Introduction to Machine Learning with Python

AI for the Junior Atmospheric Scientist

By Daeho Jin

2021/02/03 Wed

Before Start

This Lecture is

designed for AI “Beginners”

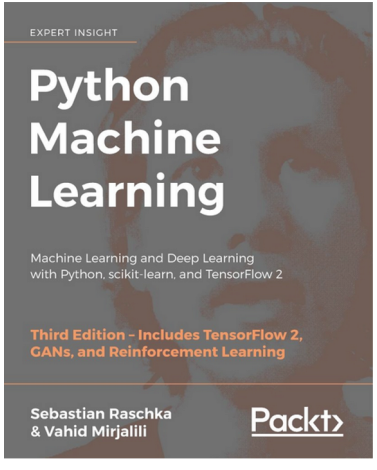
with some knowledge of Python3

(Numpy+Matplotlib)

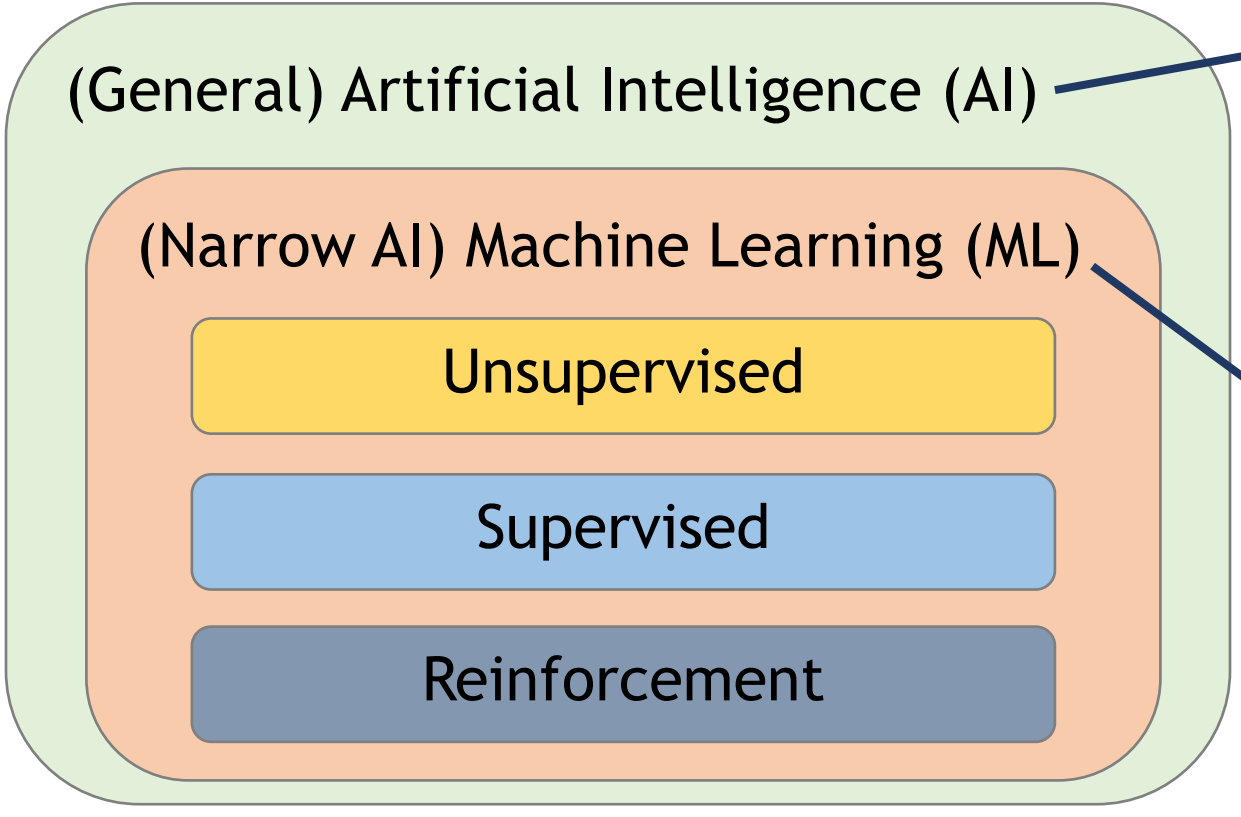
Example codes are available at

https://github.com/DJ4seasons/Python_ML_Basic4beginner

Credited to this book:



