클라우드 컴퓨팅과 AI서비스 (8주차)

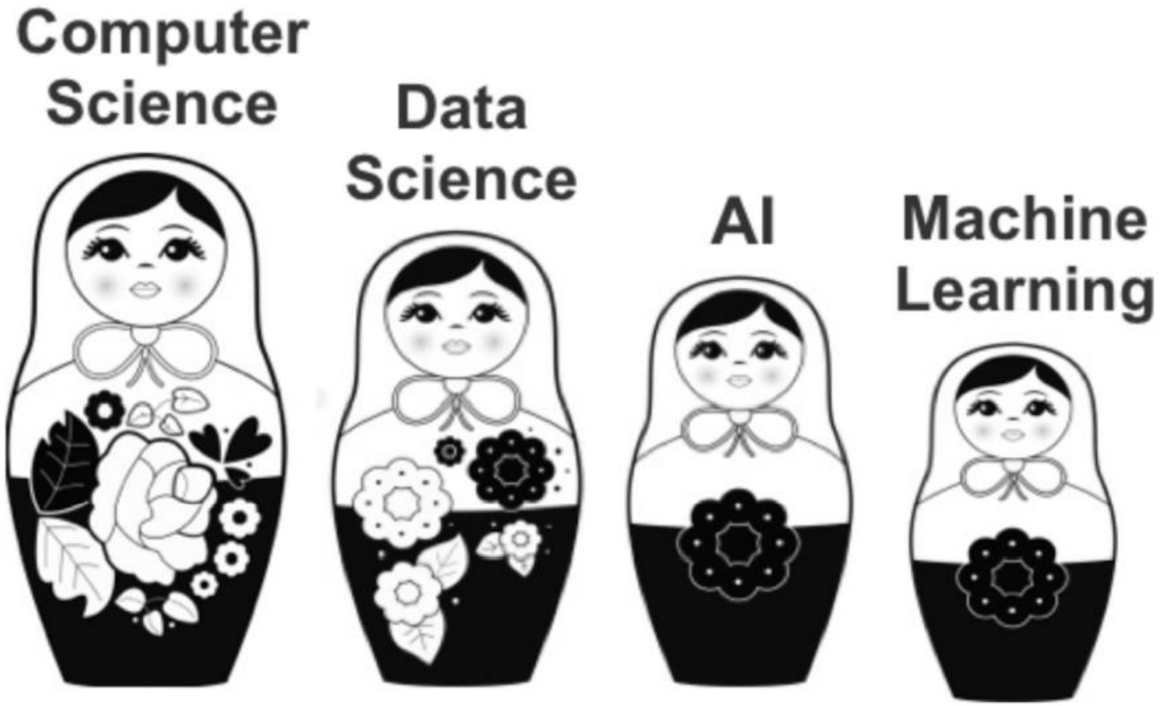
융합학과 권오영
oykwon@koreatech.ac.kr

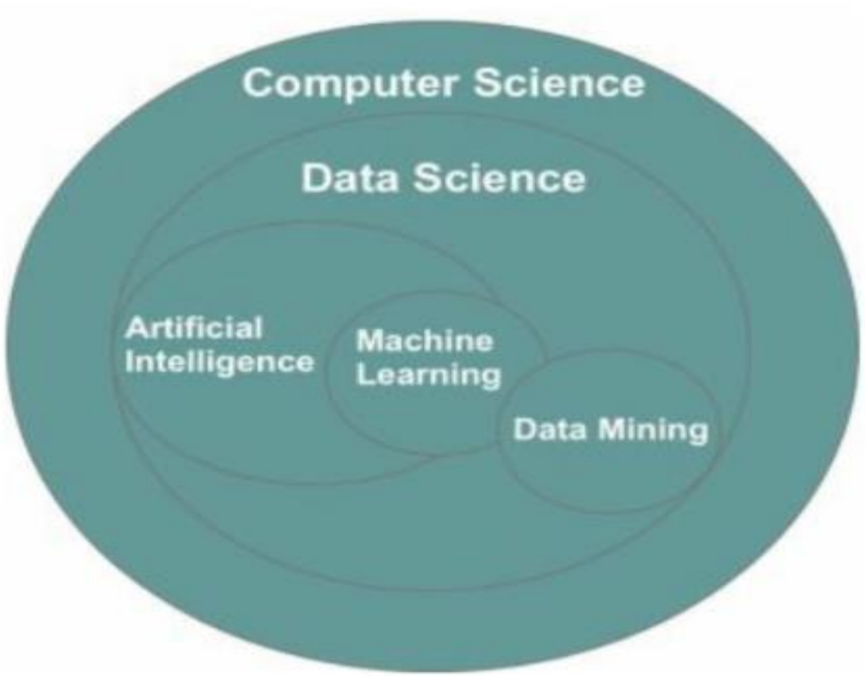
학습내용

- ❖ 기계학습
- ❖ 구글티쳐블머신
- ❖ scikit learn (sklearn) 기계학습패키지
  - ▪ MNIST 실습

기계학습

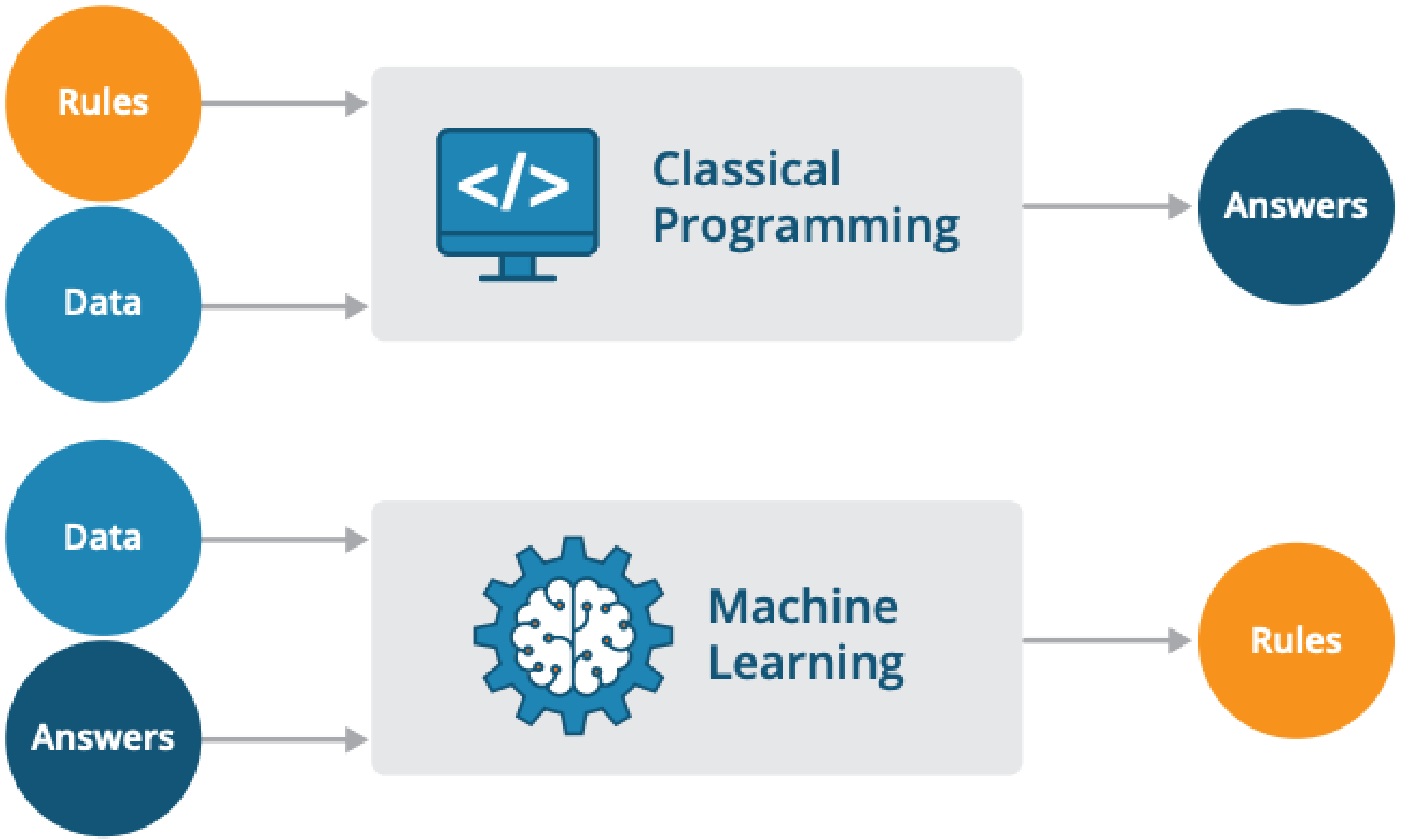
The Anatomy of Machine Learning



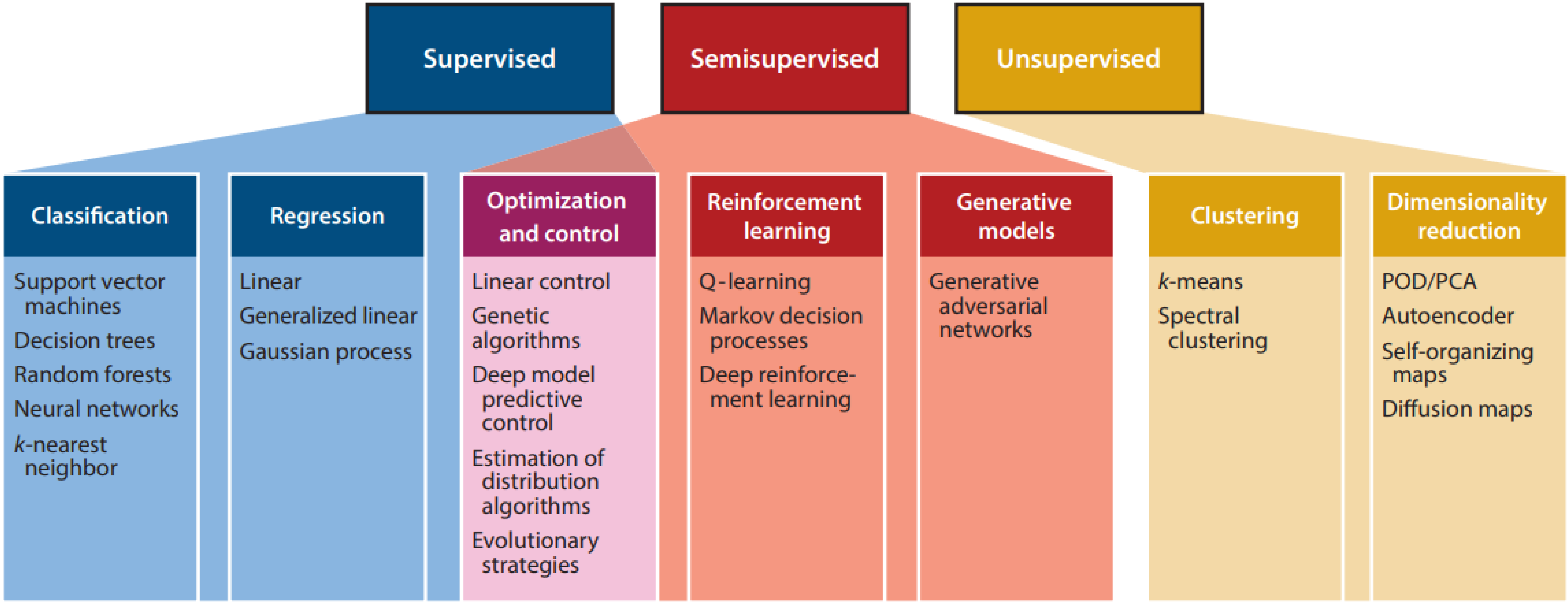


Classical Programming vs ML

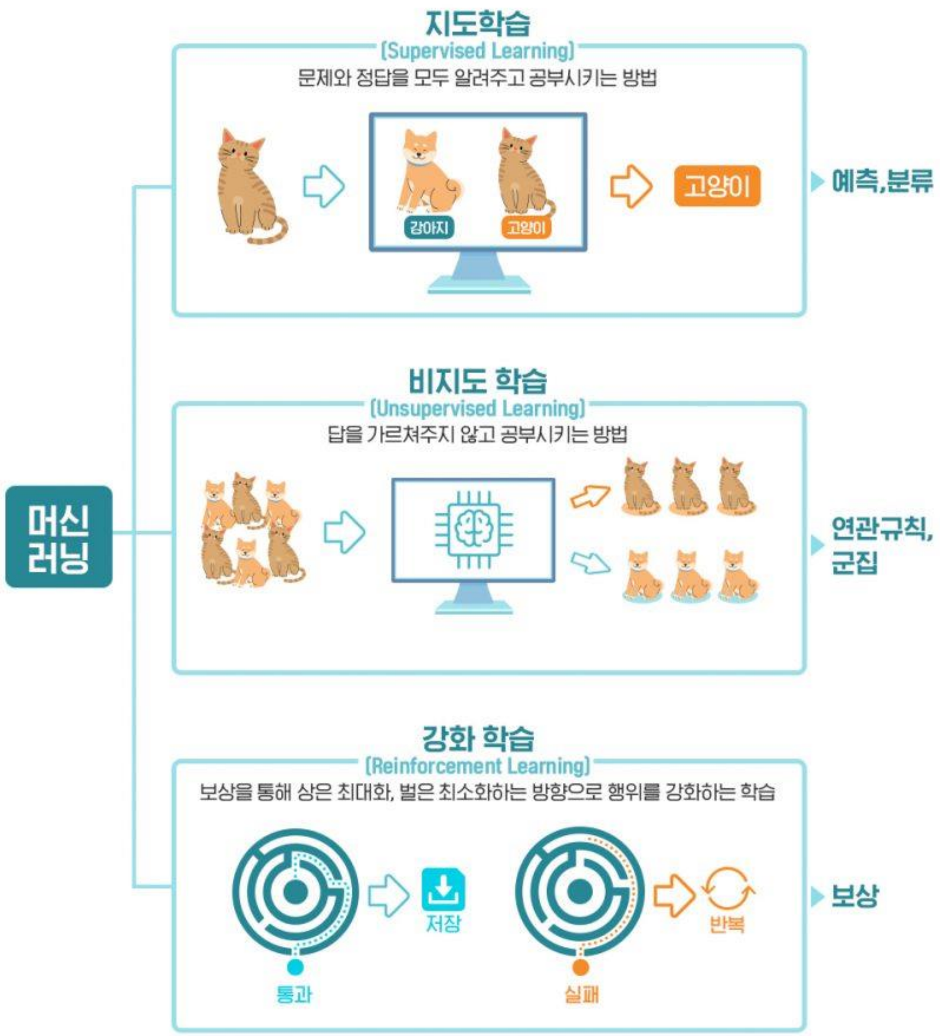
❖ Classical Programming vs ML



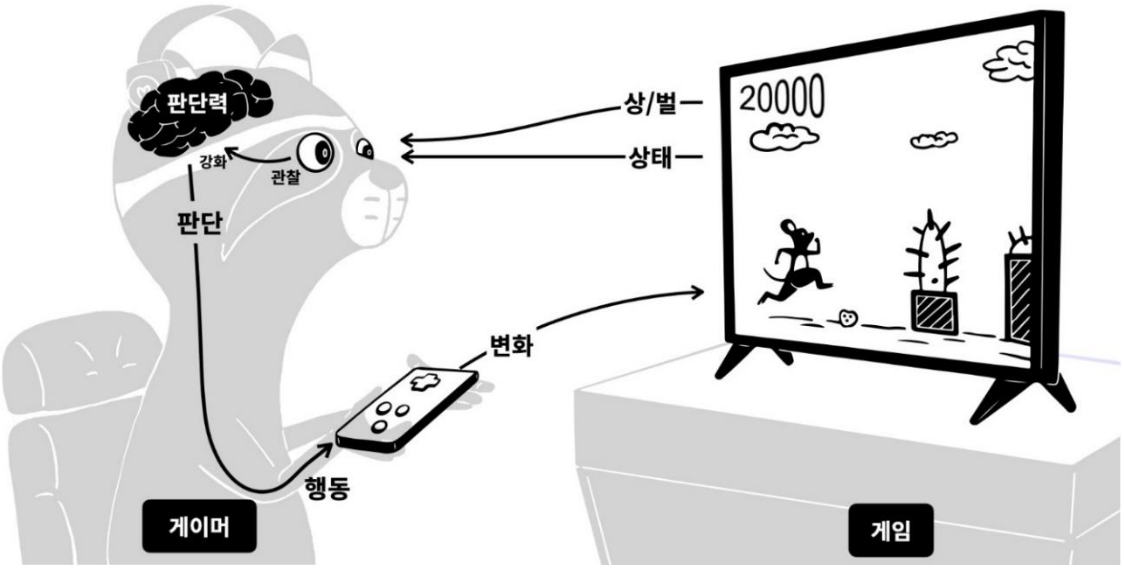
기계학습 분류


Supervised
Semisupervised
Unsupervised
Classification
Support vector machines
Decision trees
Random forests
Neural networks
k-nearest neighbor
Regression
Linear
Generalized linear
Gaussian process
Optimization and control
Linear control
Genetic algorithms
Deep model predictive control
Estimation of distribution algorithms
Evolutionary strategies
Reinforcement learning
Q-learning
Markov decision processes
Deep reinforcement learning
Generative models
Generative adversarial networks
Clustering
k-means
Spectral clustering
Dimensionality reduction
POD/PCA
Autoencoder
Self-organizing maps
Diffusion maps

