Machine Learning Concepts

Machine Learning

What is Machine Learning?

- Definition
  - *Herbert A. Simon*

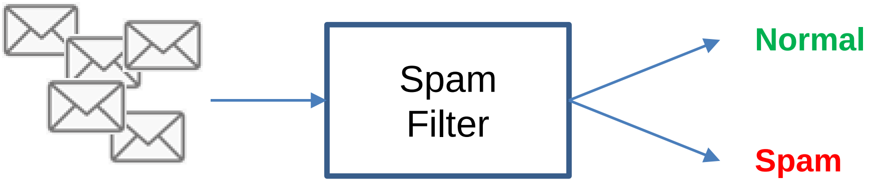
    " Learning is any process by which a system improves performance from experience "
  - *Tom M. Mitchell*

    "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*"
  - *Wikipedia*

    "The development and study of statistical algorithms that can *learn from data and generalize to unseen data*, and thus *perform tasks without explicit instructions*"

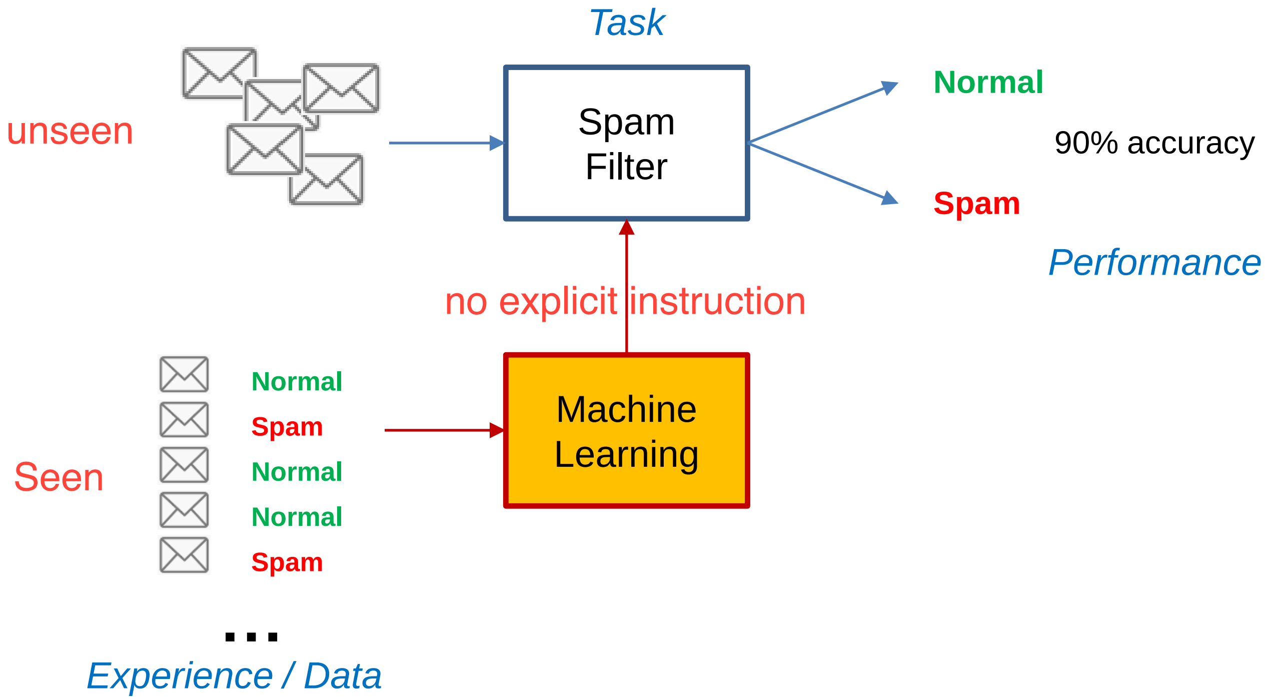
What is Machine Learning?

- Example – spam filtering program
  - Task T
    - Identify spam email and automatically remove them
  - Performance P
    - Accuracy (% of correct identification)
  - Experience E
    - Email + user feedback (spam / normal)



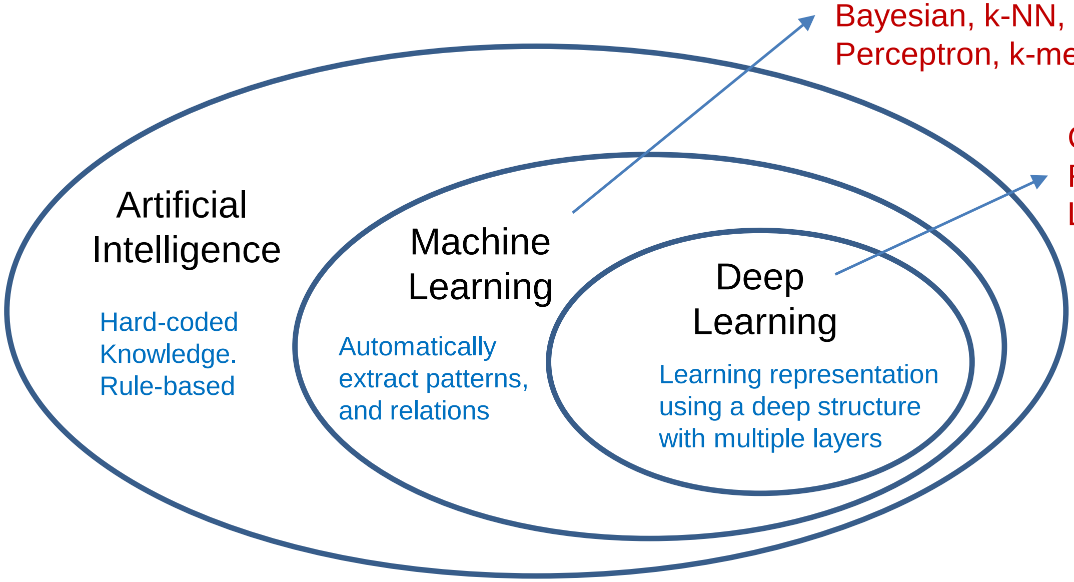
What is Machine Learning?

- Example – spam filtering program
  - Machine Learning algorithm
    - Improves performance from experience



What is Machine Learning?

- ML Models/Algorithms
  - AI – Machine Learning – Deep Learning


Artificial Intelligence

Hard-coded Knowledge. Rule-based

Machine Learning

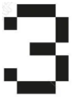
Automatically extract patterns, and relations

Deep Learning

Learning representation using a deep structure with multiple layers

Regression, decision trees,
Bayesian, k-NN, SVM,
Perceptron, k-means, …

Convolutional Neural Network,
Recurrent Neural Network,
LSTM, GAN, …


→ ?

H: if (-,-,-)
then 3

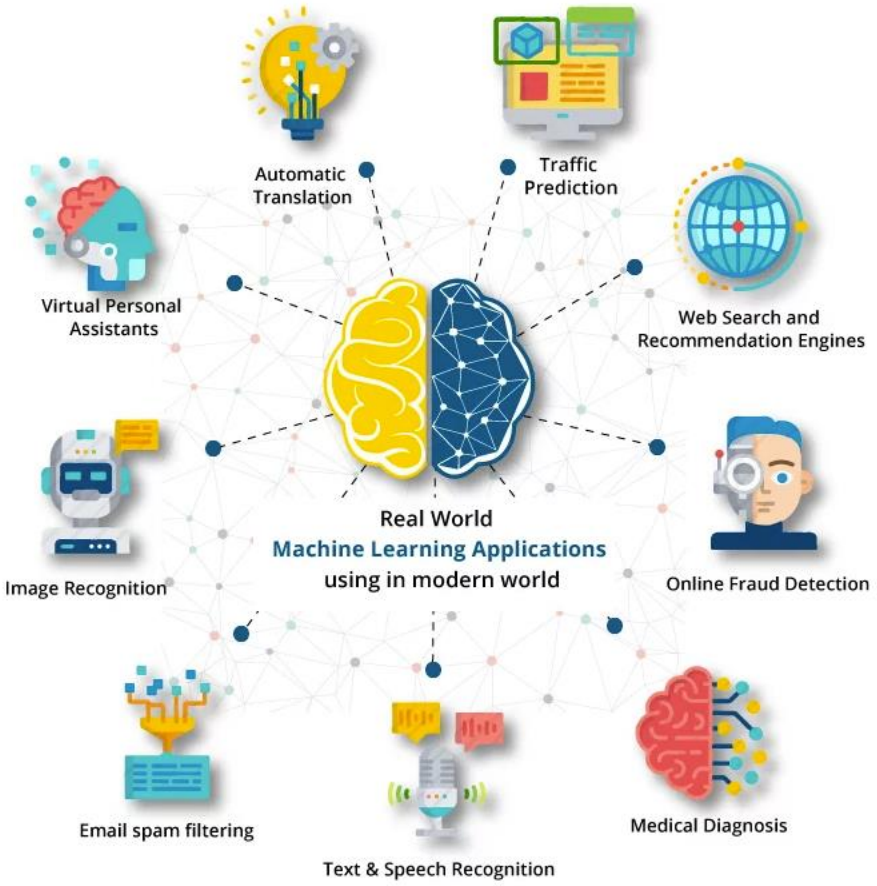
H: detect -, |
C: if (-,-,-)
then 3

C: detect -, |, x, /, □
C: if (-,-,-)
then 3

https://en.wikipedia.org/wiki/Machine_learning

What is Machine Learning?

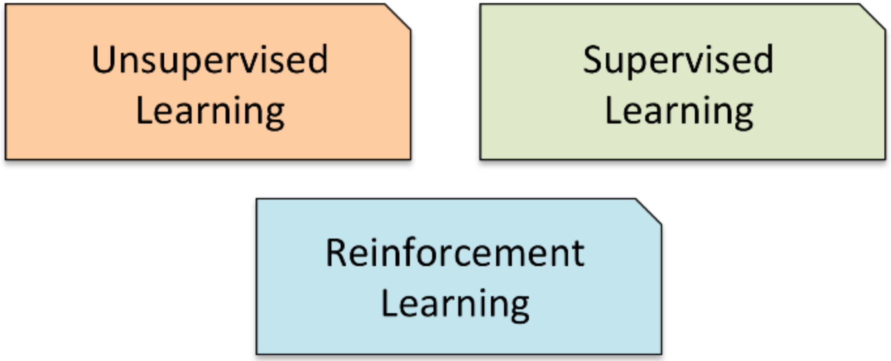
- ML Applications



https://rampavanphd2018.blogspot.com/2019/06/machine-learning-in-real-world.html

Types of Machine Learning

- Types
  - Supervised learning (지도 학습)
  - Unsupervised learning (비지도 학습)
  - Reinforcement learning (강화 학습)



Using unlabeled data
Learn hidden structure

Unsupervised Learning

Supervised Learning

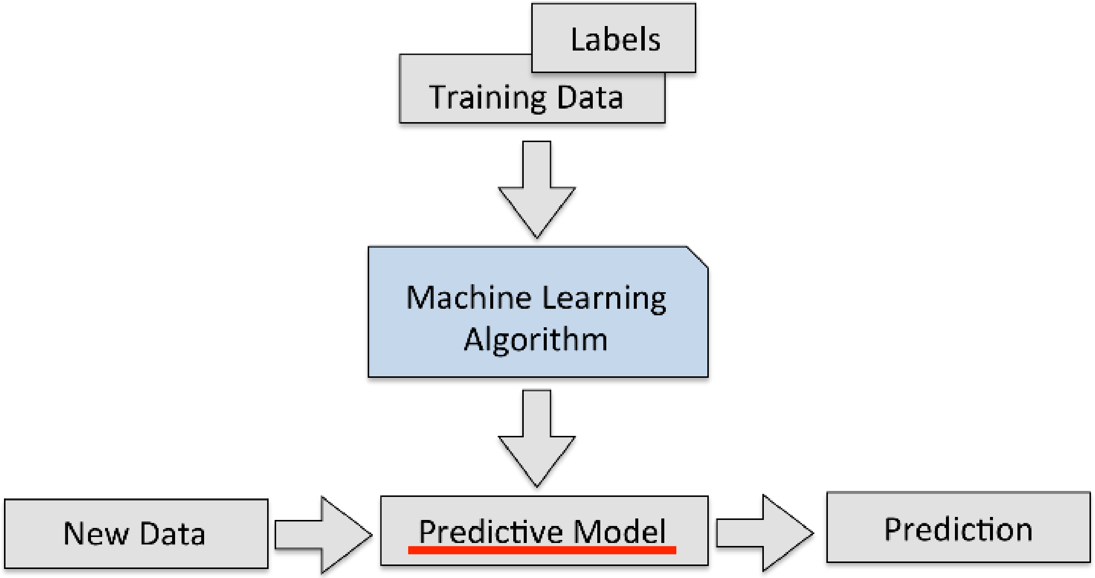
Using labeled data
Learn a model to predict label

Reinforcement Learning

Using reward from actions
Learn action policy

Supervised Learning

- Learning a function that maps an input x to an output y based on example input-output pairs (training data)
  - Given: **<x, y>** examples
  - Learning: build **<y = f(x)> (model)** to give the right answer(y) for new data(x)

