-Nearest Neighbor)

```
from sklearn.linear_model import KNeighborsClassifier
```

```
# 모델 학습
```

```
knn = KNeighborsClassifier(n_neighbors=7)
```

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

```
# 예측
```

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

```
print(f'예측값: {pred[:10]}')
```

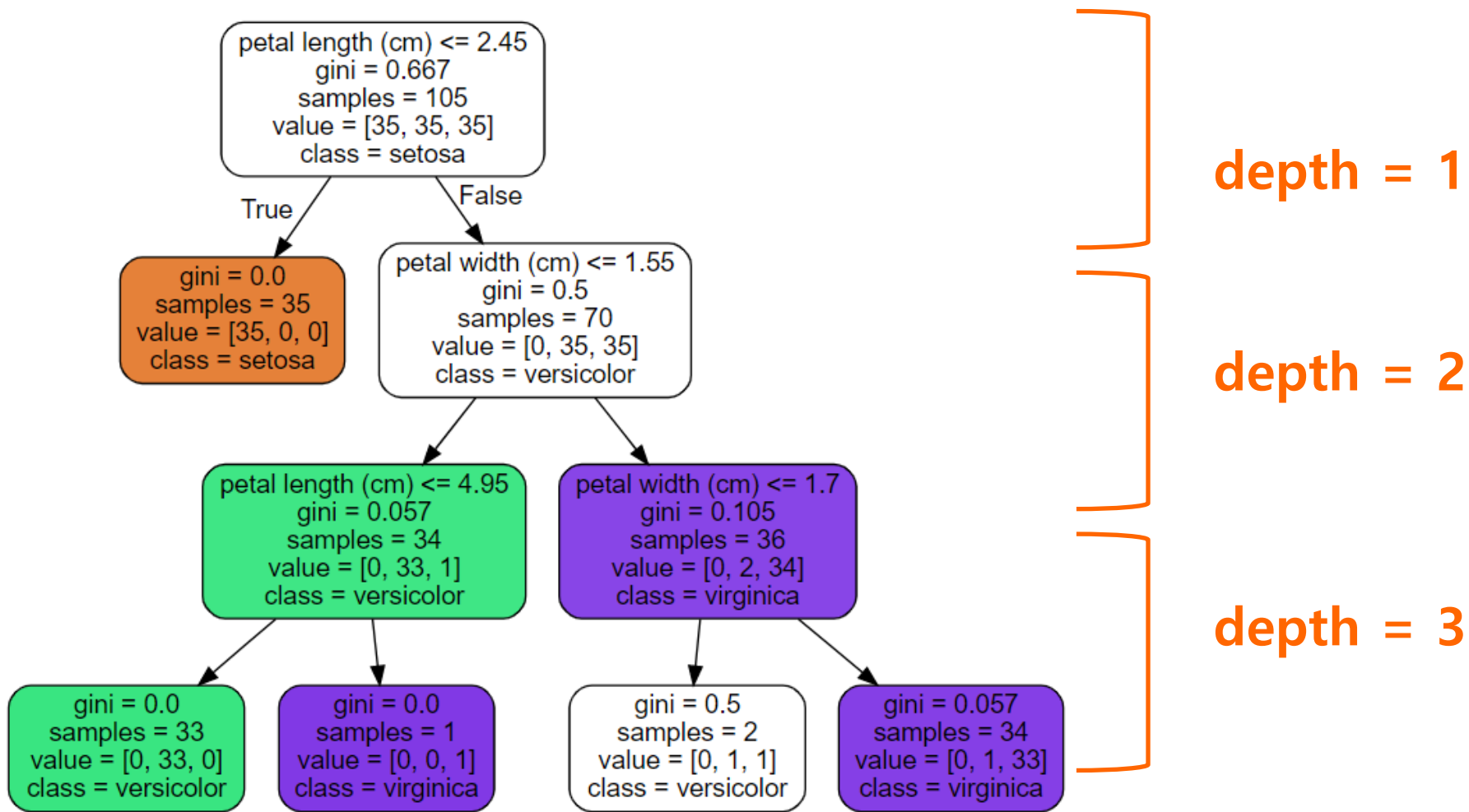
```
# 모델 성능 평가
```

```
accuracy = accuracy_score(y_test, pred)
```

```
print(f'Mean accuracy score: {accuracy:.4}')
```

의사결정트리 (Decision Tree)

의사결정트리 모델은 트리(Tree) 알고리즘을 사용합니다. 트리의 각 분기점(node)에 데이터셋의 Feature를 하나씩 위치시키고, 각 분기점(node)에서 임의의 조건식으로 가지를 나누면서 데이터를 구분합니다.