INTRO: Dfinition



Computers that can imitate human intellect and behavior

Statistical algorithms that enable AI implementation through data

INTRO: Definition

Unsupervised

No Label

Find hidden structure in data

Clustering

Supervised

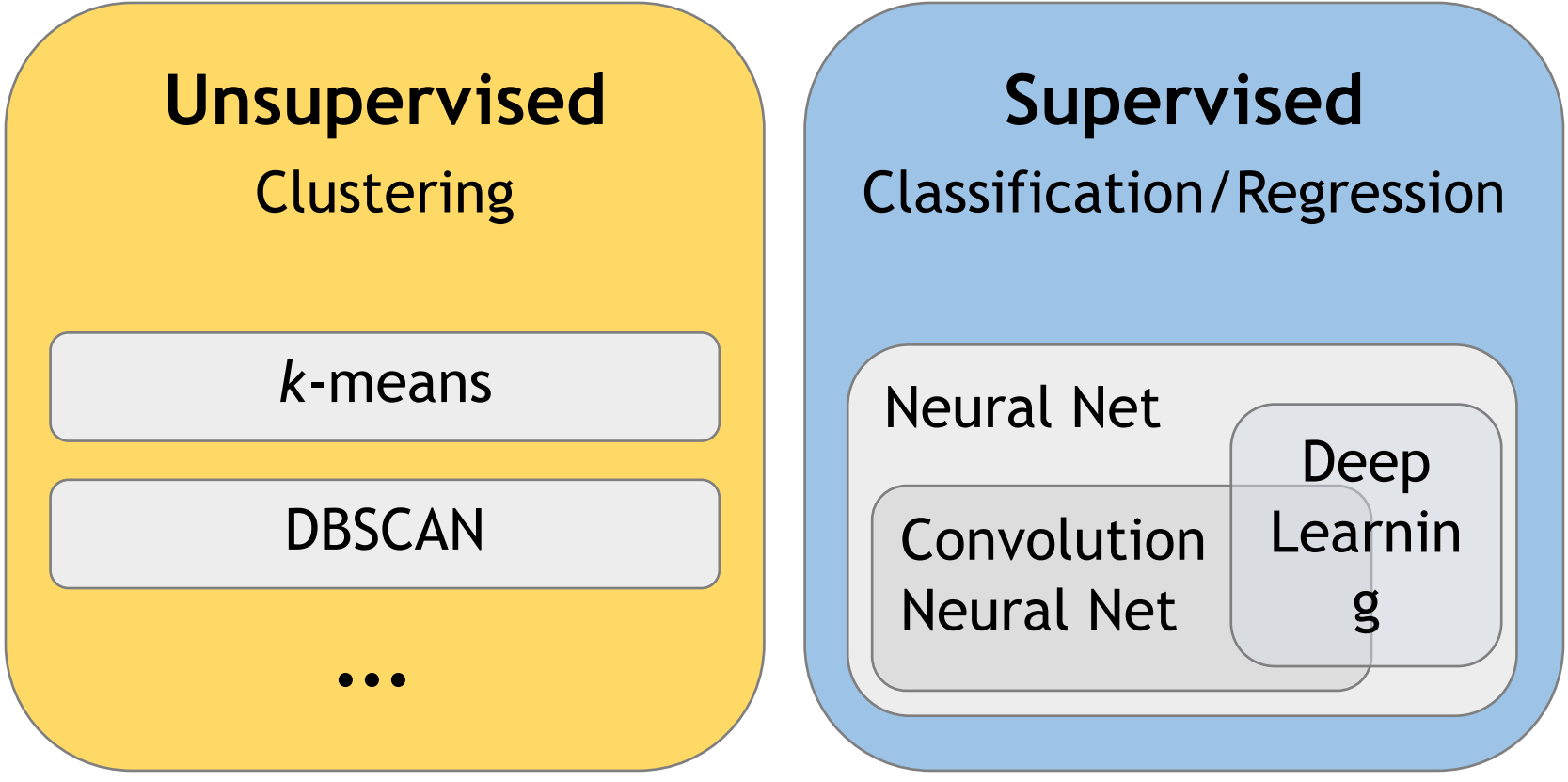
Need Label

Predict outcome/future