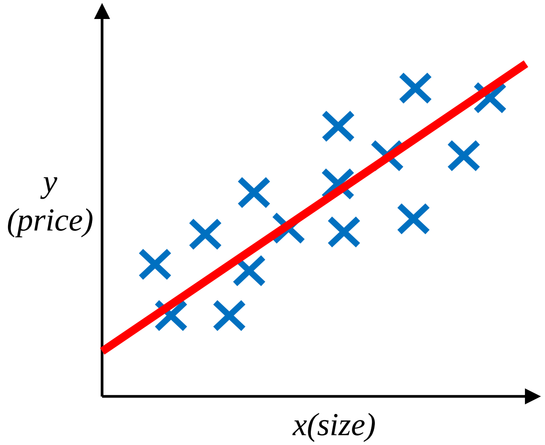


Supervised Learning

- Regression : predicting target values
  - From x (house size) → predict y (house price)

x		target value y
size		price
2104		400,000
1600		330,000
2400		369,000
3000		540,000
…	…	…

model



1800 → price ?

input x (size) → $\boxed{y = f(x)}$ → output y (price)

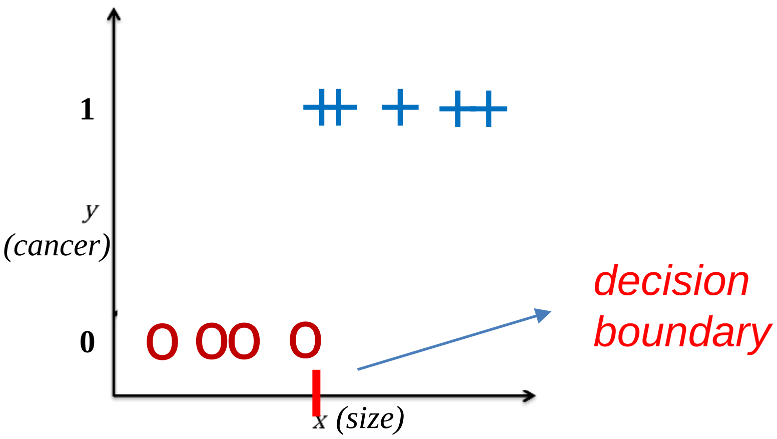
Supervised Learning

- Classification : predicting class labels
  - From x (cell sze) → predict class label y (cancer): {Yes, No}

x size		class label y cancer
0.75		1 (yes)
0.82		1 (yes)
0.26		0 (no)
0.54		0 (no)
…	…	…

0.43 → cancer ?

model

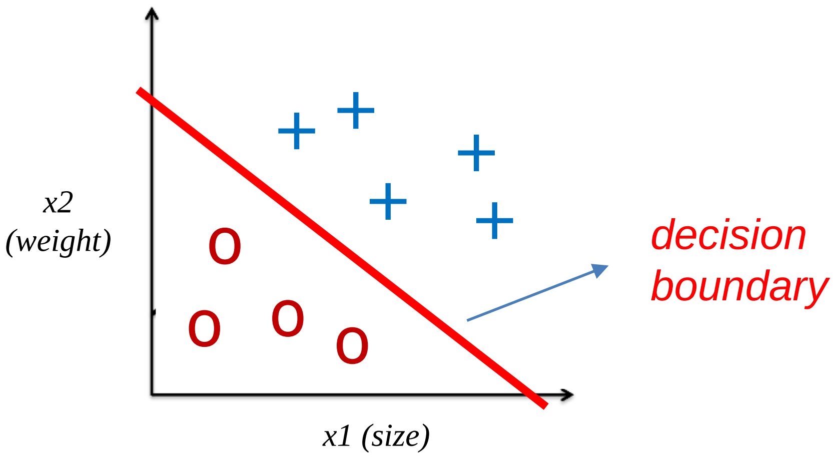


input x (size) → x > a ? → output y (0/1)

Supervised Learning

- Classification
  - From x1, x2 (cell sze and weight) → predict class label y (cancer): {Yes, No}

x1	x2	class label y
size	weight	cancer
0.75	0.9	1 (yes)
0.82	0.8	1 (yes)
0.26	0.4	0 (no)
0.54	0.7	0 (no)
…	…	…

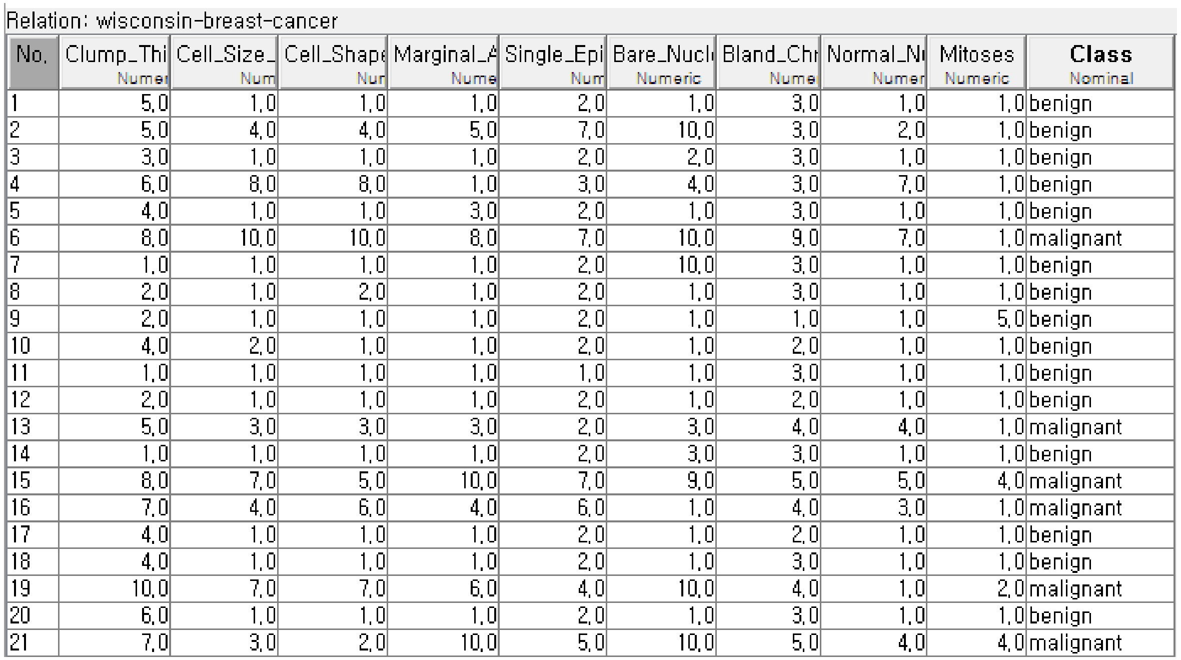


input x_1, x_2 (size, weight) → $\boxed{a \cdot x_1 + b \cdot x_2 + c > 0\ ?}$ → output y (Yes/No)

Supervised Learning

- Classification
  - From x1 ~ x9 (patient attributes) → predict class label y: {benign, malignant}

label



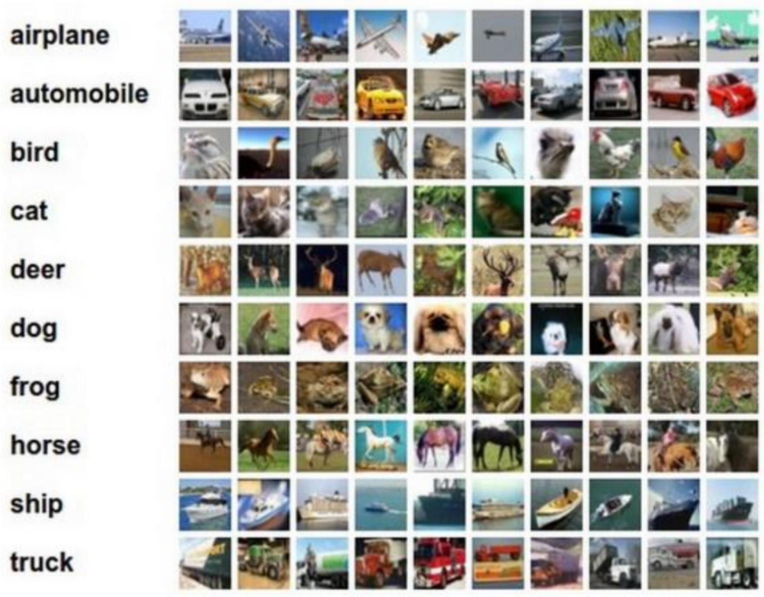
Relation: wisconsin-breast-cancer

No.	Clump_Thi	Cell_Size_	Cell_Shape	Marginal_A	Single_Epi	Bare_Nucl	Bland_Chr	Normal_Nu	Mitoses	Class
	Numer	Num	Nur	Nume	Num	Numeric	Nume	Numer	Numeric	Nominal
1	5.0	1.0	1.0	1.0	2.0	1.0	3.0	1.0	1.0	benign
2	5.0	4.0	4.0	5.0	7.0	10.0	3.0	2.0	1.0	benign
3	3.0	1.0	1.0	1.0	2.0	2.0	3.0	1.0	1.0	benign
4	6.0	8.0	8.0	1.0	3.0	4.0	3.0	7.0	1.0	benign
5	4.0	1.0	1.0	3.0	2.0	1.0	3.0	1.0	1.0	benign
6	8.0	10.0	10.0	8.0	7.0	10.0	9.0	7.0	1.0	malignant
7	1.0	1.0	1.0	1.0	2.0	10.0	3.0	1.0	1.0	benign
8	2.0	1.0	2.0	1.0	2.0	1.0	3.0	1.0	1.0	benign
9	2.0	1.0	1.0	1.0	2.0	1.0	1.0	1.0	5.0	benign
10	4.0	2.0	1.0	1.0	2.0	1.0	2.0	1.0	1.0	benign
11	1.0	1.0	1.0	1.0	1.0	1.0	3.0	1.0	1.0	benign
12	2.0	1.0	1.0	1.0	2.0	1.0	2.0	1.0	1.0	benign
13	5.0	3.0	3.0	3.0	2.0	3.0	4.0	4.0	1.0	malignant
14	1.0	1.0	1.0	1.0	2.0	3.0	3.0	1.0	1.0	benign
15	8.0	7.0	5.0	10.0	7.0	9.0	5.0	5.0	4.0	malignant
16	7.0	4.0	6.0	4.0	6.0	1.0	4.0	3.0	1.0	malignant
17	4.0	1.0	1.0	1.0	2.0	1.0	2.0	1.0	1.0	benign
18	4.0	1.0	1.0	1.0	2.0	1.0	3.0	1.0	1.0	benign
19	10.0	7.0	7.0	6.0	4.0	10.0	4.0	1.0	2.0	malignant
20	6.0	1.0	1.0	1.0	2.0	1.0	3.0	1.0	1.0	benign
21	7.0	3.0	2.0	10.0	5.0	10.0	5.0	4.0	4.0	malignant

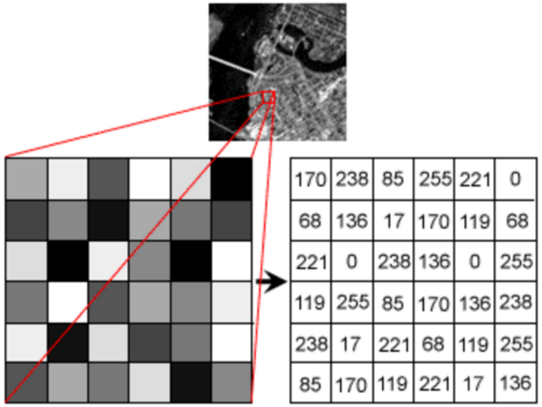
<5.0, 2.0, 3.0, 4.0, 2.0, 7.0, 3.0, 6.0, 1.0> → Model → benign/malignant?

Supervised Learning

- Classification
  - From X (image data) → predict class label y: {airplane, automobile, bird, cat, … }



label



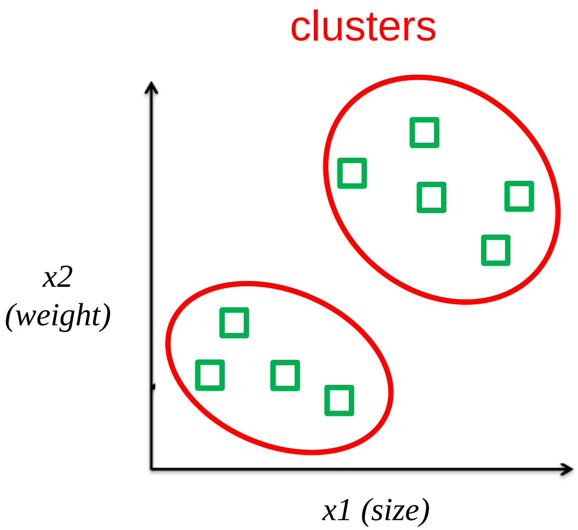
170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

→ Model → airplane/bird/cat/dog ··· ?