의사결정트리 (Decision Tree)

```
from sklearn.linear_model import DecisionTreeClassifier

# 모델 학습
dtc = DecisionTreeClassifier(max_depth=3, random_state=42)
dtc.fit(X_train, y_train)

# 예측
pred = dtc.predict(X_test)
print(f'예측값: {pred[:10]}')

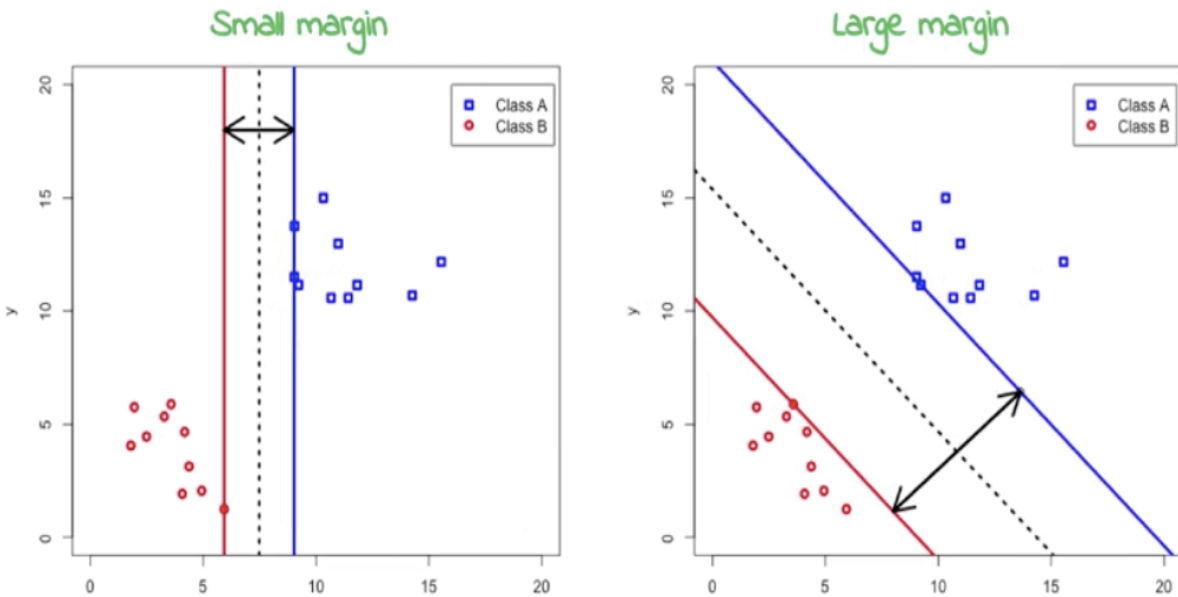
# 모델 성능 평가
accuracy = accuracy_score(y_test, pred)
print(f'Mean accuracy score: {accuracy:.4}')

# 확률값
prob = dtc.predict_proba(X_test)
print(f'Probability: {prob[0]}')
```