기계학습방법



[출처] https://live.lge.co.kr/live_with_ai_01/



- 게임 ➡ 환경(environment)
- 게이머 ➡ 에이전트(agent)
- 게임화면 ➡ 상태(state)
- 게이머의 조작 ➡ 행동(action)
- 상과 벌 ➡ 보상(reward)
- 게이머의 판단력 ➡ 정책(policy)

[출처] https://opentutorials.org/course/4548/28949

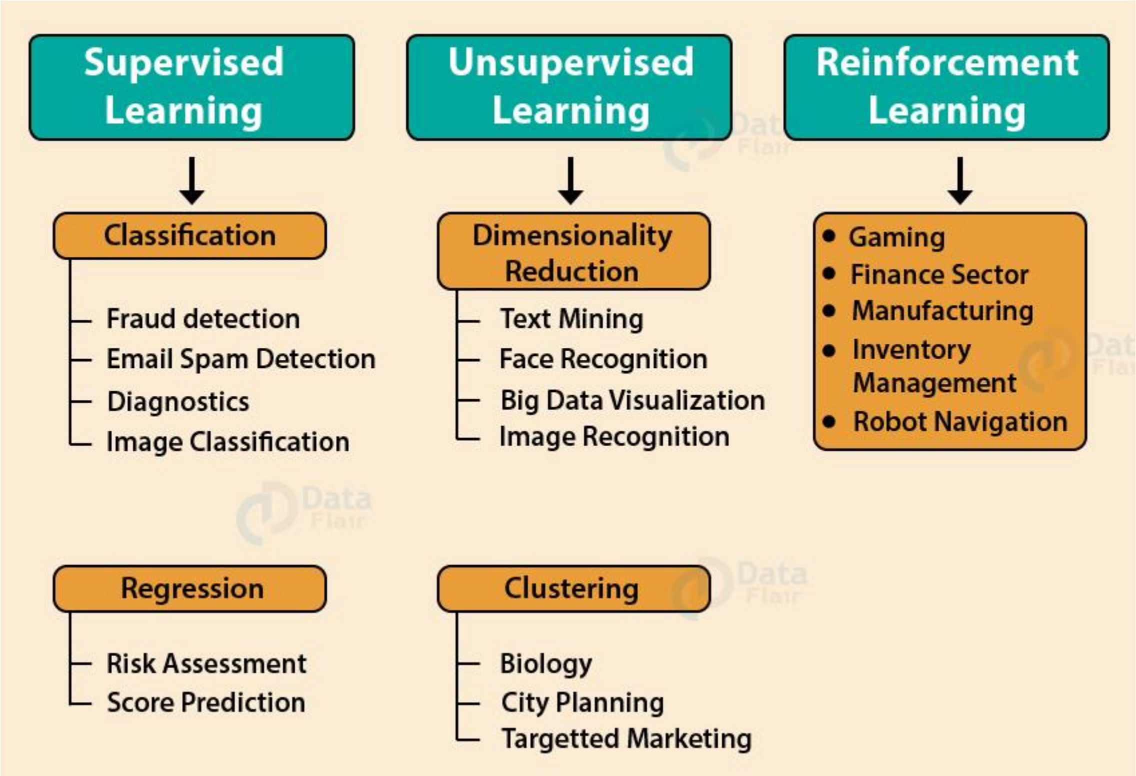
KOREATECH 한국기술교육대학교

Types of machine learning

- **Supervised**(or predictive) learning
  - learn a mapping from inputs x to outputs y
  - training set (input-output pairs) $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$

- **Unsupervised**(or descriptive) learning
  - only given inputs, $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}$
  - goal : find “interesting patterns” in the data.

- **Reinforcement** learning
  - reward or punishment signals.



KOREATECH 한국기술교육대학교

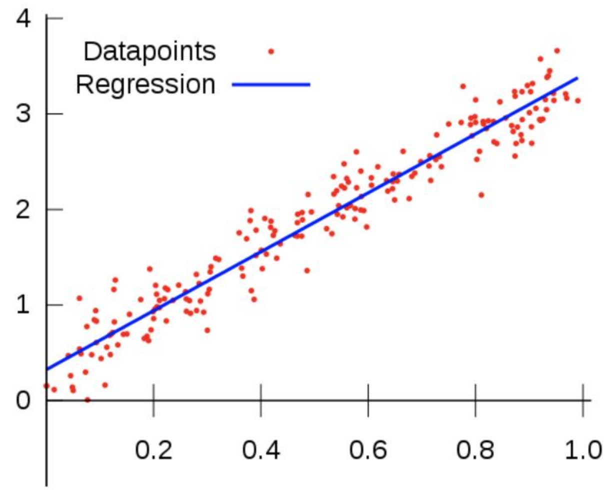
지도학습

- ❖ 결과(레이블(lable))와 입력을 같이 주어서 학습하고, 주어진 입력을 분류(classification) 하거나 예측하는 회귀(regression)가 있음
- ❖ 전통적인 기계학습 알고리즘
  - 선형 회귀: Linear Regression
  - 로지스틱 회귀: Logistic Regression
  - K-최근접 이웃: K-Nearest Neighbors
  - 결정 트리: Decision Tree
  - 랜덤 포레스트: Random Forest
  - 서포트 벡터 머신: Support Vector Machine

선형회귀

❖ 가장 기본적인 알고리즘

❖ 데이터들과 오차가 가장 적은 회귀선 생성



$$y = a_1 x + a_0$$

주어진 x, y 값을 이용해서 a_0, a_1 를 구함

$$S(a_0, a_1) = \sum_{i=0}^{n}[y_i - f(x_i)]^2 = \sum_{i=0}^{n}(y_i - a_0 - a_1 x_i)^2$$

$$\frac{\partial S}{\partial a_0} = \sum_{i=0}^{n} -2(y_i - a_0 - a_1 x_i) = 2\left[a_0(n+1) + a_1 \sum_{i=0}^{n} x_i - \sum_{i=0}^{n} y_i\right] = 0$$

$$\frac{a_0(n+1)}{n+1} + a_1 \frac{\sum_{i=0}^{n} x_i}{n+1} - \frac{\sum_{i=0}^{n} y_i}{n+1} = 0$$

$$a_0 = \bar{y} - a_1 \bar{x}$$

선형회귀

$$\frac{\partial S}{\partial a_1} = \sum_{i=0}^{n} -2(y_i - a_0 - a_1 x_i)x_i = 2\left[a_0 \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} x_i^2 - \sum_{i=0}^{n} x_i y_i\right] = 0$$

$$a_0 \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} x_i^2 - \sum_{i=0}^{n} x_i y_i = 0$$

$$(\bar{y} - a_1 \bar{x}) \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} x_i^2 - \sum_{i=0}^{n} x_i y_i = 0$$

$$\sum_{i=0}^{n} y_i \bar{x} - a_1 \sum_{i=0}^{n} x_i \bar{x} + a_1 \sum_{i=0}^{n} x_i^2 - \sum_{i=0}^{n} x_i y_i = 0$$

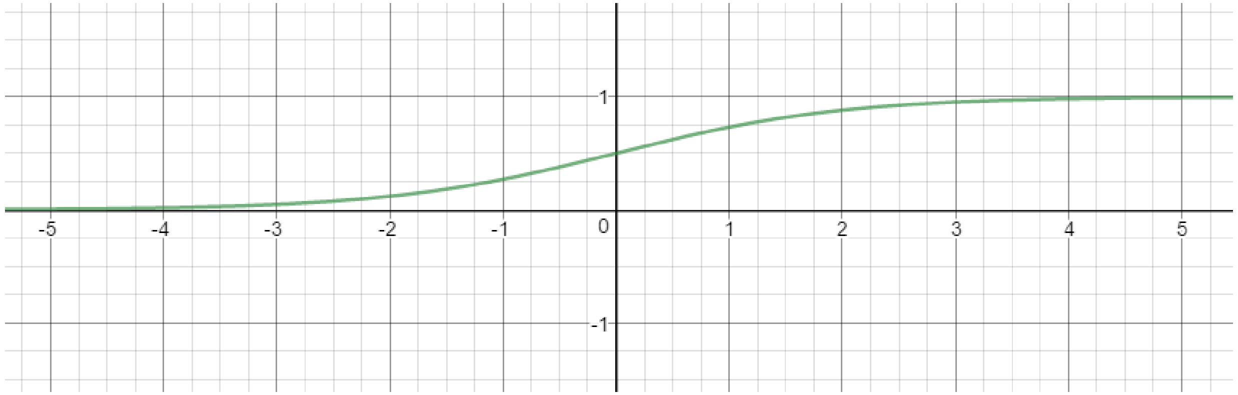
$$a_1 \sum_{i=0}^{n} x_i (x_i - \bar{x}) = \sum_{i=0}^{n} y_i (x_i - \bar{x}) \qquad a_1 = \frac{\sum_{i=0}^{n} y_i (x_i - \bar{x})}{\sum_{i=0}^{n} x_i (x_i - \bar{x})}$$