Unsupervised Learning

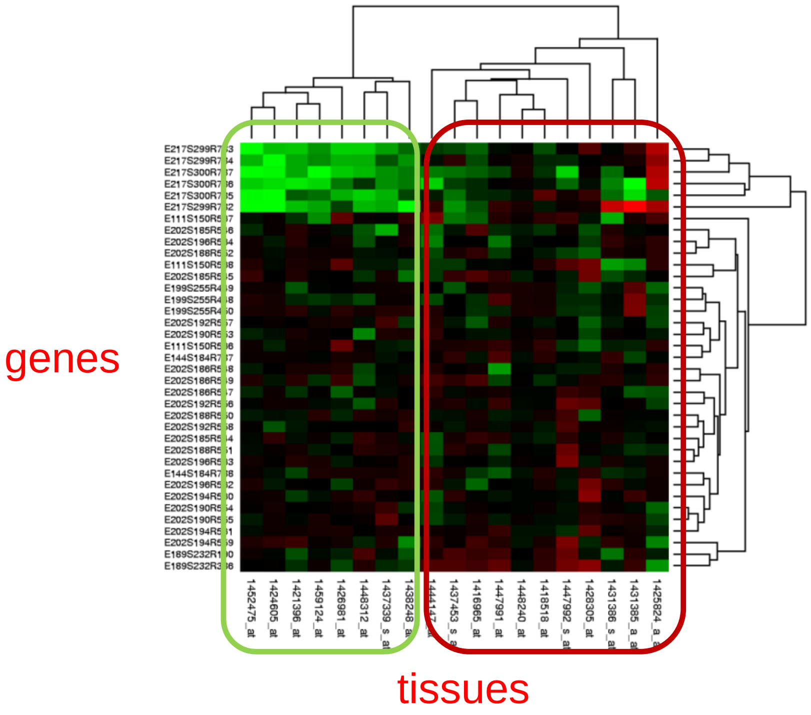
- Learning hidden structures(patterns) from unlabeled data
- Clustering
  - Given: **<x>** examples of cells
  - Learning: **<clusters>** of data

x1	x2	~~class label y~~
size	weight	~~cancer~~
0.75	0.9	~~1 (yes)~~
0.82	0.8	~~1 (yes)~~
0.26	0.4	~~0 (no)~~
0.54	0.7	~~0 (no)~~
…	…	~~…~~



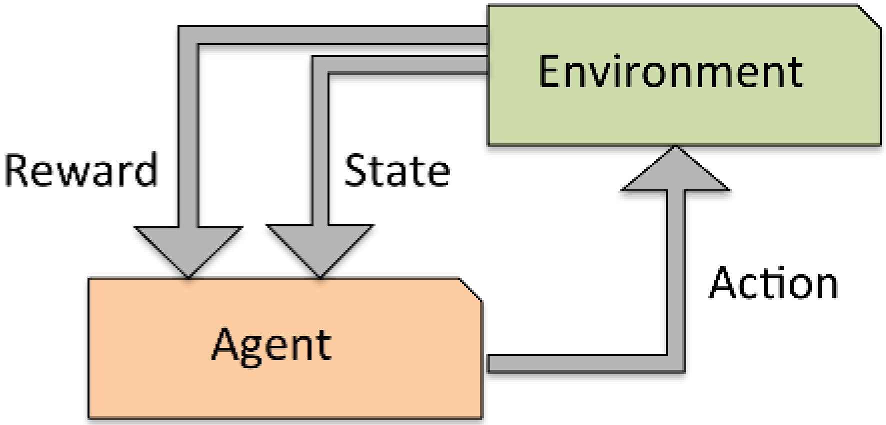
Unsupervised Learning

- Example : Microarray expression data → Find hierarchical clusters
  - Row represents an individual gene, Column represents a tissue sample
  - Each cell in the matrix represents the expression level - red is high



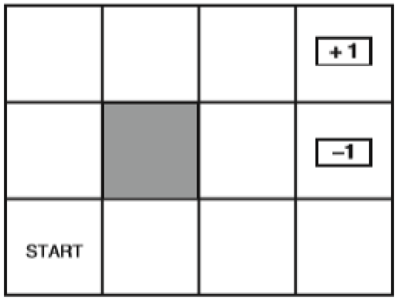
Reinforcement Learning

- learning how to take actions in an environment in order to maximize the cumulative reward
  - Given: **<action, reward>** examples
  - Learning: **<f(state)=action> (rules)** for right action



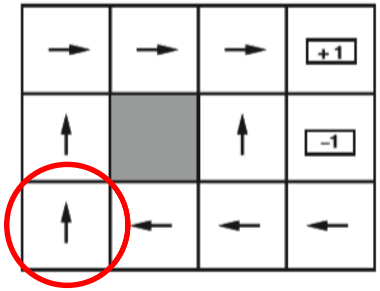
Reinforcement Learning

- Example: Find best action for each location



action	reward
→ → ↑ ↑ →	: +1
→ → ↑ →	: -1
↑ ↑ → → →	: +1
	…

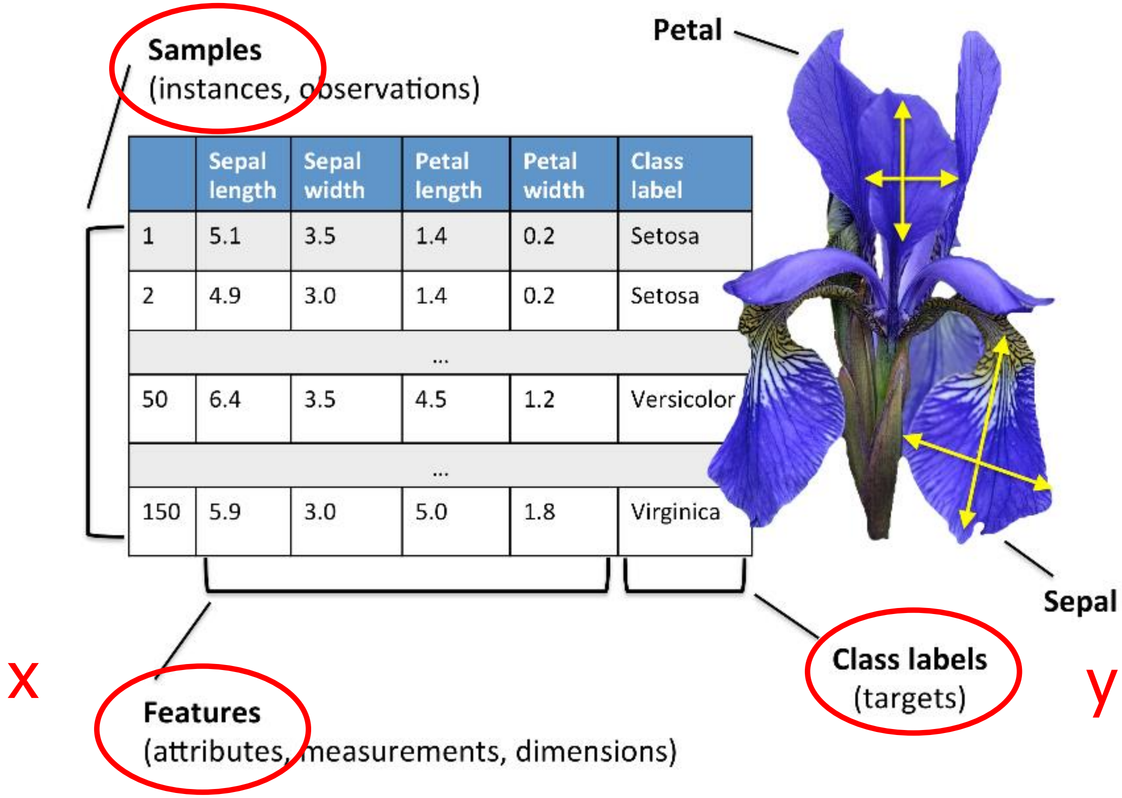
rules for action



f(s0) = Up

Basic Terminology and Notations

- Example: Classificaiton of ‘iris’ to one of 3 classes
  - Data instance : feature values + label
  - No. of features : dimension



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Samples
(instances, observations)
Petal
Sepal
x
Features
(attributes, measurements, dimensions)
Class labels
(targets)
y

Basic Terminology and Notations

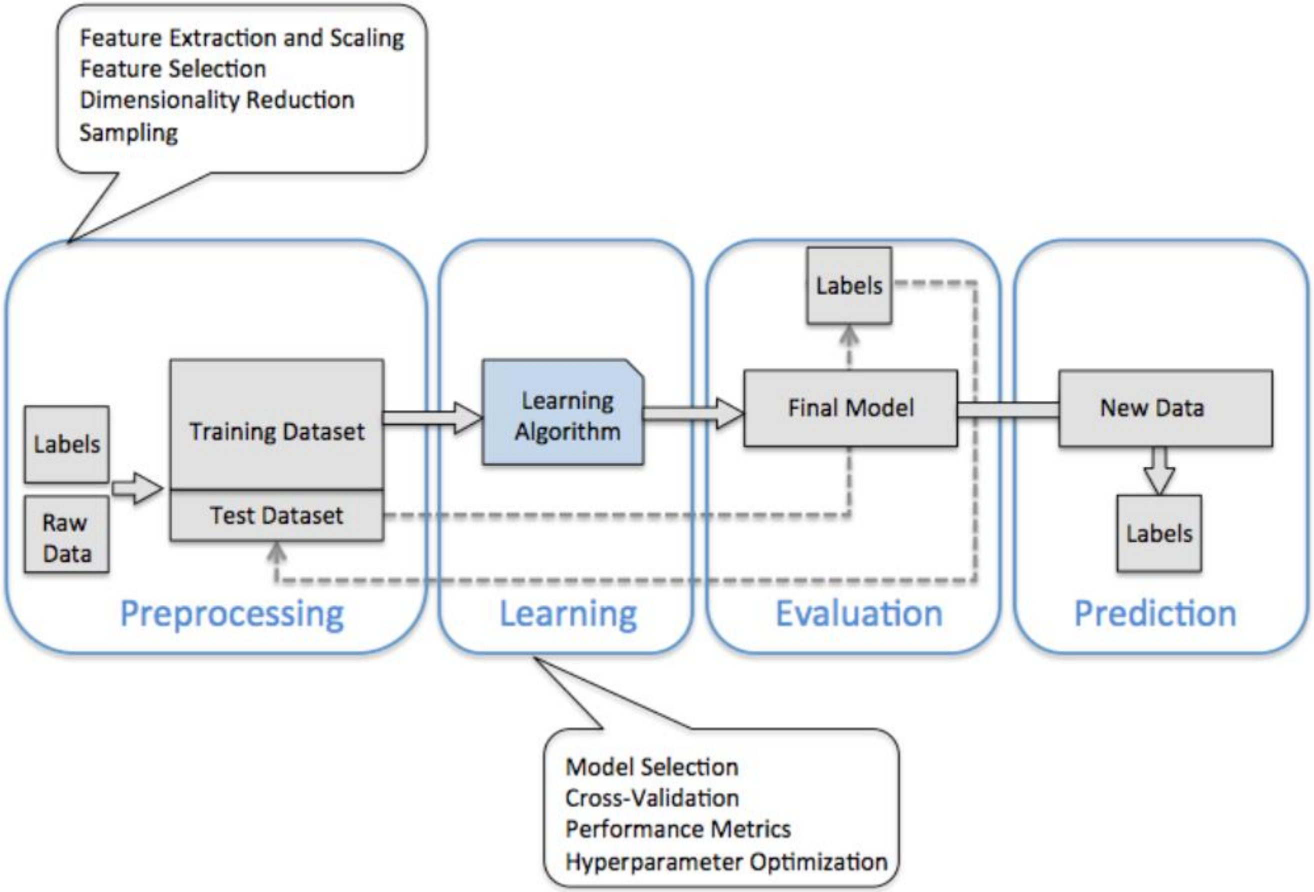
- Iris dataset
  - Features: sepal length, sepal width, petal leangth, petal width
  - Labels: {Setosa, Versicolor, Virginica}
  - Dataset size: 150 instances
- A data x = (x_1, x_2, x_3, x_4) , label = y
- Dataset X, target labels y, predicted labels ŷ

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \mathbf{x}^{(3)} \\ \cdots \\ \\ \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & x_4^{(3)} \\ & & \cdots & \\ \\ \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \cdots \\ \\ \end{bmatrix}$$

Building Machine Learning Systems

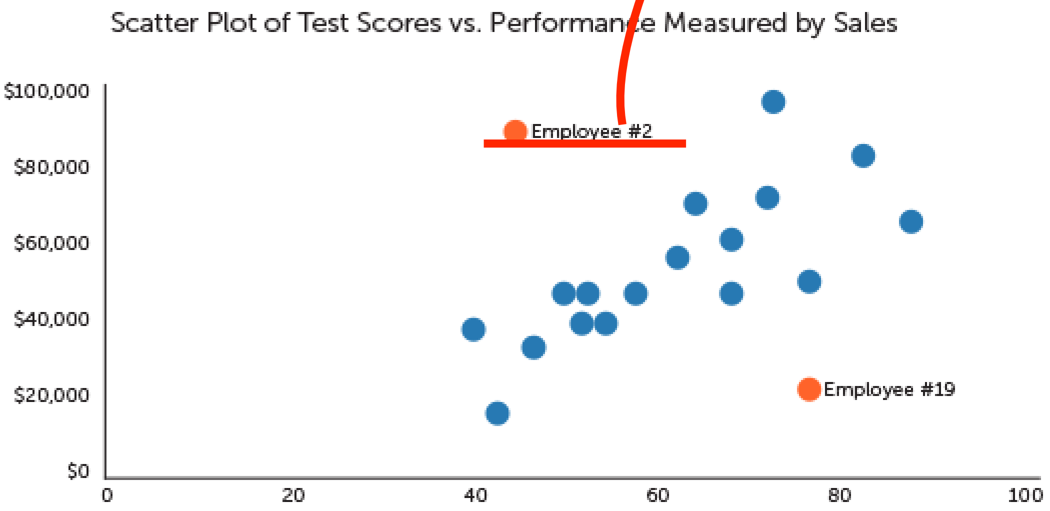
- Preprocessing → Learning (training dataset)

  → Evaluation (test dataset) → Prediction



Data Preprocessing

- Data cleaning
  - Fill in missing values
  - Handling noisy data, identify or remove outliers



Data Preprocessing

- Data transformation
  - Normalization : rescaling the feature values into a range of [0,1]
    - Make all the features contribute equally to the model
  - Standardization : rescaling the feature values to have mean of 0 and a standard deviation of 1 (unit variance)