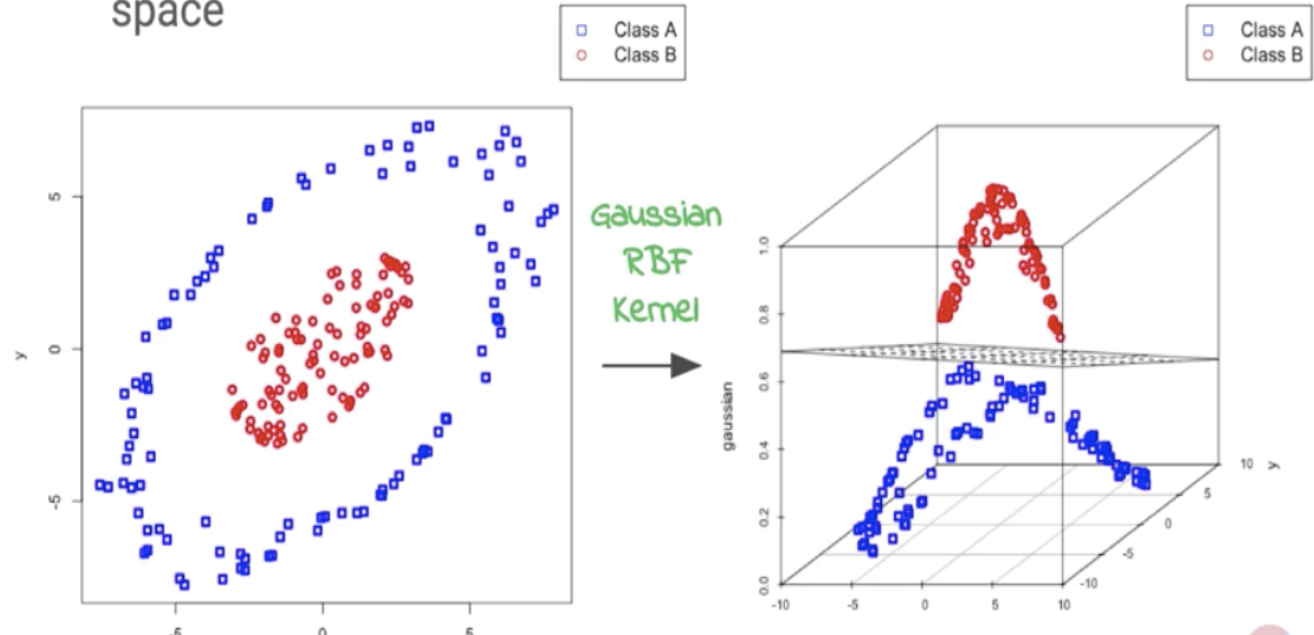
서포트 벡터 머신(SVM)

- 서포트 벡터 머신은 선형/비선형 분류, 회귀, 이상치 탐색에도 사용할 수 있는 다목적 머신러닝 모델입니다.
- SVM은 복잡한 분류 모델에 잘 들어 맞으며 작거나 중간 크기의 데이터셋에 적합합니다.

SVMs maximize the margin between two classes



Kernels transform the input space into a more usable feature space



서포트 벡터 머신(SVM)

```
from sklearn.svm import SVC
```

```
# 모델 학습
```

```
svc = SVC(kernel='rbf')
```

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

```
# 예측
```

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

```
print(f'예측값: {pred[:10]}')
```