다중선형회귀 $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon$

로지스틱 회귀

❖ 출력결과를 0과 1사이로 변환

$$y = \frac{1}{1+e^{-x}}$$



❖ 이항분류

$$P(Y=1|X=\overrightarrow{x}) = \frac{1}{1+e^{-\overrightarrow{\beta}^T\overrightarrow{x}}}$$

❖ 다항분류

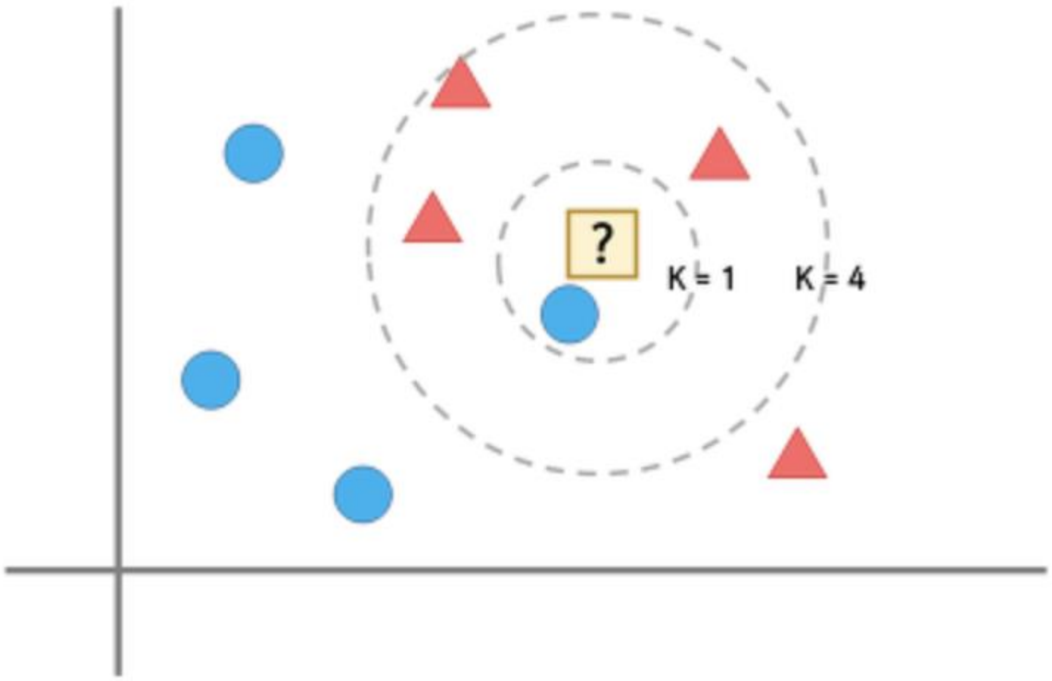
$$P(Y=k|X=\overrightarrow{x}) = \frac{e^{\overrightarrow{\beta_k}^T\overrightarrow{x}}}{1+\sum_{i=1}^{K-1} e^{\overrightarrow{\beta_i}^T\overrightarrow{x}}} \qquad (k=0,1,\ldots,K-1)$$

$$P(Y=K|X=\overrightarrow{x}) = \frac{1}{1+\sum_{i=1}^{K-1} e^{\overrightarrow{\beta_i}^T\overrightarrow{x}}}$$

KOREATECH 한국기술교육대학교

K-Nearest Neighbore

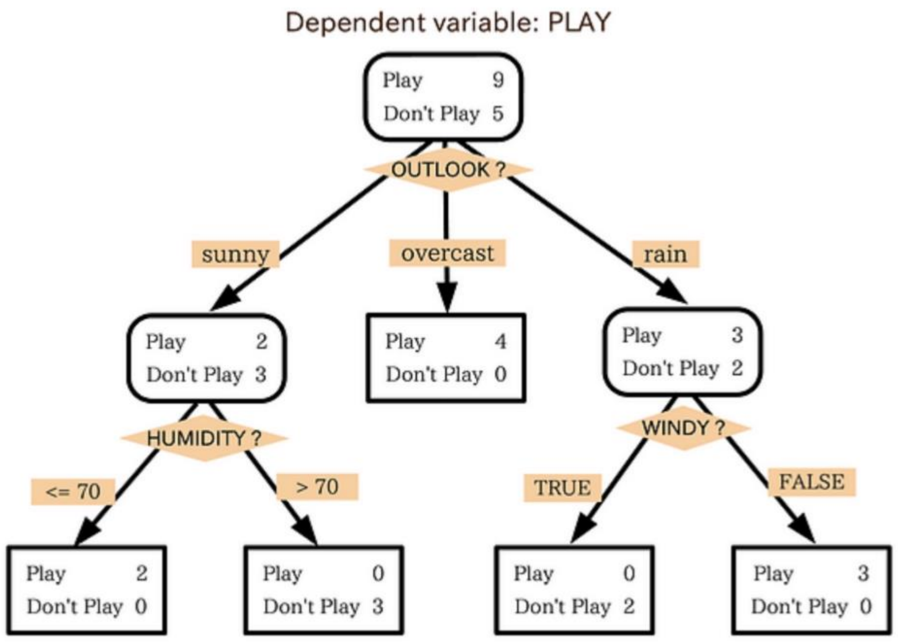
- ❖ 새로운 데이터를 입력 받았을 때 어디에 속하는지 결정하는 알고리즘
- ❖ ?는 어디에 가까운가? (k 반경에 있는 데이터들의 거리 제곱근의합)



결정 트리(Decision Tree)

❖ 운동경기가 진행여부 판단

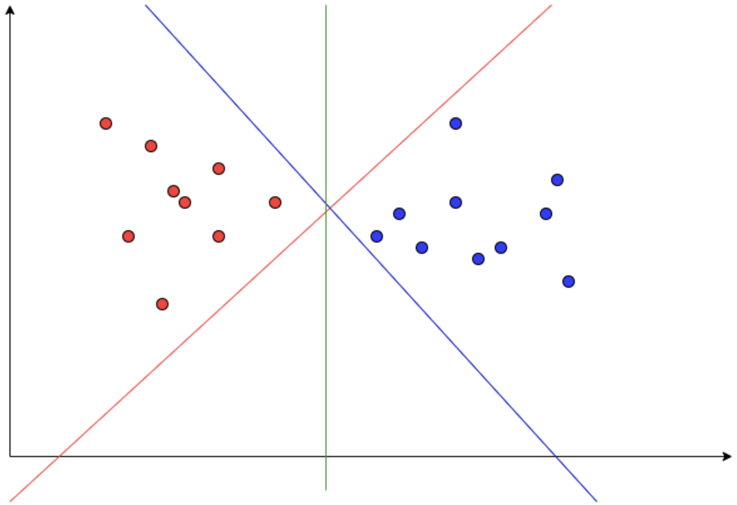
- 비가오지만 바람이 불지 않으면 경기가 열림
- 맑은날이지만 습도가 높으면 경기가 열리지 않음

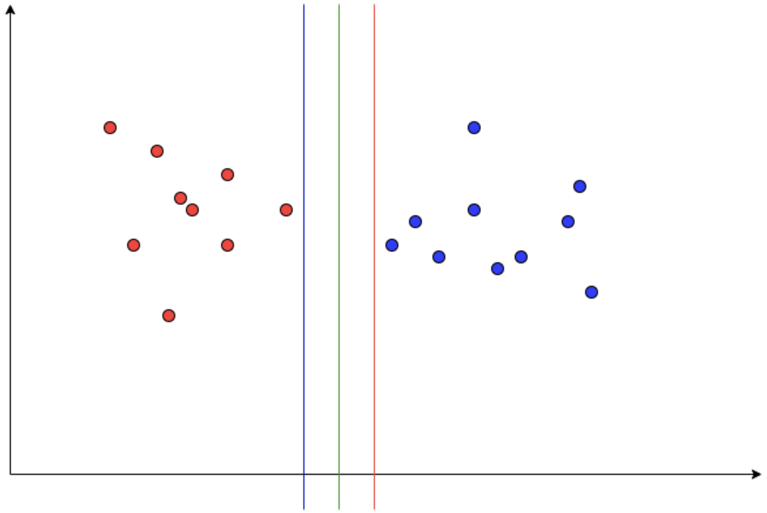


❖ 랜덤 포레스트는 결정트리들의 모임

SVM

❖ n 차원을 n-1 차원으로 나눌 수 있다.