Age	Salary
25	2000000
35	2500000
50	4000000

→

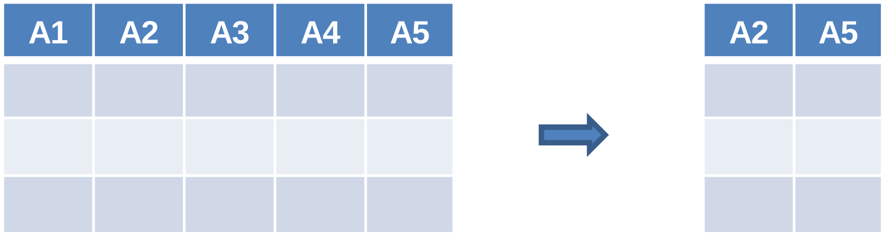
Age	Salary
-0.93	-0.80
-0.13	-0.32
1.06	1.12

  - Discretization : transferring continuous values into discrete labels

(18, 27, 63, 32, 48, …) → (Y, Y, O, Y, O, …)

Data Preprocessing

- Feature selection
  - The process of selecting relevant features for use in model construction
    - Enhanced generalization by reducing overfitting
    - Simplification of models → easier to interpret
    - Shorter training time
    - Avoid the curse of dimensionality



A1	A2	A3	A4	A5

A2	A5

Data Preprocessing

- Dimensionality reduction
  - Transform data from a high-dimensional space into a low-dimensional space
  - Principle Component Analysis (PCA)
    - Convert data into linearly uncorrelated variables - **principal components**
    - The first principal component has the largest possible variance

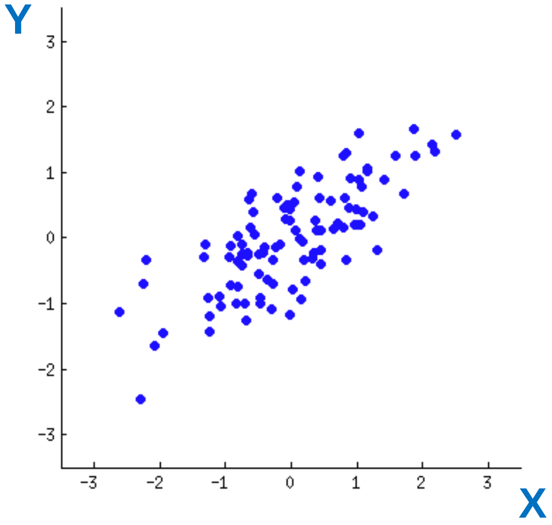
A1	A2	A3	A4

→

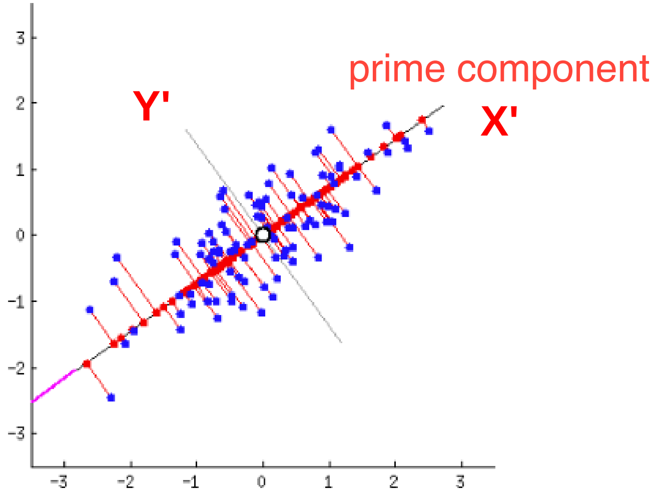
P1	P2

Data Preprocessing

- Principal components



2D data on X, Y

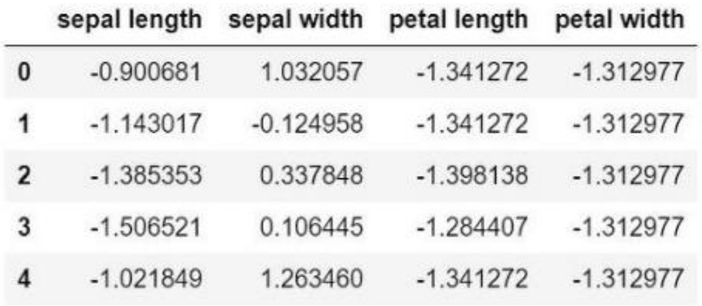


2D data on PC - X', Y'

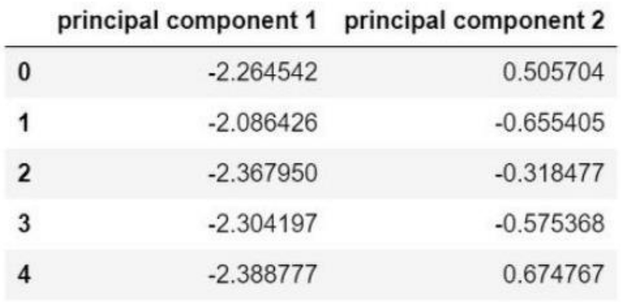
data on X' → 1D

Data Preprocessing

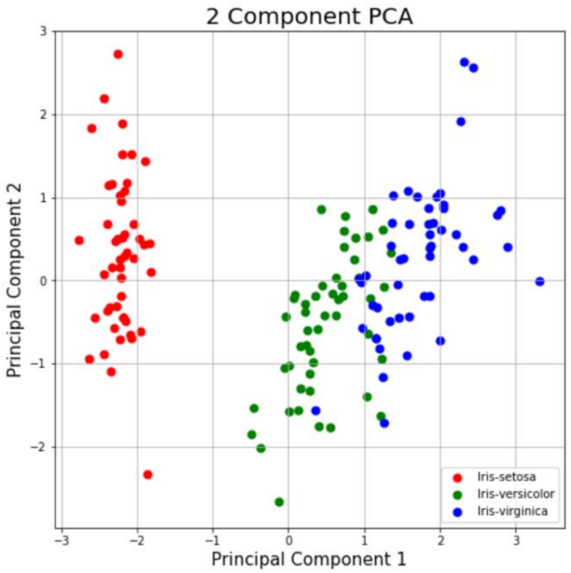
- Example: Apply PCA to iris data to reduce it to 2D



	sepal length	sepal width	petal length	petal width
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.398138	-1.312977
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977

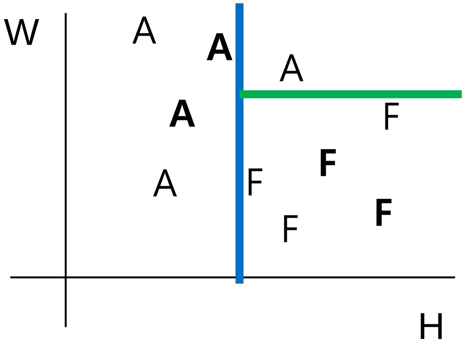


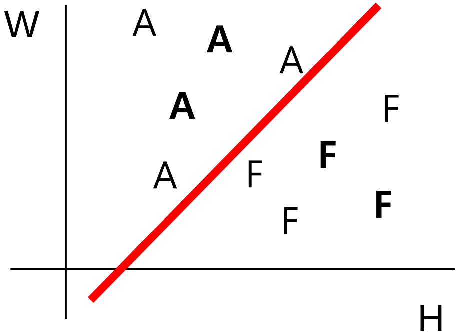
	principal component 1	principal component 2
0	-2.264542	0.505704
1	-2.086426	-0.655405
2	-2.367950	-0.318477
3	-2.304197	-0.575368
4	-2.388777	0.674767

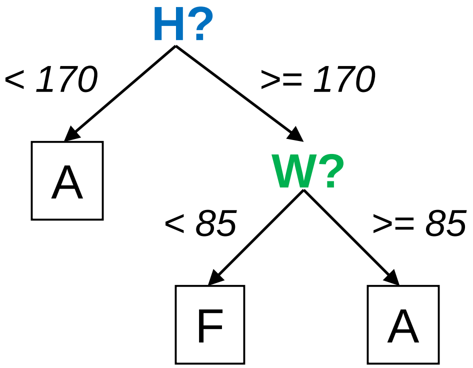


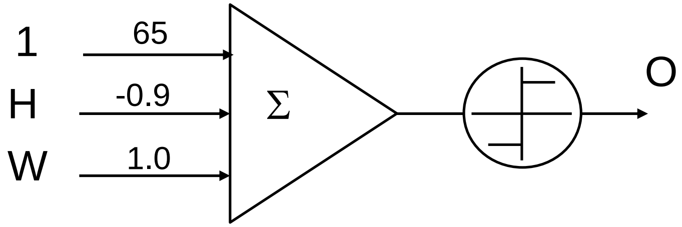
Model Selection

- Various algorithms





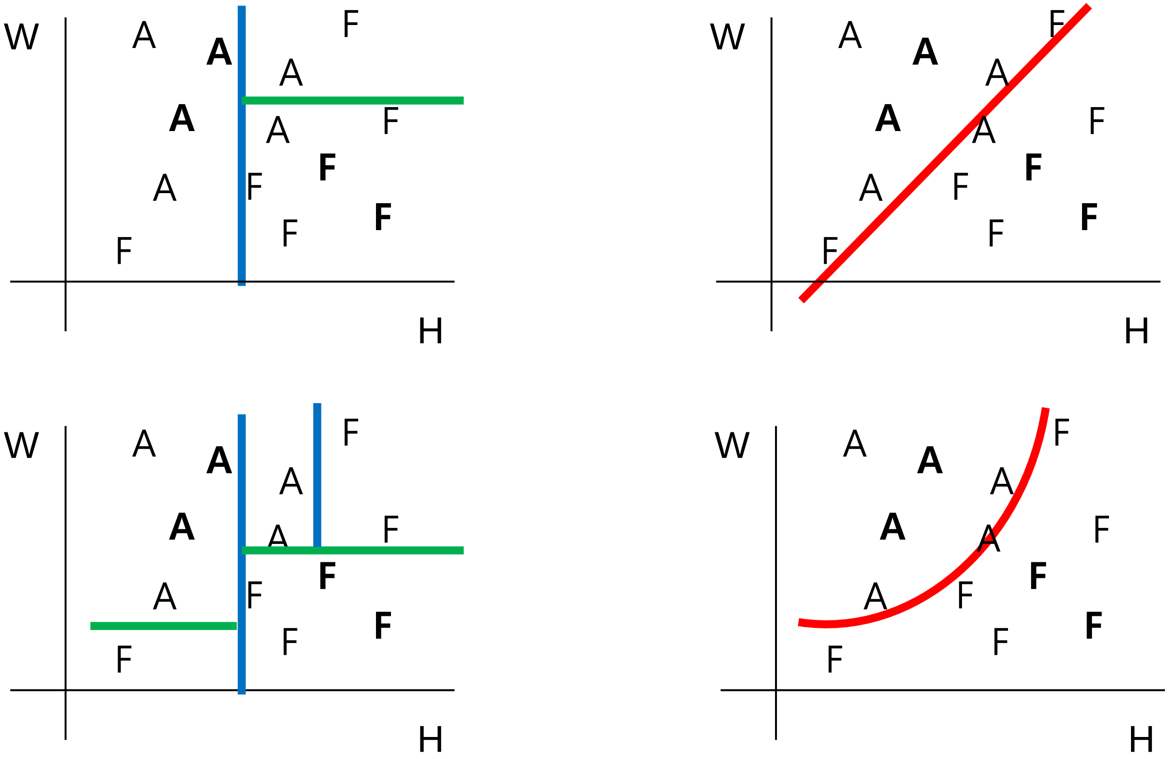




Grade = A if W - 0.9 H + 65 > 0

Model Selection

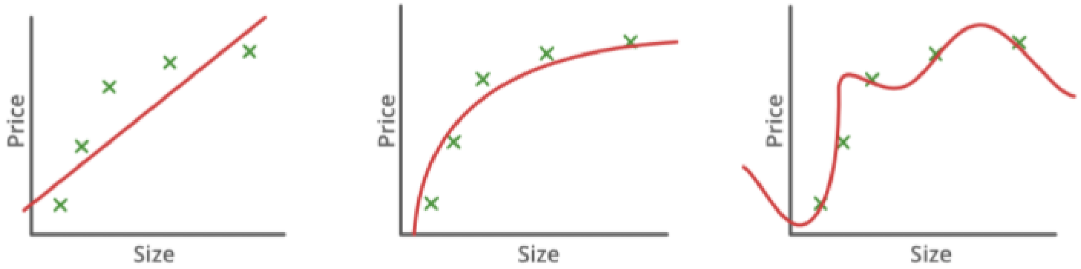
- Various model complexity



Model Selection

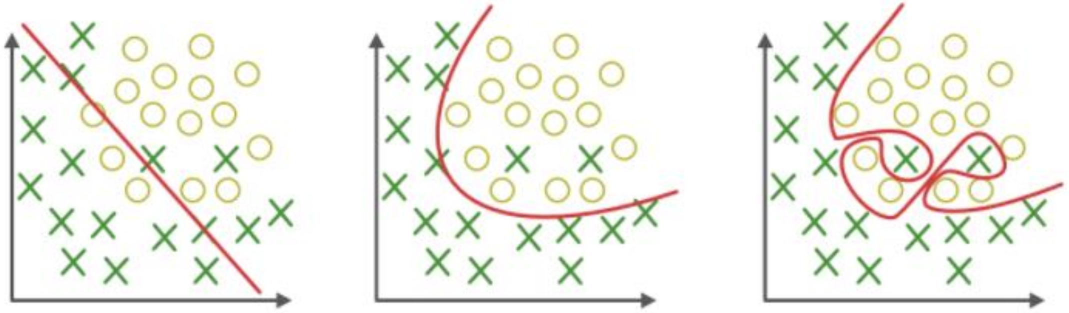
- Overfitting
  - An analysis that corresponds too closely to **training data**
  - Therefore fail to fit **test data** or predict future reliably