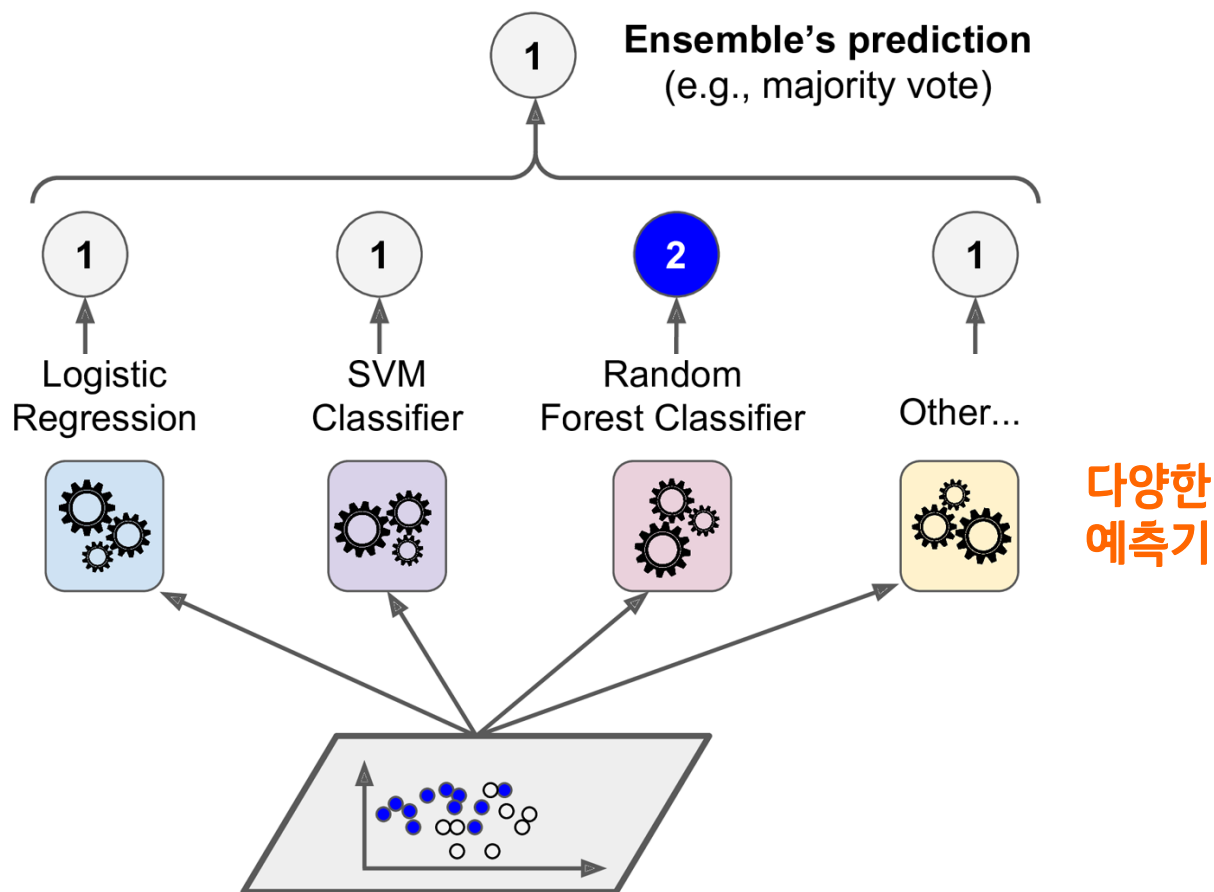
```
# 모델 성능
```

```
acc = accuracy_score(y_test, pred)
```

```
print(f'Mean accuracy score: {accuracy:.4}')
```

앙상블 학습(Ensemble Learning)

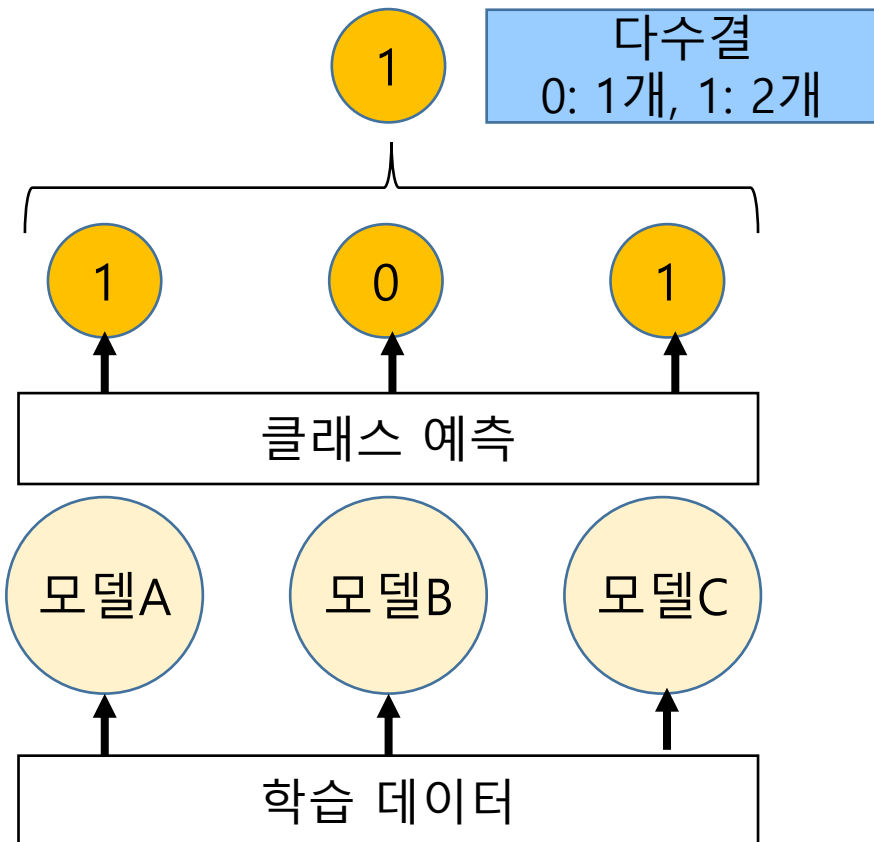
- 일련의 예측기(분류, 회귀)로부터 예측을 수집하면, 가장 좋은 모델 1개보다 더 좋은 예측을 얻을 수 있을 것입니다.
- 일련의 예측기를 앙상블이라 부르고 이를 Ensemble Learning이라고 합니다.
- 가장 인기 있는 앙상블 방법에는 배깅, 부스팅이 있습니다.