❖ 저차원에서 선형분리가 안되는 것은 고차원을 확장해서 선형분리를 수행하고 저차원으로 환원

구글 티쳐블머신

Teachable Machine

❖ 코딩없이 응용제작 (https://teachablemachine.withgoogle.com/)

❖ Teachable Machine을 이용한 인공지능 서비스 만들기 예제?
(https://www.youtube.com/watch?v=UPgxnGC8oBU)

✓ 2초 딜레이후에 6초간 동작인식시키고
✓ 훈련을 시킨후에 (시간이 걸림 중간에 말을 하던지 생략)
✓ Export 해서 다른 프로그램에 사용
✓ 쉽게 만들 수 있음 2 ~ 3분 정도

❖ Youtube The coding train
https://www.youtube.com/user/shiffman
에서 teachable machine 을 검색

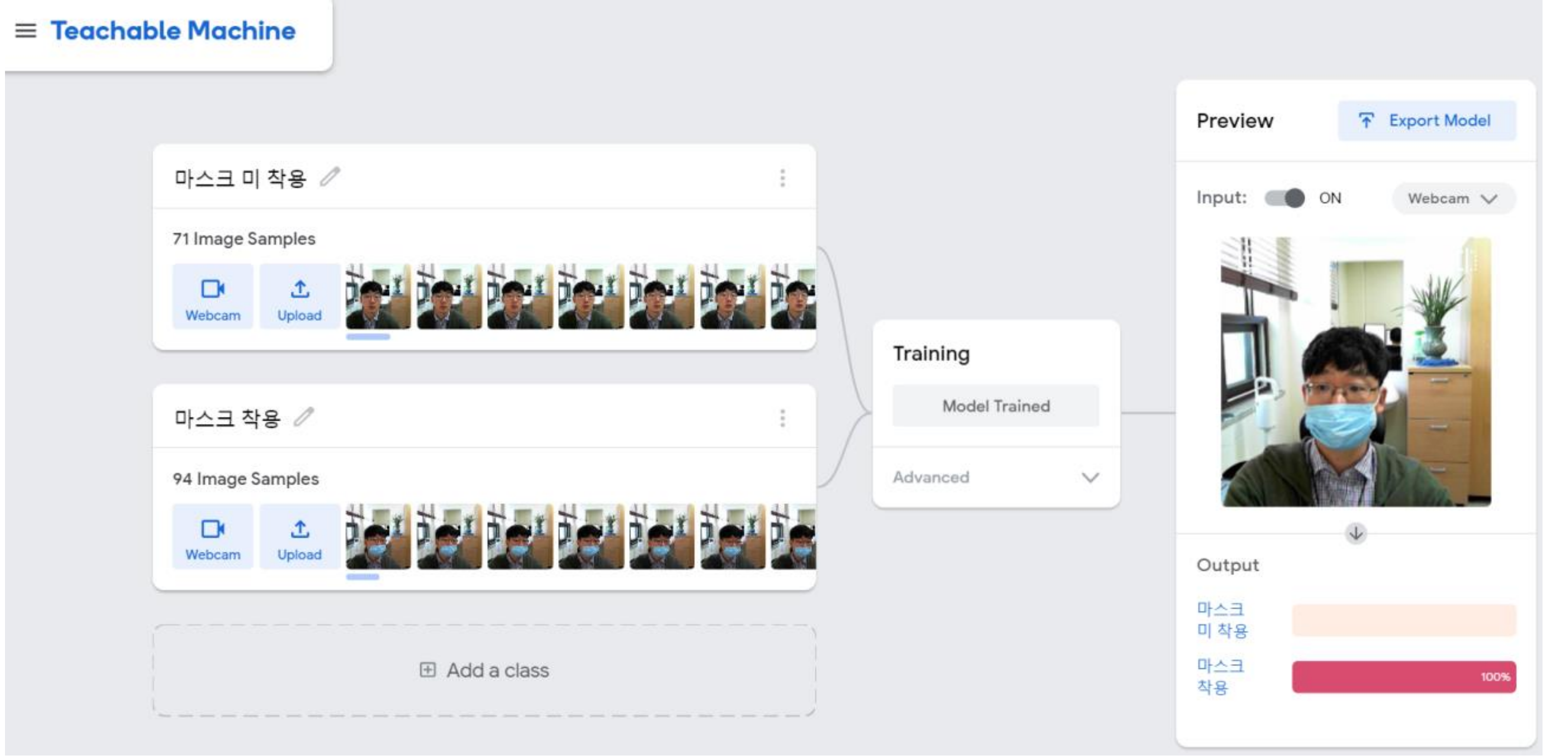
❖ 이미지, 소리, 포즈 인식

KOREATECH 한국기술교육대학교

Teachable machine 서비스 제작

❖ 각자 아이디어를 내어서 서비스 만들어 보기

❖ 예시) 마스크 착용 여부 판단



SCIKIT-LEARN

Scikit-learn (sklearn) 소개

- ❖ 기계학습 라이브러리
  - classification, regression, clustering, 차원축소 등 지원
- ❖ Numpy, Scipy, Matplotlib 를 활용하여 구성

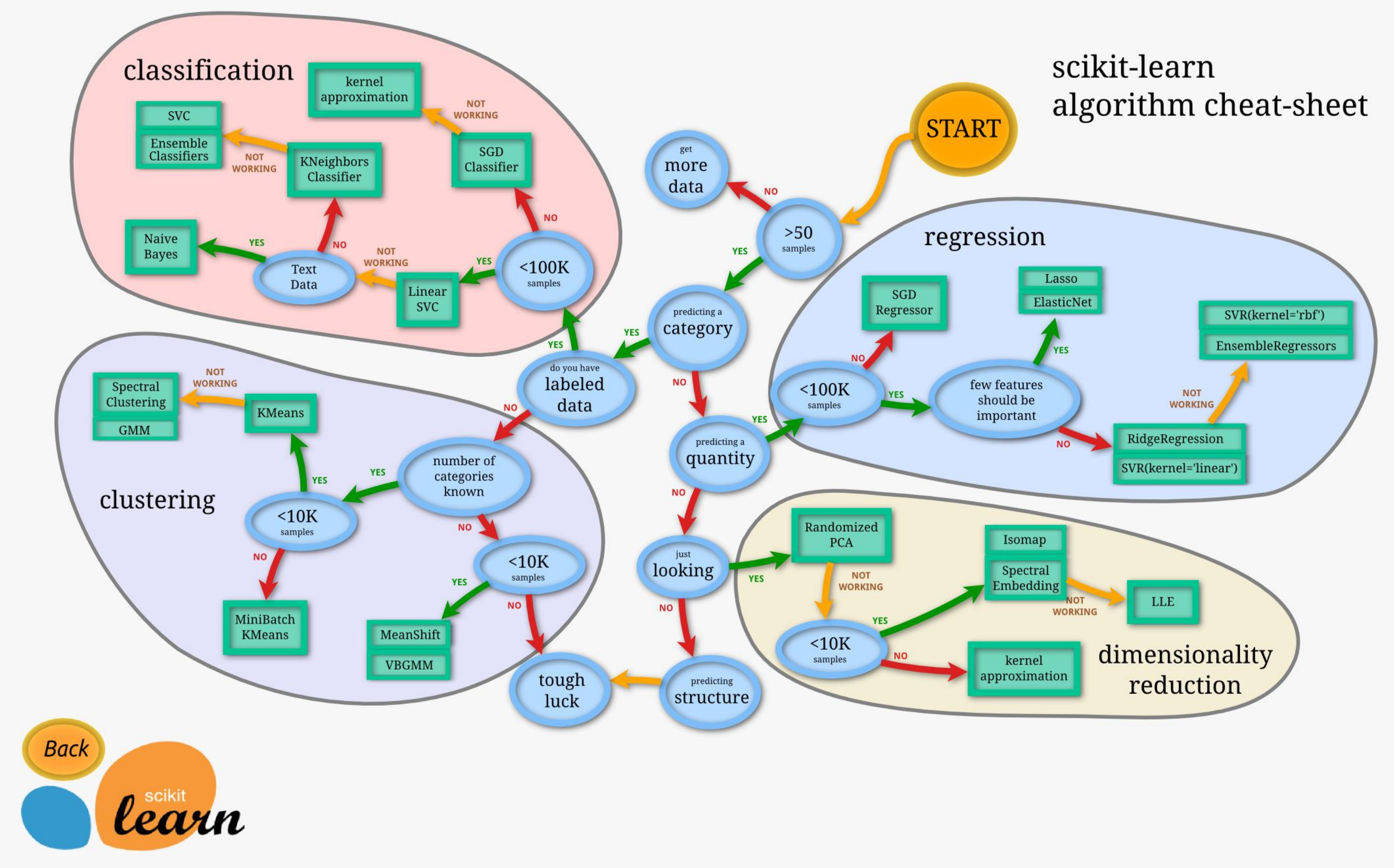
- ❖ 설치 pip install –U scikit-learn
- ❖ 데이터 모델링에 중점을 두고 라이브러리 구성