$y = ax + b$



$y = ax^5 + bx^4 + \ldots$

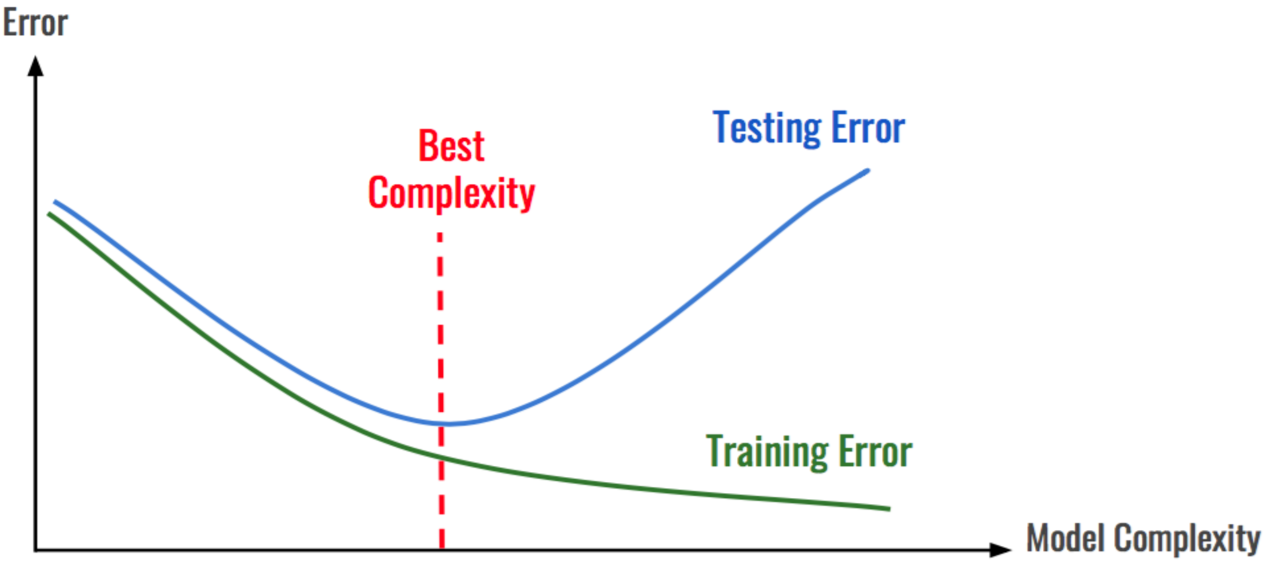
$ax_1 + bx_2 + c > 0$?



$ax_1 + bx_2 + cx_1x_2$
$+ dx_1^2 + ex_2^2$
$+ fx_1^2x_2 + gx_1x_2^2 + \ldots$
> 0 ?

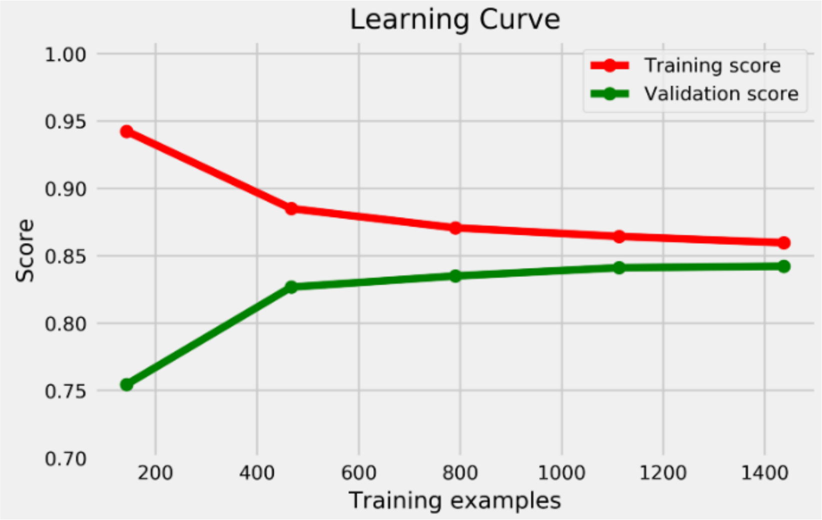
Model Selection

- Overfitting



Performance Evaluation

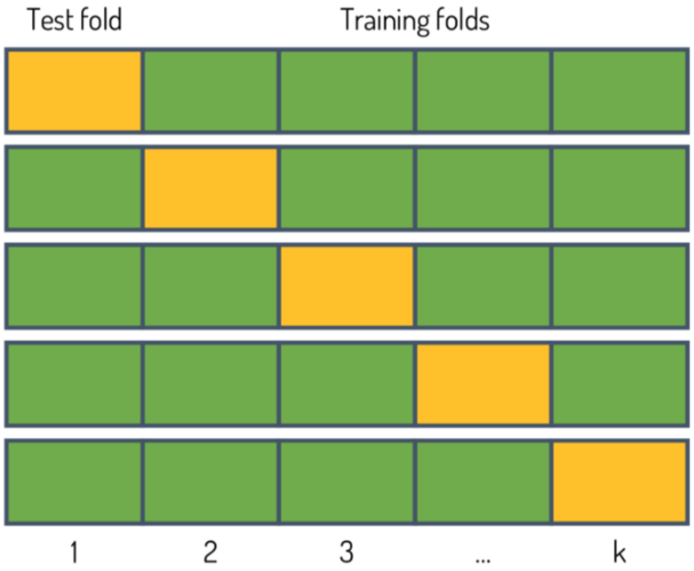
- Accuracy
  - Ratio of number of correct predictions to the total number of input data
- Learning curve



- False/true positive, false/true negative, precision, recall
- AUC
  - Area under the ROC curve

Performance Evaluation

- K-fold cross validation
  1. Shuffle the dataset
  2. Split the dataset into k group
  3. For each group, take it as a test set, and remaining groups as a training set
  4. Fit the model on the training set and evaluate it on the test set
  5. Average the results

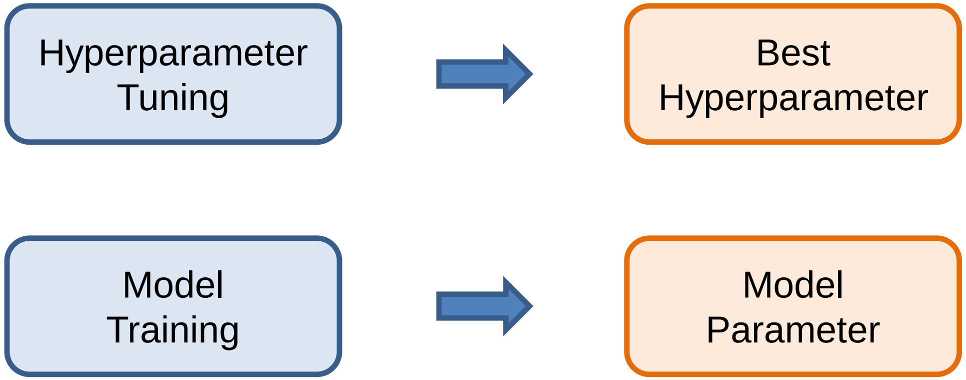


Hyperparameter Tuning

- Hyperparameter
  - Parameter – define the model
    - Ex> y = 1 if $x_2 - 0.9\,x_1 + 65 > 0$ → coefficient 0.9, 65
  - Hyperparameter – define the learning algorithm
    - Ex> gradient descent → learning rate α = 0.01 보정의 크기가 어느정도이냐?

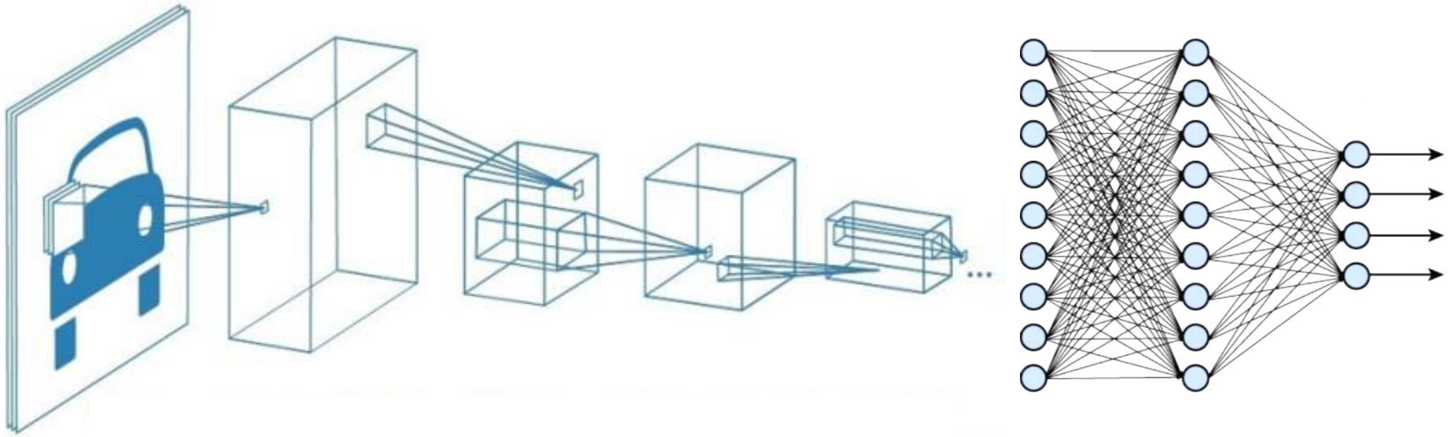
      regression parameter C = 0.1

      convolutional neural network model → size of filters



Deep Learning Models

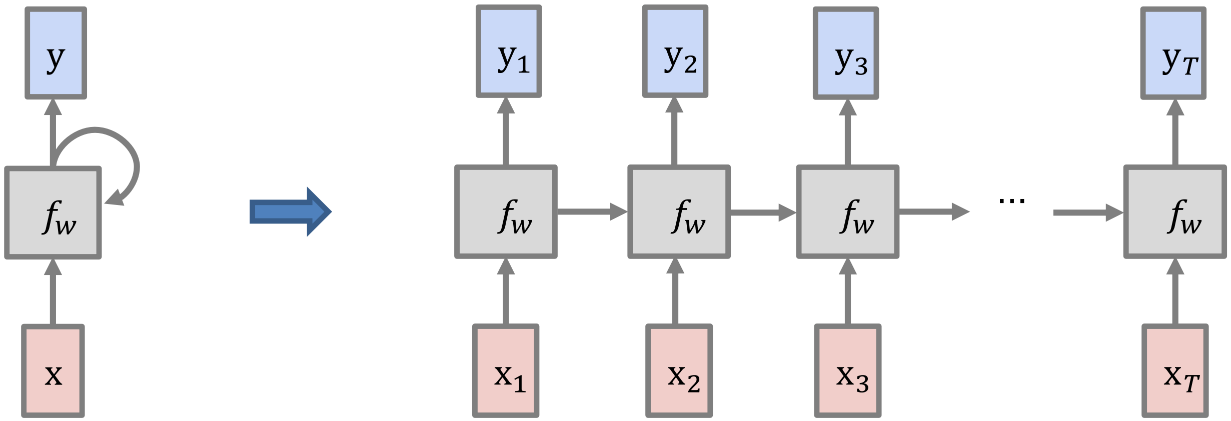
- Deep neural networks for analyzing 2D data
  - 2D data: image
  - Extract relevant features in images
  - Convolution layer
- CNN(Convolutional Neural Networks)



Deep Learning Models

- Deep neural networks for analyzing sequential data
  - Sequential data : text, stock price …
  - Process a sequence
  - Applying a recurrence formula
- RNN(Recurrent Neural Networks)