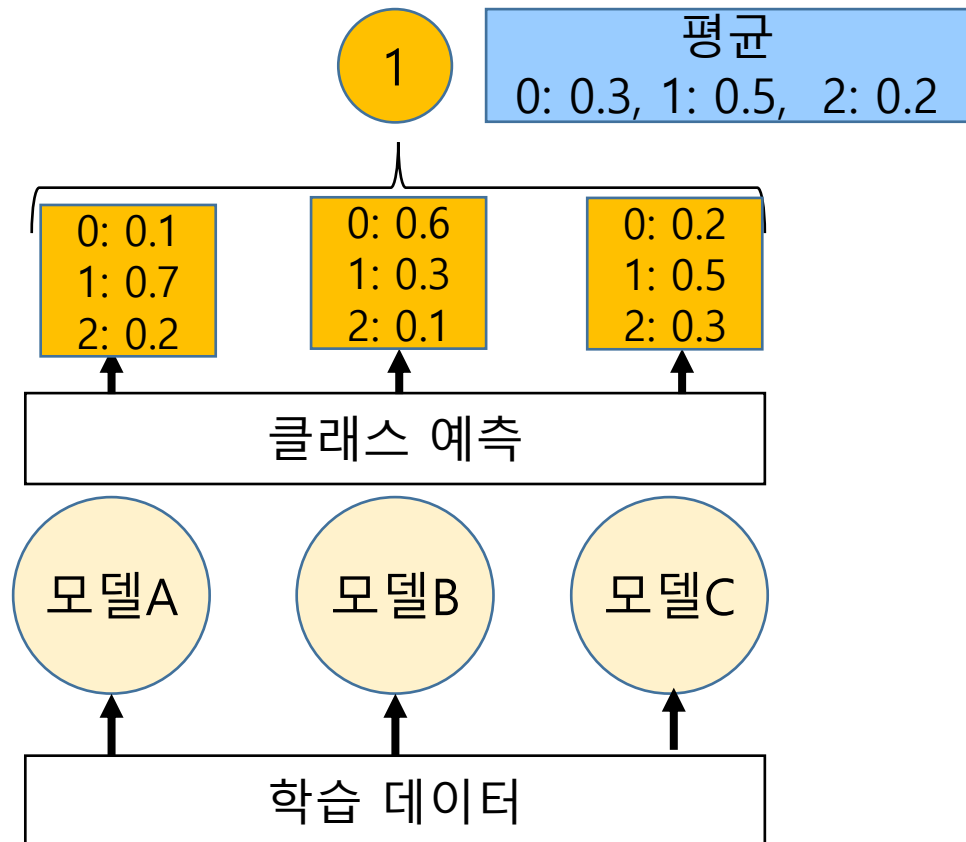
앙상블 모델 - 보팅(Voting)

- 보팅(Voting)은 여러 개의 모델이 예측한 값을 결합하여 최종 예측값을 결정하는 앙상블 방법입니다.
- 하드 보팅(hard voting)은 모델이 예측한 값 중에서 다수결로 최종 분류 클래스를 정합니다.
- 소프트 보팅(soft voting)은 각 분류 클래스별 예측 확률을 평균하여 최종 분류 클래스를 정합니다.