- ❖ Supervised Learning Algorithms
  - Linear Regression, Support Vector Machine(SVM), Decision Tree 등
- ❖ Unsupervised Learning Algorithms
  - clustering, factor analysis, PCA(Principal Component Analysis)

Modelling

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

- ❖ 데이터 준비
  (데이터 전처리)
- ❖ 데이터 로딩
- ❖ 데이터 분할
  (train, test;
  train, test, validation)
- ❖ 모델 선정 및 학습
  (estimator)
- ❖ 활용 (예측에 사용)



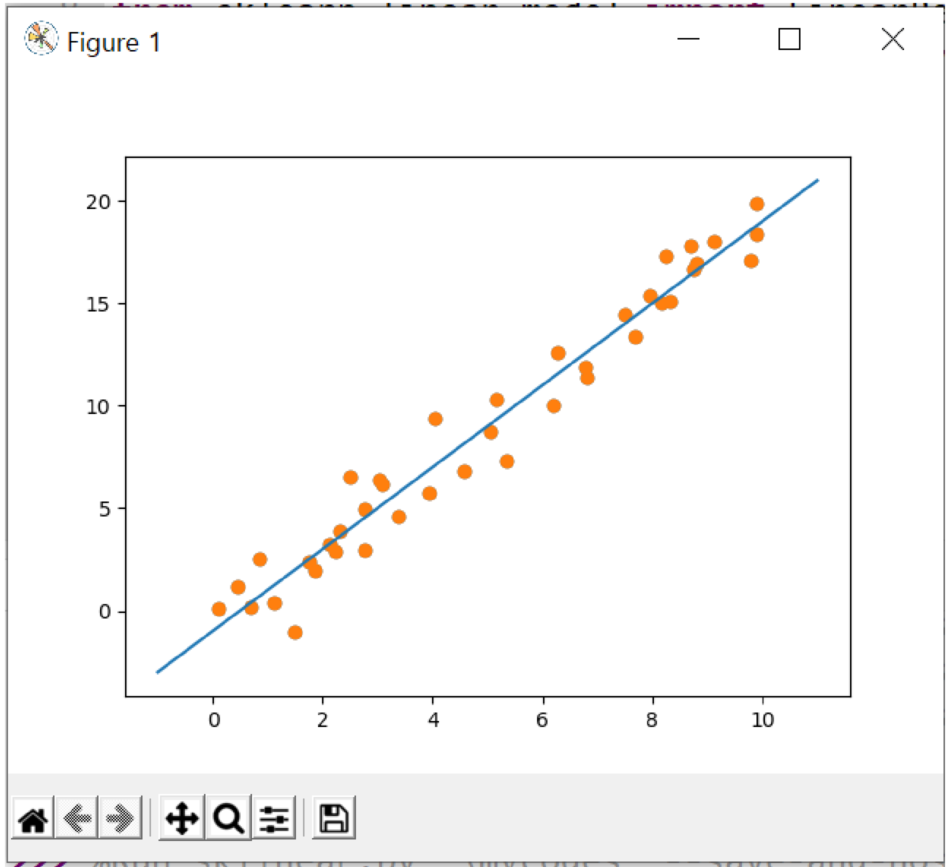
KOREATECH 한국기술교육대학교

Steps in using Estimator API

- estimator: 데이터로부터 학습하는 객체
- **Step 1: Choose a class of model**
  - It can be done by importing the appropriate Estimator class from Scikit-learn.
- **Step 2: Choose model hyperparameters**
  - It can be done by instantiating the class with desired values.
- **Step 3: Arranging the data**
  - to arrange the data into features matrix (X) and target vector(y).
- **Step 4: Model Fitting**
  - to fit the model to your data. (calling **fit()** method)
- **Step 5: Applying the model**
  - apply it to new data.
  - for supervised learning, use **predict()** method
  - for unsupervised learning, use **predict()** or **transform()**

Supervised Learning Example

❖ simple linear regression

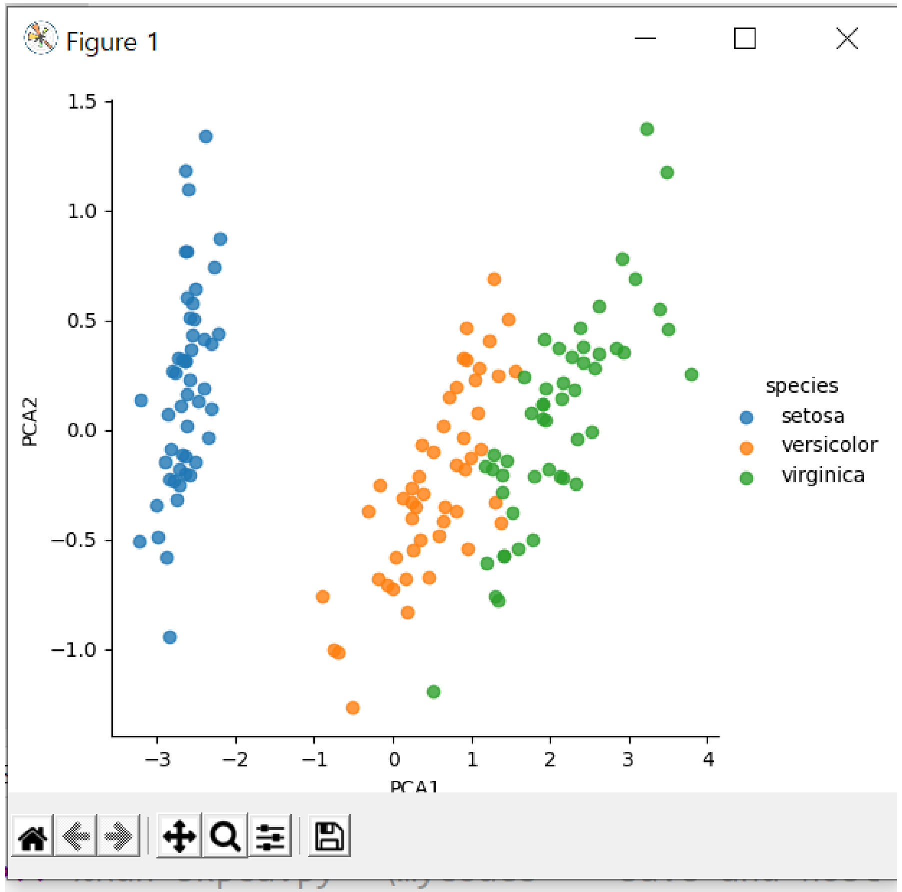



```
LinearRegression()
[1.99839352]
-0.989545945777504
```

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
rng = np.random.RandomState(35)
x = 10*rng.rand(40)
y = 2*x-1+rng.randn(40)
plt.scatter(x,y)
plt.show()
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
X = x[:, np.newaxis]
print(model.fit(X, y))
print(model.coef_)
print(model.intercept_)
xfit = np.linspace(-1, 11)
Xfit = xfit[:, np.newaxis]
yfit = model.predict(Xfit)
plt.scatter(x, y)
plt.plot(xfit, yfit)
plt.show()
```

Unsupervised Learning Example

❖ 차원축소 방법




```
(150, 4)
(150,)
PCA(n_components=2)
```

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
iris = sns.load_dataset('iris')
X_iris = iris.drop('species', axis = 1)
print(X_iris.shape)
y_iris = iris['species']
print(y_iris.shape)
from sklearn.decomposition import PCA
model = PCA(n_components=2)
print(model.fit(X_iris))
X_2D = model.transform(X_iris)
iris['PCA1'] = X_2D[:, 0]
iris['PCA2'] = X_2D[:, 1]
sns.lmplot("PCA1", "PCA2", hue='species', data=iris, fit_reg=False)
plt.show()
```