$y_1, y_2, y_3, \dots$



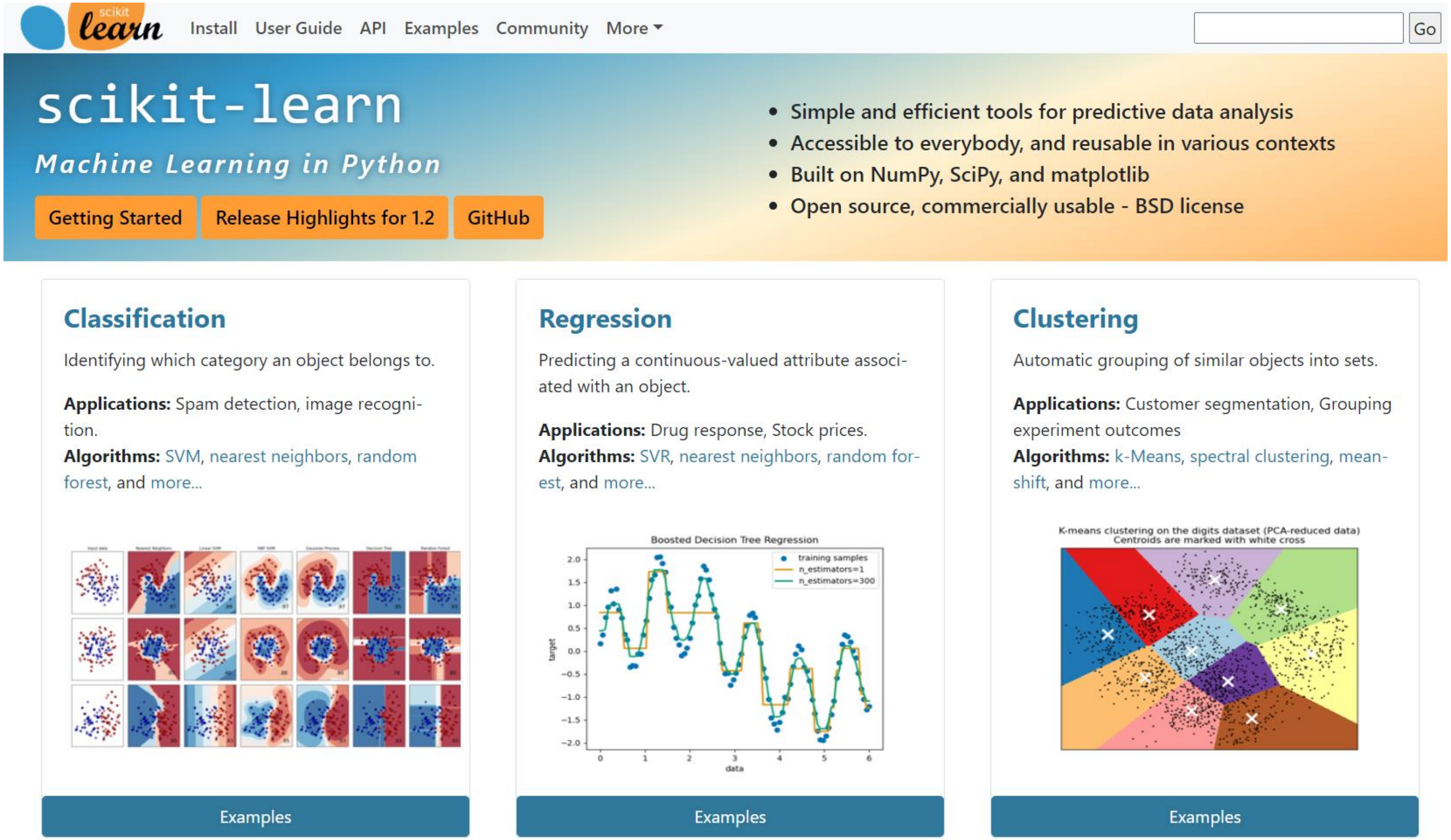
$x_1, x_2, x_3, \dots$

What is Scikit-learn?

- Scikit-learn
  - Open source machine learning library for the Python that supports supervised and unsupervised learning
  - It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries
  - It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.

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

scikit-learn.org



scikit-learn

Machine Learning in Python

- 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

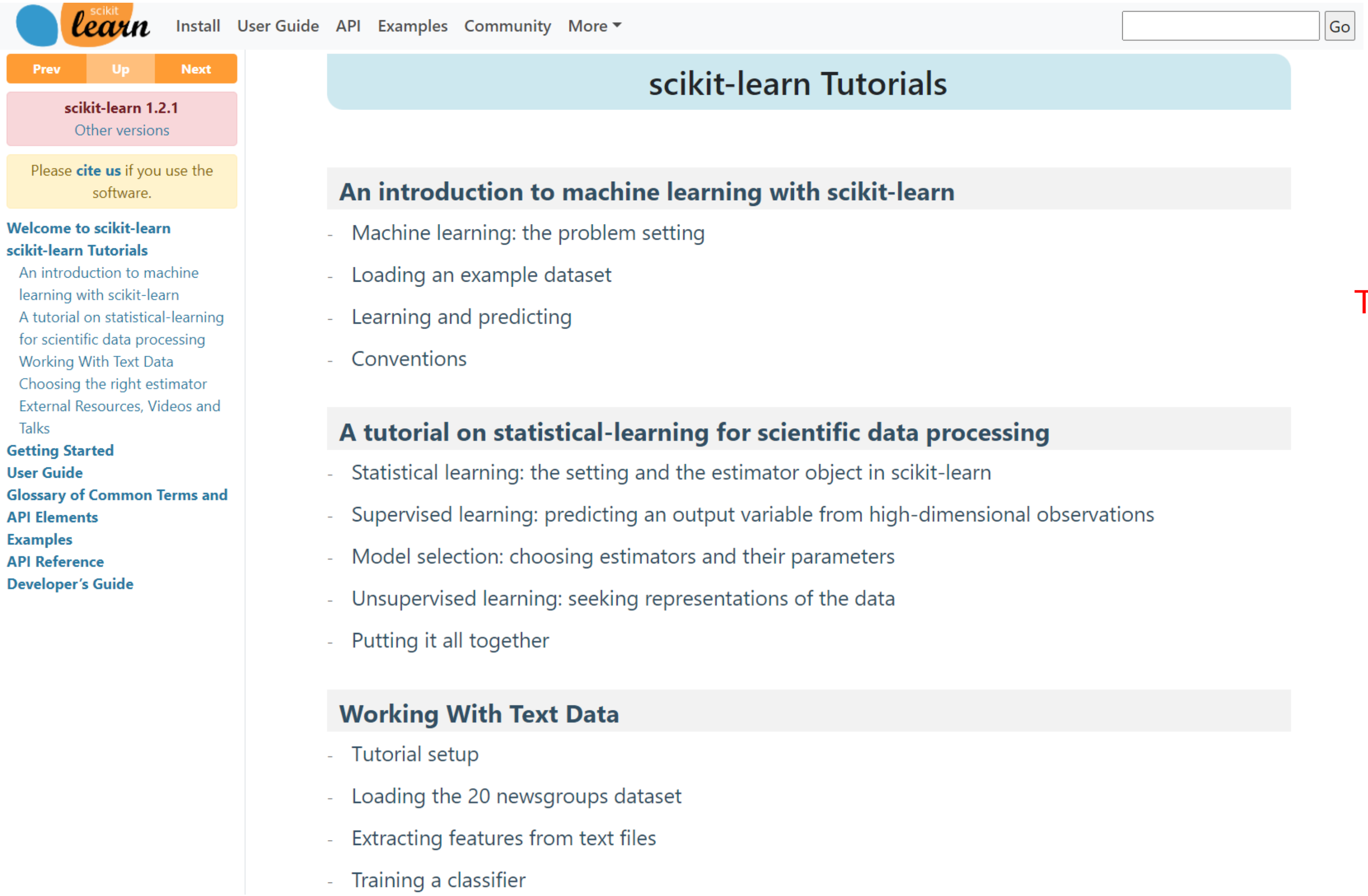
Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross
Examples
Examples
Examples

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

Tutorial



scikit-learn 1.2.1
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

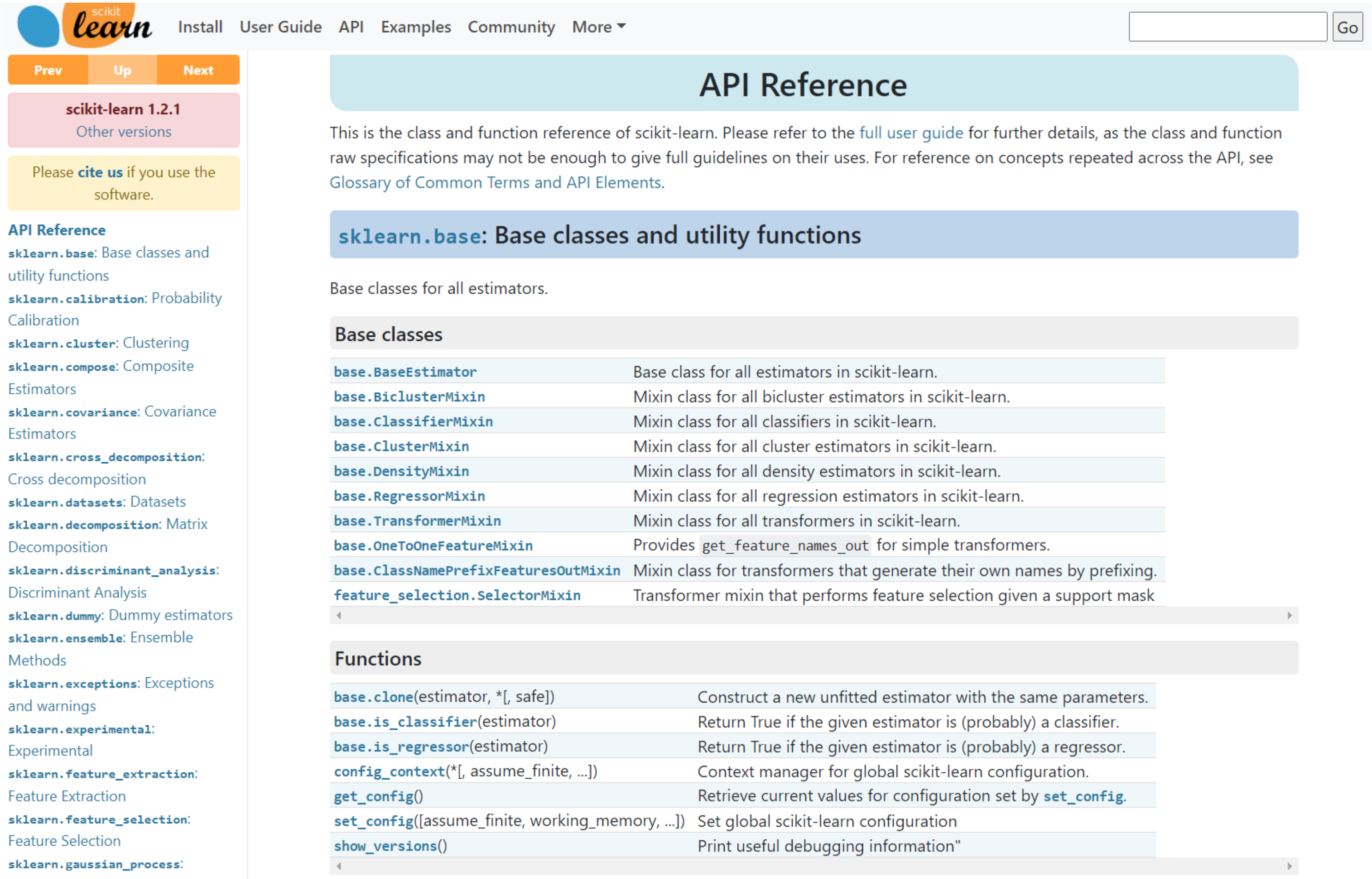
Working With Text Data

- Tutorial setup
- Loading the 20 newsgroups dataset
- Extracting features from text files
- Training a classifier

More
⬇
Tutorial

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

API



scikit-learn 1.2.1
Other versions

Please **cite us** if you use the software.

API Reference
sklearn.base: Base classes and utility functions
sklearn.calibration: Probability Calibration
sklearn.cluster: Clustering
sklearn.compose: Composite Estimators
sklearn.covariance: Covariance Estimators
sklearn.cross_decomposition: Cross decomposition
sklearn.datasets: Datasets
sklearn.decomposition: Matrix Decomposition
sklearn.discriminant_analysis: Discriminant Analysis
sklearn.dummy: Dummy estimators
sklearn.ensemble: Ensemble Methods
sklearn.exceptions: Exceptions and warnings
sklearn.experimental: Experimental
sklearn.feature_extraction: Feature Extraction
sklearn.feature_selection: Feature Selection
sklearn.gaussian_process:

API Reference

This is the class and function reference of scikit-learn. Please refer to the full user guide for further details, as the class and function raw specifications may not be enough to give full guidelines on their uses. For reference on concepts repeated across the API, see Glossary of Common Terms and API Elements.

sklearn.base: Base classes and utility functions

Base classes for all estimators.

Base classes

base.BaseEstimator	Base class for all estimators in scikit-learn.
base.BiclusterMixin	Mixin class for all bicluster estimators in scikit-learn.
base.ClassifierMixin	Mixin class for all classifiers in scikit-learn.
base.ClusterMixin	Mixin class for all cluster estimators in scikit-learn.
base.DensityMixin	Mixin class for all density estimators in scikit-learn.
base.RegressorMixin	Mixin class for all regression estimators in scikit-learn.
base.TransformerMixin	Mixin class for all transformers in scikit-learn.
base.OneToOneFeatureMixin	Provides get_feature_names_out for simple transformers.
base.ClassNamePrefixFeaturesOutMixin	Mixin class for transformers that generate their own names by prefixing.
feature_selection.SelectorMixin	Transformer mixin that performs feature selection given a support mask

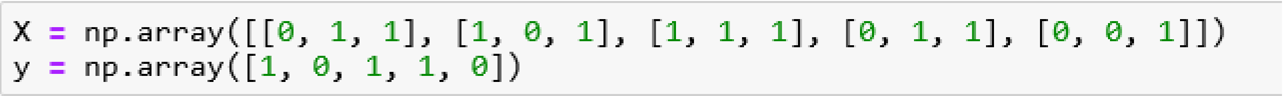
Functions

base.clone(estimator, *[, safe])	Construct a new unfitted estimator with the same parameters.
base.is_classifier(estimator)	Return True if the given estimator is (probably) a classifier.
base.is_regressor(estimator)	Return True if the given estimator is (probably) a regressor.
config_context(*[, assume_finite, ...])	Context manager for global scikit-learn configuration.
get_config()	Retrieve current values for configuration set by set_config.
set_config([assume_finite, working_memory, ...])	Set global scikit-learn configuration
show_versions()	Print useful debugging information"

https://scikit-learn.org/stable/modules/classes.html

Training a Model

- Task
  - Classify x = (x1, x2, x3) to y = 1 or 0
- Training dataset
  - 5 instances (example data) with known labels



```
X = np.array([[0, 1, 1], [1, 0, 1], [1, 1, 1], [0, 1, 1], [0, 0, 1]])
y = np.array([1, 0, 1, 1, 0])
```