■ 하드 보팅(hard voting)



■ 소프트 보팅(soft voting)



앙상블 모델 - 보팅(Voting)

```
from sklearn.ensemble import VotingClassifier

# 모델 학습
hvc = VotingClassifier(estimators=[('KNN', knn), ('DT', dtc),
                                   ('SVM', svc)], voting='hard')
hvc.fit(X_train, y_train)

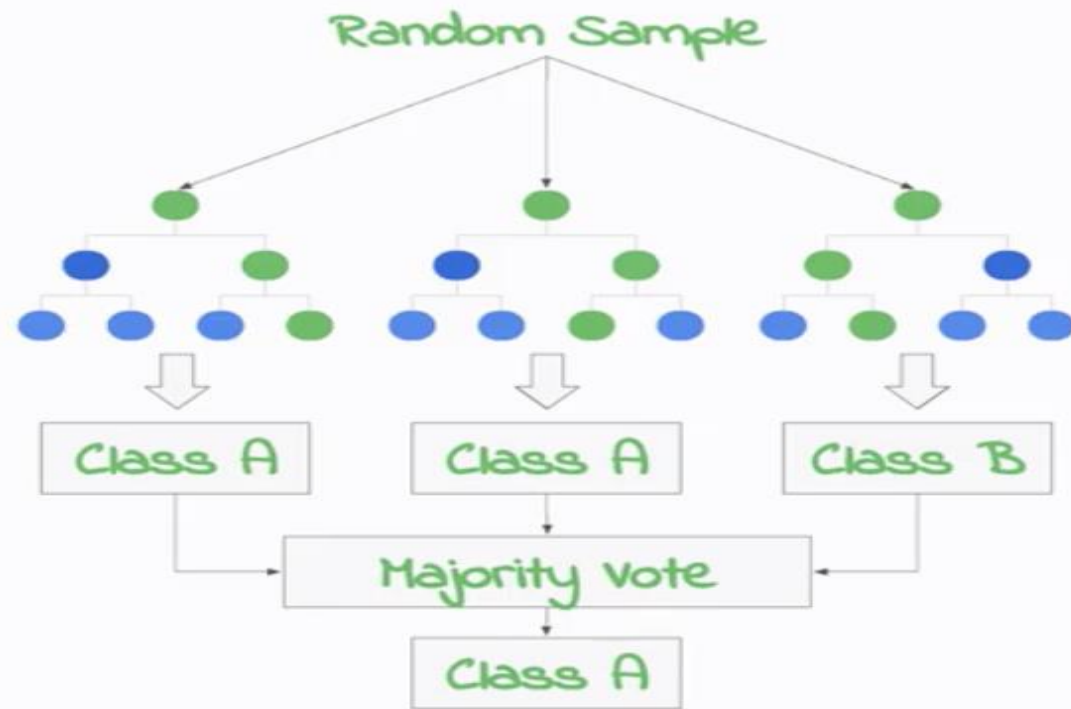
# 예측
pred = hvc.predict(X_test)
print(f'예측값: {pred[:10]}')

# 모델 성능 평가
accuracy = accuracy_score(y_test, pred)
print(f'Mean accuracy score: {accuracy:.4}')
```

앙상블 모델 - 배깅(Bagging)

- 다양한 분류기를 만드는 각기 다른 훈련 알고리즘을 사용하는 것과, 같은 알고리즘을 사용하고, 훈련 세트의 서브셋을 무작위로 구성하여 각기 다르게 학습시키는 방법이 있습니다.
- 훈련세트에서 중복을 허용하여 샘플링 하는 방식을 **bootstrap aggregating**, 배깅(**bagging**)라고 합니다.
- 통계학에서 중복을 허용한 리샘플링을 부트스트래핑(bootstrapping)이라고 합니다.
- 중복을 허용하지 않고 샘플링 하는 방식은 페이스팅(pasting)이라고 합니다.
- 랜덤 포레스트(Random Forest)는 일반적으로 배깅(또는 페이스팅)을 적용한 의사결정트리의 앙상블입니다.