모델링과정

Modelling

❖ Dataset Loading

- Features: 입력 데이터
  - ✓ Feature matrix: It is the collection of features, in case there are more than one.
  - ✓ Feature Names: It is the list of all the names of the features.
- Response: 출력
  - ✓ Response Vector: It is used to represent response column. (We have just one response column.)
  - ✓ Target Names: It represent the possible values taken by a response vector

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names
print("Feature names:", feature_names)
print("Target names:", target_names)
print("₩nFirst 10 rows of X:₩n", X[:10])
```

```
[[5.1 3.5 1.4 0.2]
[4.9 3. 1.4 0.2]
[4.7 3.2 1.3 0.2]
[4.6 3.1 1.5 0.2]
[5. 3.6 1.4 0.2]
[5.4 3.9 1.7 0.4]
[4.6 3.4 1.4 0.3]
[5. 3.4 1.5 0.2]
[4.4 2.9 1.4 0.2]
[4.9 3.1 1.5 0.1]]
```

```
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Target names: ['setosa' 'versicolor' 'virginica']
```

Modelling

❖ 데이터 셋의 분할

- training set (70%) : testing set (30%)
- 150 * 0.7 = 105

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data     # feature matrix
y = iris.target   # response vector
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
print(X_train.shape)
print(X_test.shape)                    (105, 4)
print(y_train.shape)                   (45, 4)
print(y_test.shape)                    (105,)
                                       (45,)
```

Modeling

❖ Train the model

- scikit-learn에서 제공하는 ML 알고리즘을 활용하여 학습 (예. KNN: K nearest neighbors)

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
classifier_knn = KNeighborsClassifier(n_neighbors=3)
classifier_knn.fit(X_train, y_train)
y_pred = classifier_knn.predict(X_test)
# Finding accuracy by comparing actual response values(y_test)with predicted response value(y_pred)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
# Providing sample data and the model will make prediction out of that data
sample = [[5, 5, 3, 2], [2, 4, 3, 5]]
preds = classifier_knn.predict(sample)
pred_species = [iris.target_names[p] for p in preds]
print("Predictions:", pred_species)
```

```
Accuracy: 0.9833333333333333
Predictions: ['versicolor', 'virginica’]
```

정확도 = (올바르게 예측한 샘플수)/(전체 샘플수)
= (TP+TN)/(TP+TN+FP+FN)

Modelling (모델저장)

❖ Model Persistence

- 학습된 모델의 보관

❖ 모델 dump

```
from sklearn.externals import joblib
joblib.dump(classifier_knn, 'iris_classifier_knn.joblib')
```

❖ 저장된 모델의 load

```
joblib.load('iris_classifier_knn.joblib')
```

Modelling (전처리)

❖ 입력데이터의 전처리

- 획득한 raw data를 학습(인공지능모델)에 활용할 수 있도록 데이터 가공이 필요

❖ 이진화(Binarisation)

- 0.5 기준으로 이진화

```
import numpy as np
from sklearn import preprocessing

input_data = np.array([[2.1, -1.9, 5.5],
                       [-1.5, 2.4, 3.5],
                       [0.5, -7.9, 5.6],
                       [5.9, 2.3, -5.8]])
data_binarized = preprocessing.Binarizer(threshold=0.5).transform(input_data)
print("₩nBinarized data:₩n", data_binarized)
```

```
Binarized data:
[[ 1. 0. 1.]
[ 0. 1. 1.]
[ 0. 0. 1.]
[ 1. 1. 0.]]
```

Modelling (전처리)

❖ Mean removal

```
import numpy as np
from sklearn import preprocessing
input_data = np.array([[2.1, -1.9, 5.5],
                       [-1.5, 2.4, 3.5],
                       [0.5, -7.9, 5.6],
                       [5.9, 2.3, -5.8]])
#displaying the mean and the standard deviation of the input data
print("Mean =", input_data.mean(axis=0)) # 세로축
print("Stddeviation = ", input_data.std(axis=0))
#Removing the mean and the standard deviation of the input data
data_scaled = preprocessing.scale(input_data)
print(data_scaled)
print("Mean_removed =", data_scaled.mean(axis=0))
print("Stddeviation_removed =", data_scaled.std(axis=0))
```

```
Mean = [ 1.75   -1.275   2.2   ]
Stddeviation =  [2.71431391 4.20022321 4.69414529]
[[ 0.12894603 -0.14880162  0.70300338]
 [-1.19735598  0.8749535   0.27694073]
 [-0.46052153 -1.57729713  0.72430651]
 [ 1.52893149  0.85114524 -1.70425062]]
Mean_removed = [1.11022302e-16 0.00000000e+00 0.00000000e+00]
Stddeviation_removed = [1. 1. 1.]
```

Modelling (전처리)

❖ Scaling (0 ~ 1 사이 값으로 정리)

```
import numpy as np
from sklearn import preprocessing
input_data = np.array([[2.1, -1.9, 5.5],
                       [-1.5, 2.4, 3.5],
                       [0.5, -7.9, 5.6],
                       [5.9, 2.3, -5.8]])
data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0,1))
data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
print ("₩nMin max scaled data:₩n", data_scaled_minmax)   # 컬럼기준으로 해석
```

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

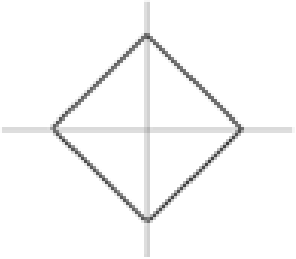
```
Min max scaled data:
[[ 0.48648649  0.58252427   0.99122807]
 [ 0.          1.           0.81578947]
 [ 0.27027027  0.           1.        ]
 [ 1.          0.99029126   0.        ]]
```

Modelling (전처리)

❖ Normalizaiton (정규화)

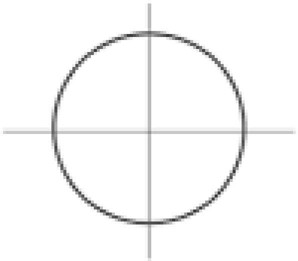
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



```
import numpy as np
from sklearn import preprocessing
input_data = np.array([[2.1, -1.9, 5.5],
                       [-1.5, 2.4, 3.5],
                       [0.5, -7.9, 5.6],
                       [5.9, 2.3, -5.8]])
data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
print("₩nL1 normalized data:₩n", data_normalized_l1)    # 가로축을 기준으로 값을 정렬
data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')
print("₩nL1 normalized data:₩n", data_normalized_l2)    # 예) l1 = 2.1/(2.1+1.9+5.5)
                                                        #     l2 = 2.1/sqrt(2.1*2.1 + 1.9*1.9 + 5.5*5.5)
```

```
L1 normalized data:
[[ 0.22105263 -0.2 0.57894737]
[-0.2027027 0.32432432 0.47297297]
[ 0.03571429 -0.56428571 0.4 ]
[ 0.42142857 0.16428571 -0.41428571]]
```

```
L2 normalized data:
[[ 0.33946114 -0.30713151 0.88906489]
[-0.33325106 0.53320169 0.7775858 ]
[ 0.05156558 -0.81473612 0.57753446]
[ 0.68706914 0.26784051 -0.6754239 ]]
```

MNIST (숫자인식)

MNIST

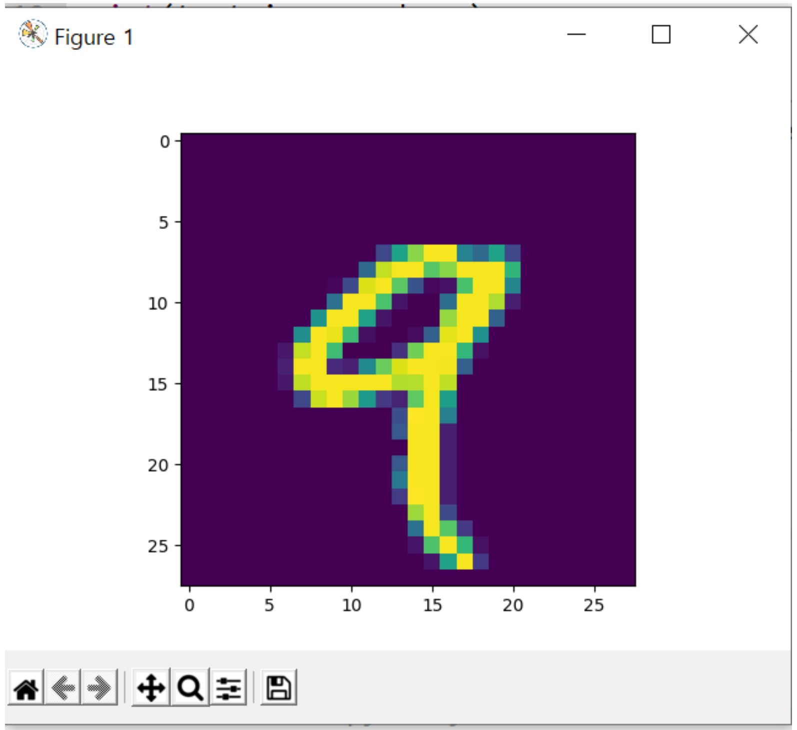
❖ The MNIST dataset is a well-known dataset consisting of 28x28 grayscale images. For each image, we know the corresponding digits (from 0 to 9). It is available here: http://yann.lecun.com/exdb/mnist/index.html