3 features

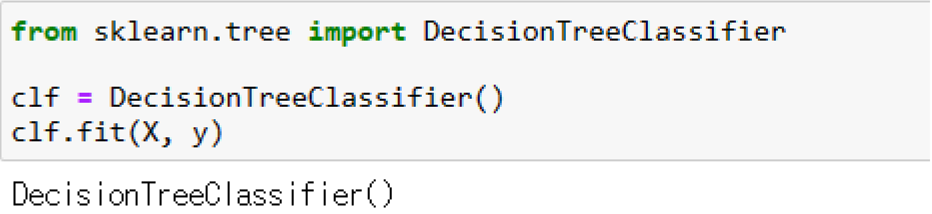
label

5 instances

X			y
0,	1,	1	1
1,	0,	1	**0**
1,	1,	1	**1**
0,	1,	1	**1**
0,	0,	1	**0**

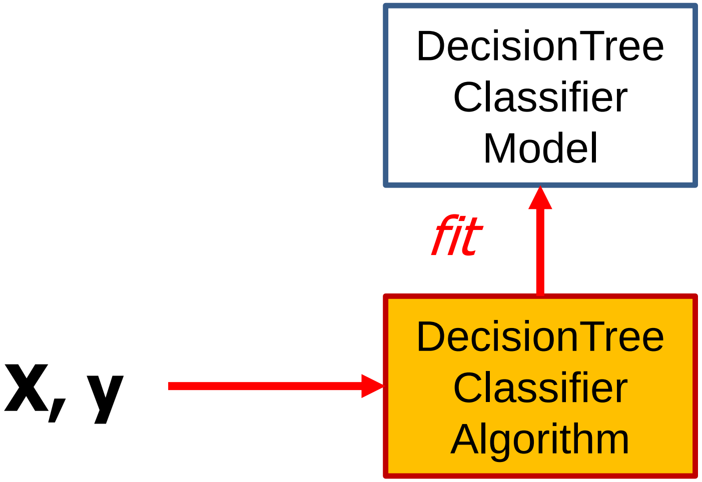
Training a Model

- Training (learning) - **fit**
    - Learning Decision Tree Classifier model with the training dataset



```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()
clf.fit(X, y)
```

```
DecisionTreeClassifier()
```



DecisionTree Classifier Model

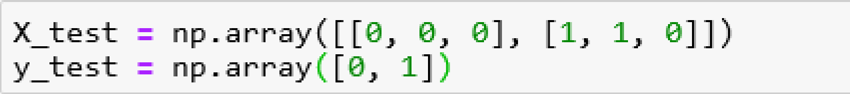
fit

X, y

DecisionTree Classifier Algorithm

Predicting using the Model

- Test dataset
  - 2 new instances



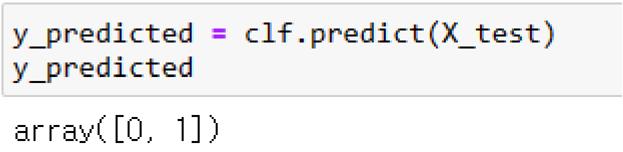
```
X_test = np.array([[0, 0, 0], [1, 1, 0]])
y_test = np.array([0, 1])
```

new instances

X_test	y_predicted
0, 0, 0	?
1, 1, 0	**?**

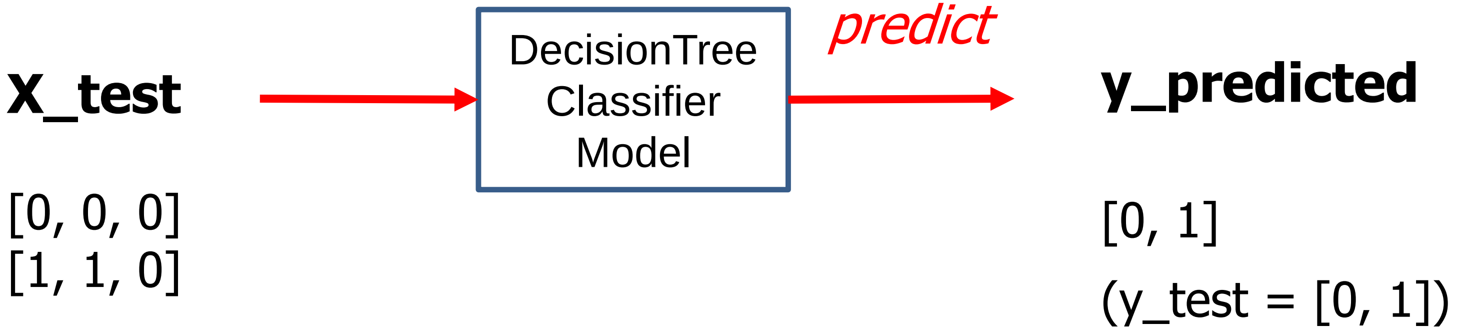
Predicting using the Model

- Predicting labels – **predict**
  - Predict the label of new x using the learned model



```
y_predicted = clf.predict(X_test)
y_predicted
```

```
array([0, 1])
```



X_test

[0, 0, 0]
[1, 1, 0]

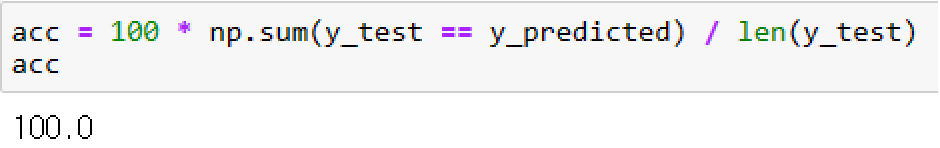
DecisionTree Classifier Model

predict

y_predicted

[0, 1]

(y_test = [0, 1])



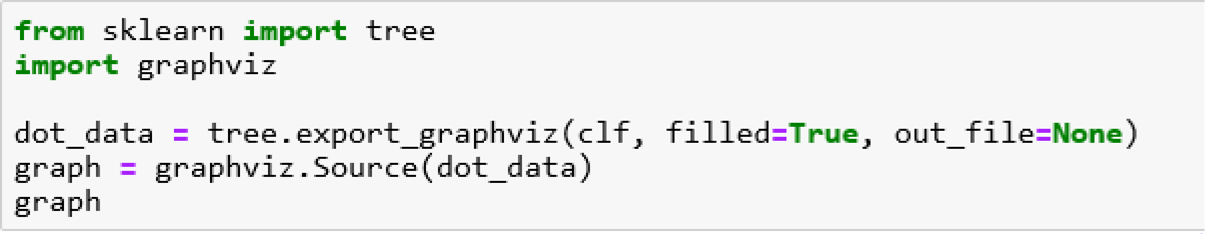
```
acc = 100 * np.sum(y_test == y_predicted) / len(y_test)
acc
```

```
100.0
```

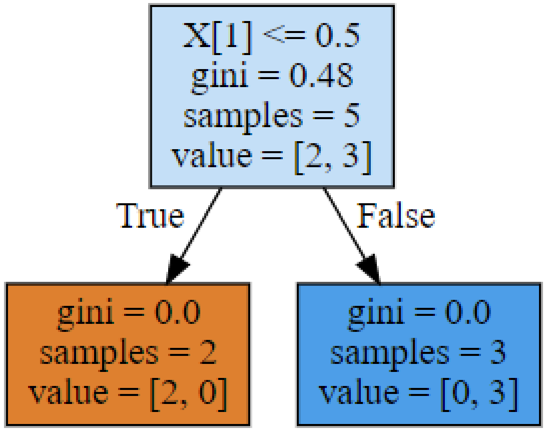
⇒ *100 % accuracy*

Visualize the Model

- Visualizing Decision Tree model using graphviz



```
from sklearn import tree
import graphviz

dot_data = tree.export_graphviz(clf, filled=True, out_file=None)
graph = graphviz.Source(dot_data)
graph
```



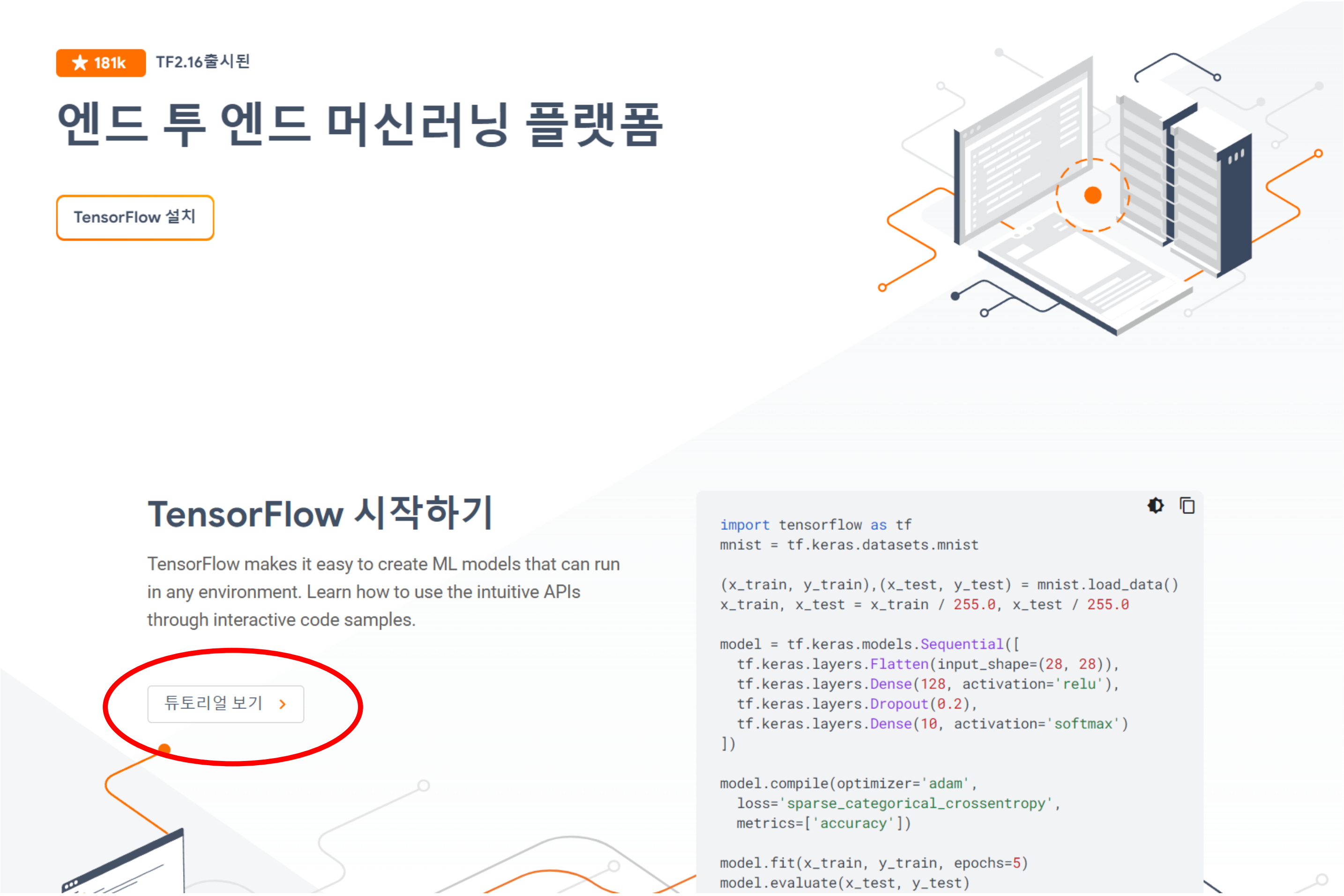
What is TensorFlow?

- TensorFlow
  - TensorFlow is an open source platform for machine learning
    - Originally developed by Google Brain team to conduct machine learning and deep neural networks research
  - It has a comprehensive ecosystem of tools, libraries and resources
  - TensorFlow computations are expressed as dataflow graphs on tensors
  - It can run on multiple CPUs and GPUs
- TensorFlow 2.0
  - Introduced a number of simplifications
  - Improvements to the performance on GPU