Random forest: Strong learner from many weak learners



앙상블 모델 - 랜덤 포레스트(Random Forest, 배경)

```
from sklearn.ensemble import RandomForestClassifier
```

모델 학습

```
rfc = RandomForestClassifier(n_estimators=50, max_depth=3,  
                             random_state=20)
```

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

예측

```
pred = rfc.predict(X_test)  
print(f'예측값: {pred[:10]}')
```

모델 성능 평가

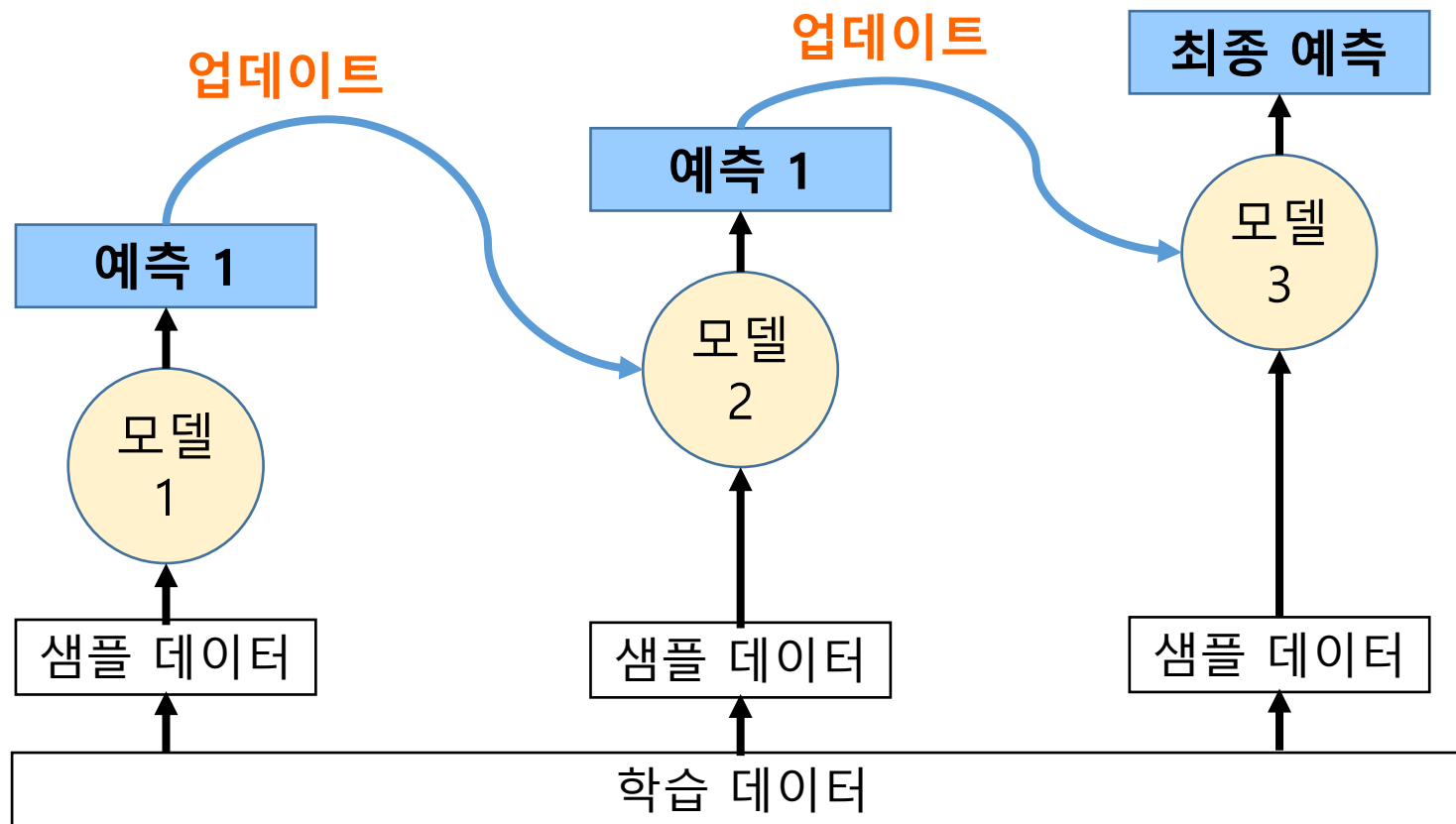
```
accuracy = accuracy_score(y_test, pred)  
print(f'Mean accuracy score: {accuracy:.4}')
```