```
import mnist      # need to install
import matplotlib.pyplot as plt

# Load dataset
train_images = mnist.train_images()
train_labels = mnist.train_labels()
test_images = mnist.test_images()
test_labels = mnist.test_labels()
print(train_images.shape)
print(test_images.shape)

# Pick the fifth image from the dataset (it's a 9)
image, label = train_images[4], train_labels[4]
print(label)

plt.imshow(image)
plt.show()
```




```
(60000, 28, 28)
(10000, 28, 28)
9
```

MNIST

❖ KNN

```
import mnist
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
# Load dataset
train_images = mnist.train_images()
train_labels = mnist.train_labels()
test_images = mnist.test_images()
test_labels = mnist.test_labels()
# preprocessing
train_images = train_images.reshape(-1, 28*28)
test_images = test_images.reshape(-1, 28*28)

clf = KNeighborsClassifier()
#clf.fit(train_images, train_labels)
clf.fit(train_images[:10000], train_labels[:10000])

# Test on the next 100 images:
test_x = test_images[:100]
expected = test_labels[:100].tolist()
print("Compute predictions")
predicted = clf.predict(test_x)
print("Accuracy: ", accuracy_score(expected, predicted))
```

MNIST

❖ Random Forest

```
import mnist
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# Load dataset
train_images = mnist.train_images()
train_labels = mnist.train_labels()
test_images = mnist.test_images()
test_labels = mnist.test_labels()
# preprocessing
train_images = train_images.reshape(-1, 28*28)
test_images = test_images.reshape(-1, 28*28)

clf = RandomForestClassifier(n_estimators=100)
clf.fit(train_images[:10000], train_labels[:10000])
# Test on the next 1000 images:
test_x = train_images[10000:11000]
expected = train_labels[10000:11000].tolist()
print("Compute predictions")
predicted = clf.predict(test_x)
print("Accuracy: ", accuracy_score(expected, predicted))
```

MNIST

- Linear Support Vector Classification

```
import mnist
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
# Load dataset
train_images = mnist.train_images()
train_labels = mnist.train_labels()
test_images = mnist.test_images()
test_labels = mnist.test_labels()
# preprocessing
train_images = train_images.reshape(-1, 28*28)
test_images = test_images.reshape(-1, 28*28)

clf = LinearSVC()
clf.fit(train_images[:10000], train_labels[:10000])
# Test on the next 1000 images:
test_x = train_images[10000:11000]
expected = train_labels[10000:11000].tolist()
print("Compute predictions")
predicted = clf.predict(test_x)
print("Accuracy: ", accuracy_score(expected, predicted))
```

베이즈정리

베이즈정리

- ❖ 확률을 지식 또는 믿음의 정도를 나타내는 양이라는 관점에서 접근
- ❖ 주어진 데이터를 바탕으로 확률을 변경할 수 있다.
- ❖ 베이즈 정리 $$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$
  - P(A) 사전확률(prior)로써 사건 B가 발생하기 전에 사건 A의 확률
  - P(A|B) 사후확률(posterior): 사건 B가 발생하여 갱신된 사건 A의 확률
  - P(B|A) 가능도(likelihood; 우도), 사건 A가 발생한 경우 사건 B의 확률
  - P(B) 증거(evidence)
- ❖ 확장

$$\begin{aligned} P(A_1|B) &= \frac{P(B|A_1)P(A_1)}{P(B)} \\ &= \frac{P(B|A_1)P(A_1)}{\sum_i P(A_i, B)} \qquad A_i \cap A_j = \emptyset \\ &= \frac{P(B|A_1)P(A_1)}{\sum_i P(B|A_i)P(A_i)} \qquad A_1 \cup A_2 \cup \cdots = \Omega \end{aligned}$$

베이즈 정리

- ❖ 검사 시약 문제: 특정 질병을 검사하는 시약으로 특정 질병에 걸리 환자를 대상으로 시약 검사를 하면 99%의 확률로 양성반응을 보인다.
- ❖ 양성반응을 보였지만 실제 병에 걸렸을 확률은?

  - 병에 걸리는 경우 : 사건 D
  - 양성 반응을 보이는 경우 : 사건 S
  - 병에 걸린 사람이 양성 반응을 보이는 경우 : 조건부 사건 $S|D$
  - 양성 반응을 보이는 사람이 병에 걸려 있을 경우 : 조건부 사건 $D|S$
  - 문제
  - $P(S|D) = 0.99$가 주어졌을 때, $P(D|S)$를 구하라.

$$P(D|S) = \frac{P(S|D)P(D)}{P(S)}$$

KOREATECH 한국기술교육대학교

베이즈정리

❖ 베이즈 정리를 이용하려면 추가 정보가 필요하다.

- 특정질병은 희귀병으로 전체인구의 0.2%가만 걸렸다. P(D) = 0.002
- 시약검사를 했을때 잘못된 양성반응이 나오는 확률이 5%이다. P(S|(1-D)) = 0.05

$$
\begin{aligned}
P(D|S) &= \frac{P(S|D)P(D)}{P(S)} \\
&= \frac{P(S|D)P(D)}{P(S,D) + P(S,D^C)} \\
&= \frac{P(S|D)P(D)}{P(S|D)P(D) + P(S|D^C)P(D^C)} \\
&= \frac{P(S|D)P(D)}{P(S|D)P(D) + P(S|D^C)(1 - P(D))} \\
&= \frac{0.99 \cdot 0.002}{0.99 \cdot 0.002 + 0.05 \cdot (1 - 0.002)} \\
&= 0.038
\end{aligned}
$$

베이즈정리확장

$$P(A|B,C) = \frac{P(B|A,C)P(A|C)}{P(B|C)}$$

$$P(A|B,C,D) = \frac{P(D|A,B,C)P(A|B,C)}{P(D|B,C)}$$

$$P(A,B|C,D) = \frac{P(D|A,B,C)P(A,B|C)}{P(D|C)}$$

베이즈분류모형

- ❖ 주어진 데이터를 가지고 가능도를 추정 $P(x \mid y=k) = P(x_1, \ldots, x_D \mid y=k)$
  - ▪ 입력데이터의 차원이 높아지면 가능도 추정이 어려워짐
- ❖ 모든 차원의 개별 독립변수가 서로 조건부 독립이라면 (navie assumption)

$$P(x_1, \ldots, x_D \mid y=k) = \prod_{d=1}^{D} P(x_d \mid y=k)$$

$$P(y=k \mid x) = \frac{P(x_1, \ldots, x_D \mid y=k) P(y=k)}{P(x)}$$

$$= \frac{\left(\prod_{d=1}^{D} P(x_d \mid y=k)\right) P(y=k)}{P(x)}$$

- ❖ 가능도를 정규분포로 가정 $P(x_d \mid y=k) = \frac{1}{\sqrt{2\pi\sigma_{d,k}^2}} \exp\left(-\frac{(x_d - \mu_{d,k})^2}{2\sigma_{d,k}^2}\right)$

KOREATECH 한국기술교육대학교

나이브베이즈 모형

❖ 정규분포 나이브베이즈 모형

- 주어진 데이터가 정규분포라고 생각하고, 데이터에 기반한 평균, 분산을 추정

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
iris = load_iris()
X1 = iris.data
y1 = iris.target

from sklearn.naive_bayes import GaussianNB
model1 = GaussianNB().fit(X1, y1)
print(model1.class_prior_)
print(model1.theta_)
print(model1.sigma_)
y1_pred = model1.predict(X1)

from sklearn.metrics import confusion_matrix
print(confusion_matrix(y1, y1_pred))
from sklearn.metrics import classification_report
print(classification_report(y1, y1_pred))
```

학습정리

- 기계학습
- 구글티쳐블머신
  - no coding
- scikit learn (sklearn)
  - 기계학습패키지
  - MNIST 활용