TensorFlow

https://www.tensorflow.org/

tensorflow.org



엔드 투 엔드 머신러닝 플랫폼

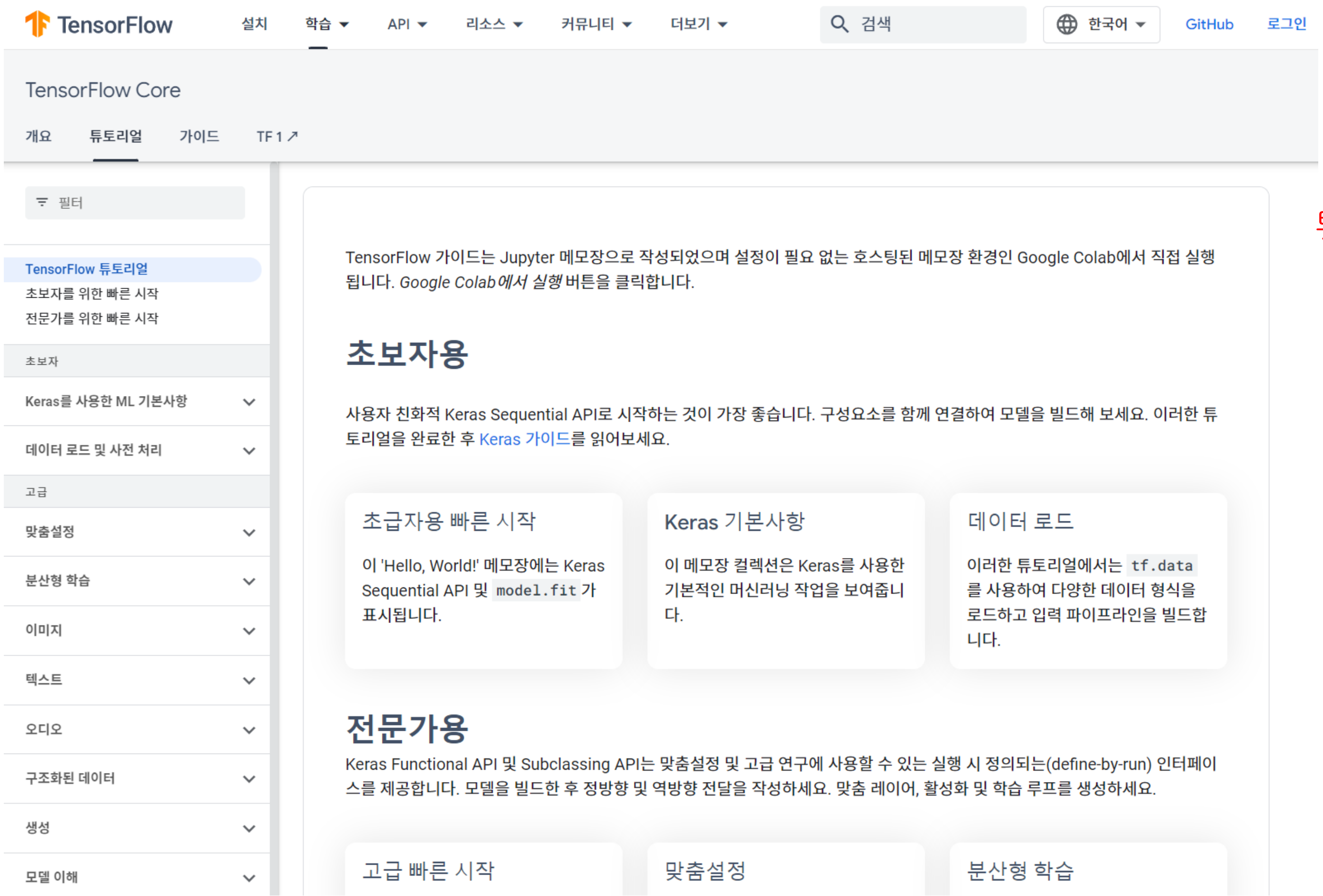
TensorFlow 시작하기

TensorFlow makes it easy to create ML models that can run in any environment. Learn how to use the intuitive APIs through interactive code samples.

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
  loss='sparse_categorical_crossentropy',
  metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

https://www.tensorflow.org/

Tutorial



TensorFlow 가이드는 Jupyter 메모장으로 작성되었으며 설정이 필요 없는 호스팅된 메모장 환경인 Google Colab에서 직접 실행됩니다. *Google Colab에서 실행* 버튼을 클릭합니다.

초보자용

사용자 친화적 Keras Sequential API로 시작하는 것이 가장 좋습니다. 구성요소를 함께 연결하여 모델을 빌드해 보세요. 이러한 튜토리얼을 완료한 후 Keras 가이드를 읽어보세요.

초급자용 빠른 시작

이 'Hello, World!' 메모장에는 Keras Sequential API 및 `model.fit`가 표시됩니다.

Keras 기본사항

이 메모장 컬렉션은 Keras를 사용한 기본적인 머신러닝 작업을 보여줍니다.

데이터 로드

이러한 튜토리얼에서는 `tf.data`를 사용하여 다양한 데이터 형식을 로드하고 입력 파이프라인을 빌드합니다.

전문가용

Keras Functional API 및 Subclassing API는 맞춤설정 및 고급 연구에 사용할 수 있는 실행 시 정의되는(define-by-run) 인터페이스를 제공합니다. 모델을 빌드한 후 정방향 및 역방향 전달을 작성하세요. 맞춤 레이어, 활성화 및 학습 루프를 생성하세요.

고급 빠른 시작

맞춤설정

분산형 학습

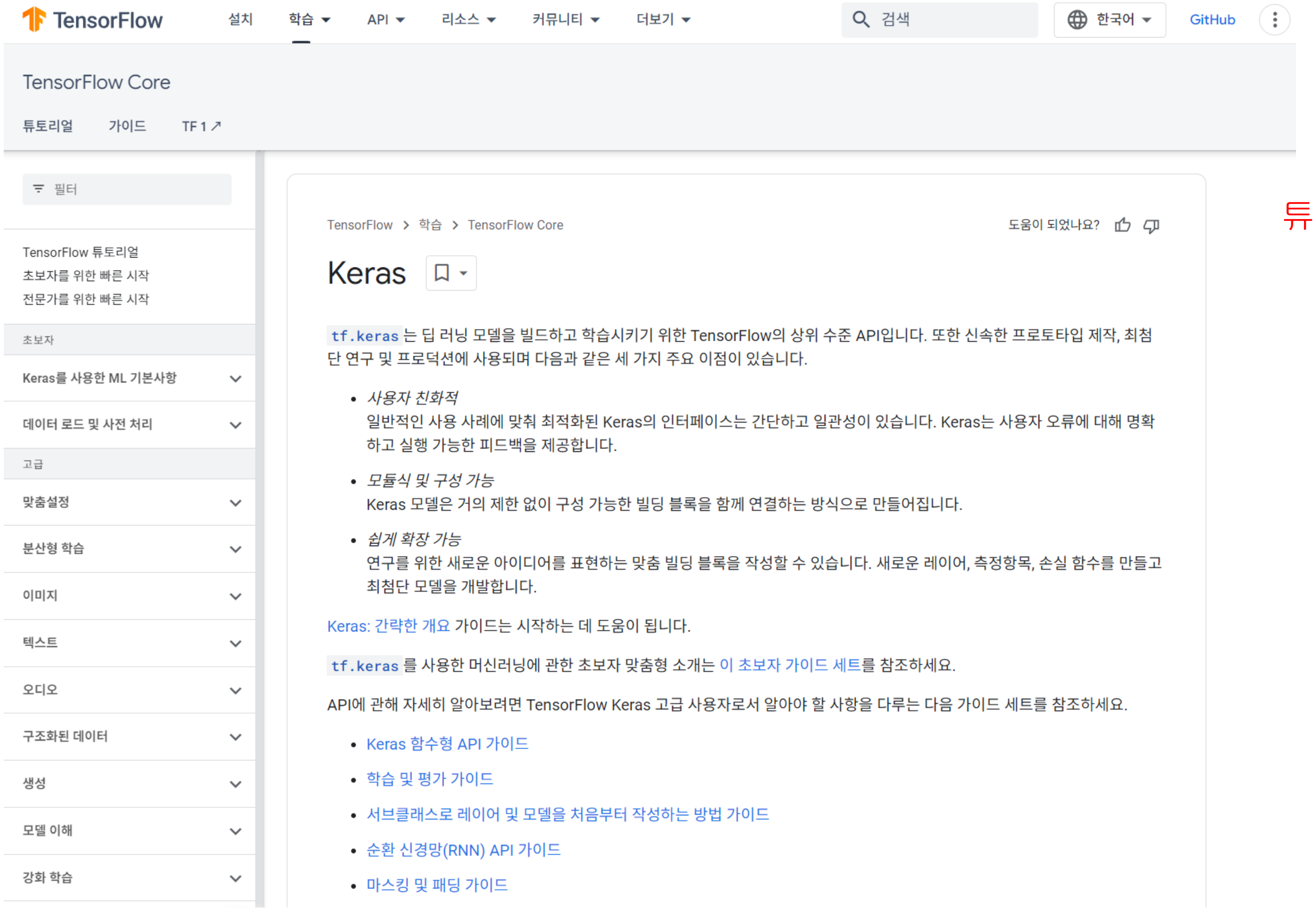
튜토리얼 보기

⬇

튜토리얼

https://www.tensorflow.org/tutorials

Keras



Keras

`tf.keras` 는 딥 러닝 모델을 빌드하고 학습시키기 위한 TensorFlow의 상위 수준 API입니다. 또한 신속한 프로토타입 제작, 최첨단 연구 및 프로덕션에 사용되며 다음과 같은 세 가지 주요 이점이 있습니다.

- *사용자 친화적*
  일반적인 사용 사례에 맞춰 최적화된 Keras의 인터페이스는 간단하고 일관성이 있습니다. Keras는 사용자 오류에 대해 명확하고 실행 가능한 피드백을 제공합니다.
- *모듈식 및 구성 가능*
  Keras 모델은 거의 제한 없이 구성 가능한 빌딩 블록을 함께 연결하는 방식으로 만들어집니다.
- *쉽게 확장 가능*
  연구를 위한 새로운 아이디어를 표현하는 맞춤 빌딩 블록을 작성할 수 있습니다. 새로운 레이어, 측정항목, 손실 함수를 만들고 최첨단 모델을 개발합니다.

Keras: 간략한 개요 가이드는 시작하는 데 도움이 됩니다.

`tf.keras` 를 사용한 머신러닝에 관한 초보자 맞춤형 소개는 이 초보자 가이드 세트를 참조하세요.

API에 관해 자세히 알아보려면 TensorFlow Keras 고급 사용자로서 알아야 할 사항을 다루는 다음 가이드 세트를 참조하세요.

- Keras 함수형 API 가이드
- 학습 및 평가 가이드
- 서브클래스로 레이어 및 모델을 처음부터 작성하는 방법 가이드
- 순환 신경망(RNN) API 가이드
- 마스킹 및 패딩 가이드

튜토리얼 보기

⬇

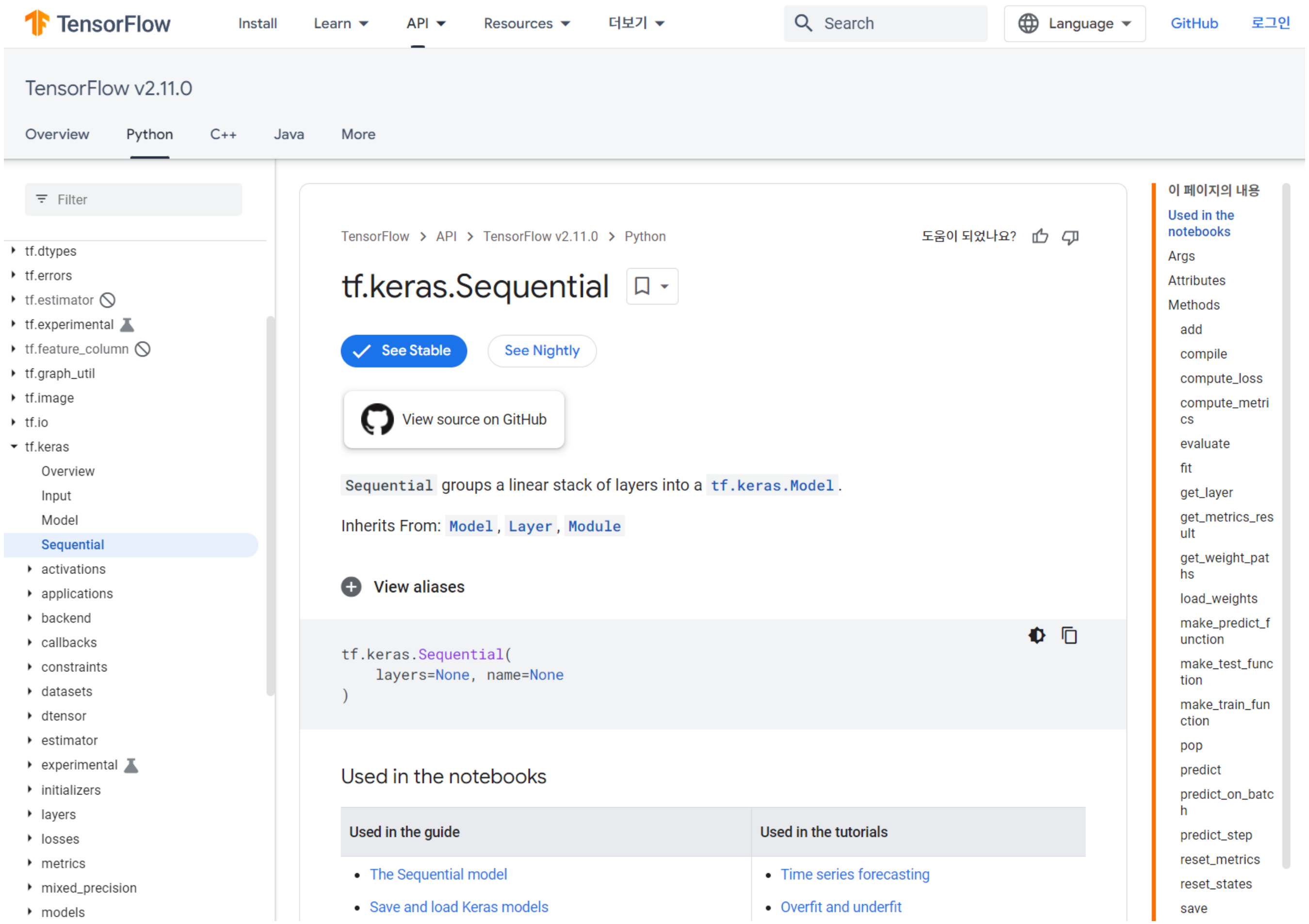
가이드

⬇

Keras

https://www.tensorflow.org/guide/keras

API



TensorFlow > API > TensorFlow v2.11.0 > Python

tf.keras.Sequential

`Sequential` groups a linear stack of layers into a `tf.keras.Model`.

Inherits From: `Model`, `Layer`, `Module`

```
tf.keras.Sequential(
    layers=None, name=None
)
```

Used in the notebooks

Used in the guide	Used in the tutorials
• The Sequential model	• Time series forecasting
• Save and load Keras models	• Overfit and underfit

https://www.tensorflow.org/api_docs/python/tf