shift+tab키 : 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',
```

앙상블 모델 - 부스팅(Boosting)

- 부스팅(Boosting)은 여러 개의 모델을 순차적으로 학습합니다.
- 잘못 예측한 데이터에 대한 예측 오차를 줄일 수 있는 방향으로 모델을 계속 업데이트 합니다.
- Kaggle(<https://www.kaggle.com/>) 경진대회에서 많이 사용되고 있는 알고리즘입니다.



앙상블 모델 - XGBoost (Extreme Gradient Boosting, 부스팅)

```
!pip install xgboost
```

```
from xgboost import XGBClassifier
```

```
# 모델 학습
```

```
xgbc = XGBClassifier(n_estimators=50, max_depth=3, random_state=42)
```

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

```
# 예측
```

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

```
print(f'예측값: {pred[:10]}')
```

```
# 모델 성능 평가
```

```
acc = accuracy_score(y_test, pred)
```

```
print(f'Mean accuracy score: {accuracy:.4}')
```

Mlflow 실습



BasicMLCode.py

```
lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
lr.fit(train_x, train_y)
predicted_qualities = lr.predict(test_x)

(rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)

print(f"Elasticnet model (alpha={alpha:f}, l1_ratio={l1_ratio:f})")
print(f"RMSE : {rmse}")
print(f"MAE : {mae}")
print(f"R2 : {r2}")
```

■ 프로그램 실행

Python 가상환경 : py311\Scripts\activate.bat

code 디렉토리로 이동 : cd code

python BasicMLCode.py

BasicMLCode_MLflow.py

```
exp = mlflow.set_experiment(experiment_name="experment_1")

with mlflow.start_run(experiment_id=exp.experiment_id):
    lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
    lr.fit(train_x, train_y)
    predicted_qualities = lr.predict(test_x)
    (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)

    mlflow.log_param("alpha", alpha)
    mlflow.log_param("l1_ratio", l1_ratio)

    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("r2", r2)
    mlflow.log_metric("mae", mae)

    mlflow.sklearn.log_model(lr, "mymodel")
```

■ 프로그램 실행

python BasicMLCode_MLflow.py

python BasicMLCode_MLflow.py --alpha 0.5 --l1_ratio 0.5

python BasicMLCode_MLflow.py --alpha 0.7 --l1_ratio 0.7

python BasicMLCode_MLflow.py --alpha 0.9 --l1_ratio 0.7

MLflow 실습

mlflow ui

<http://localhost:5000/> 접속

mlflow up -p 1234

<http://localhost:1234/> 접속

The screenshot displays the MLflow 2.10.2 Experiments page. The left sidebar shows a list of experiments, with 'experiment_1' selected. The main area shows the details for 'experiment_1', including its ID and a search filter. A dropdown menu is open, showing the 'Columns' section with 'Metrics (3)' selected, listing 'mae', 'r2', and 'rmse'. The 'Experimental' tab is active, showing a table of runs. The 'Metrics' section for the selected run is highlighted, showing values for 'mae', 'r2', and 'rmse'.

Metrics							
	mae	r2	rmse				
0.665527385...	0.020410131...	0.831486...					
0.664718804...	0.020574420...	0.831416...					
0.664912271...	0.020386384...	0.831496...					

Thank you 😊