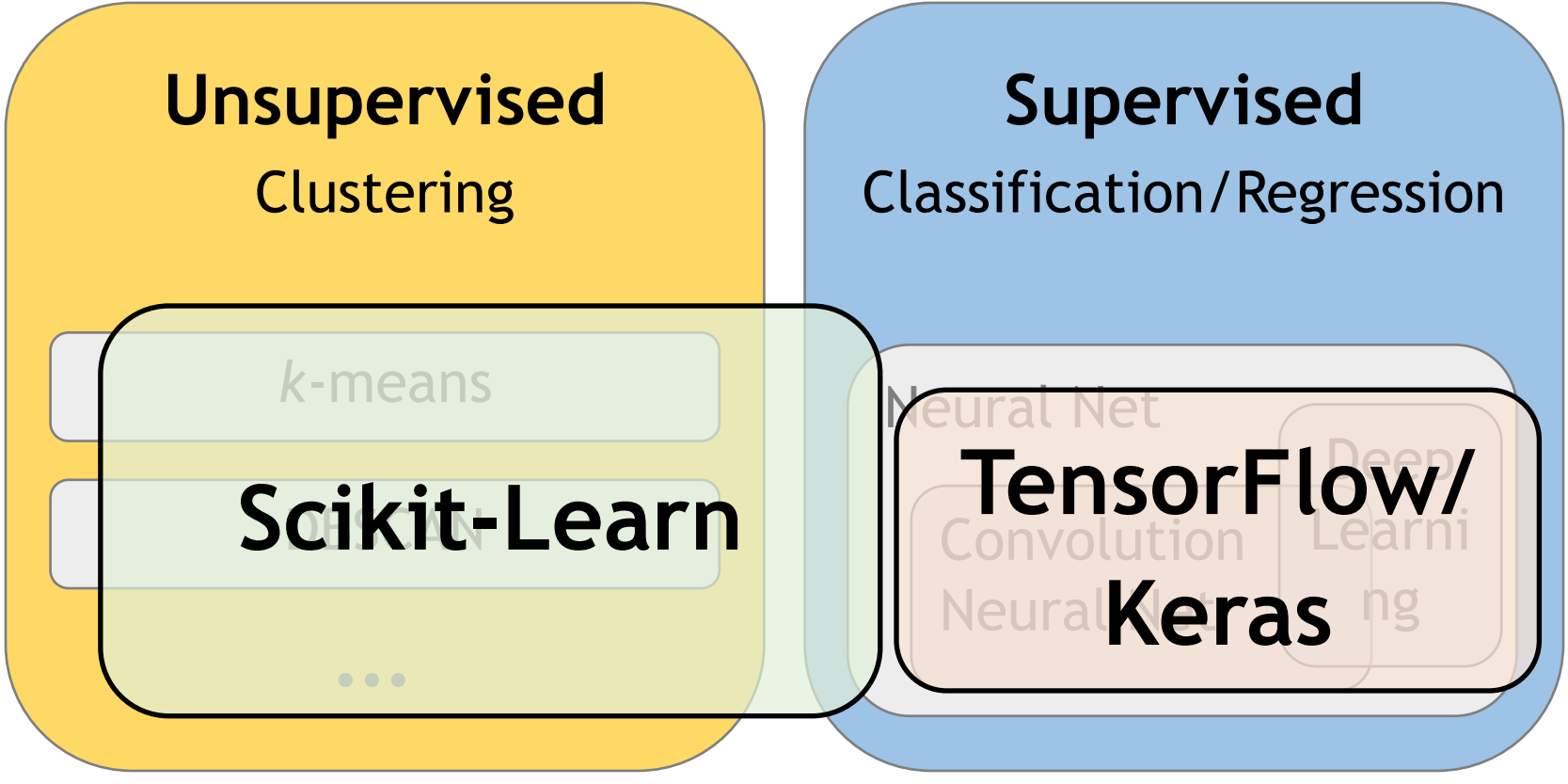
Classification

Regression

INTRO: Algorithm

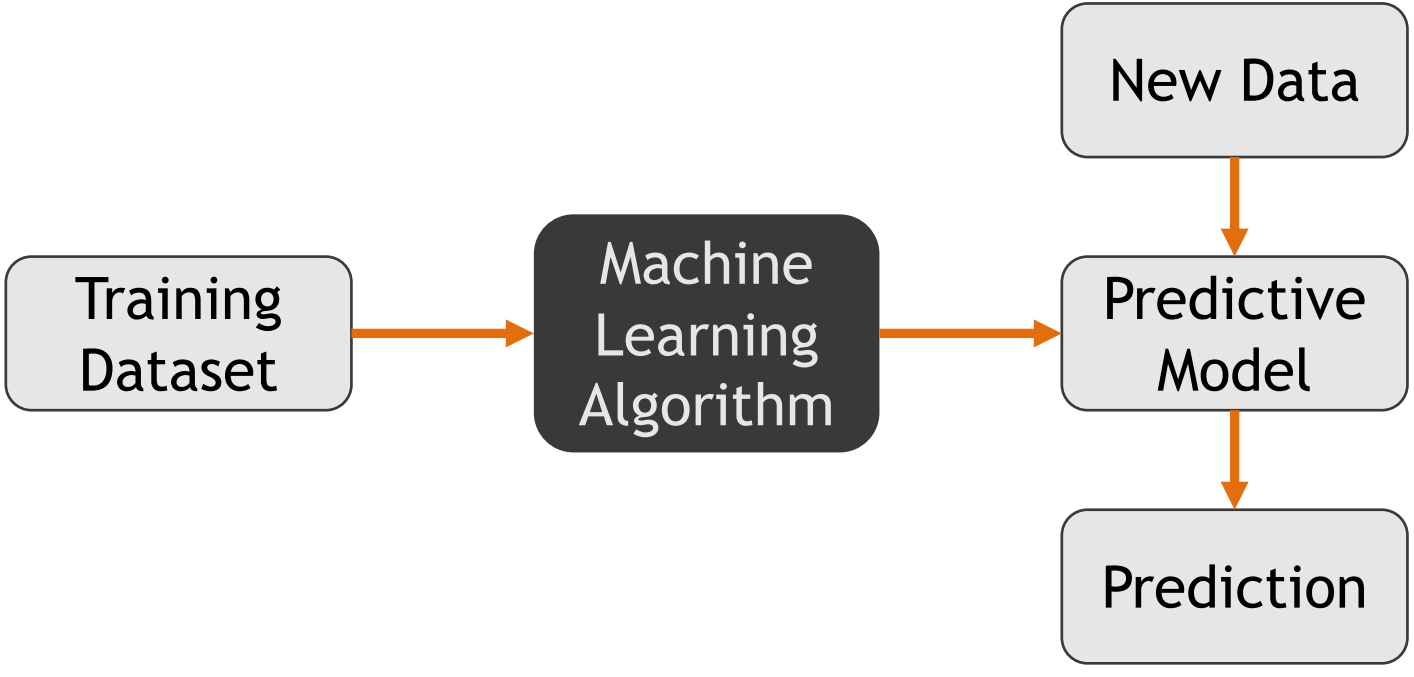


INTRO: Python Package

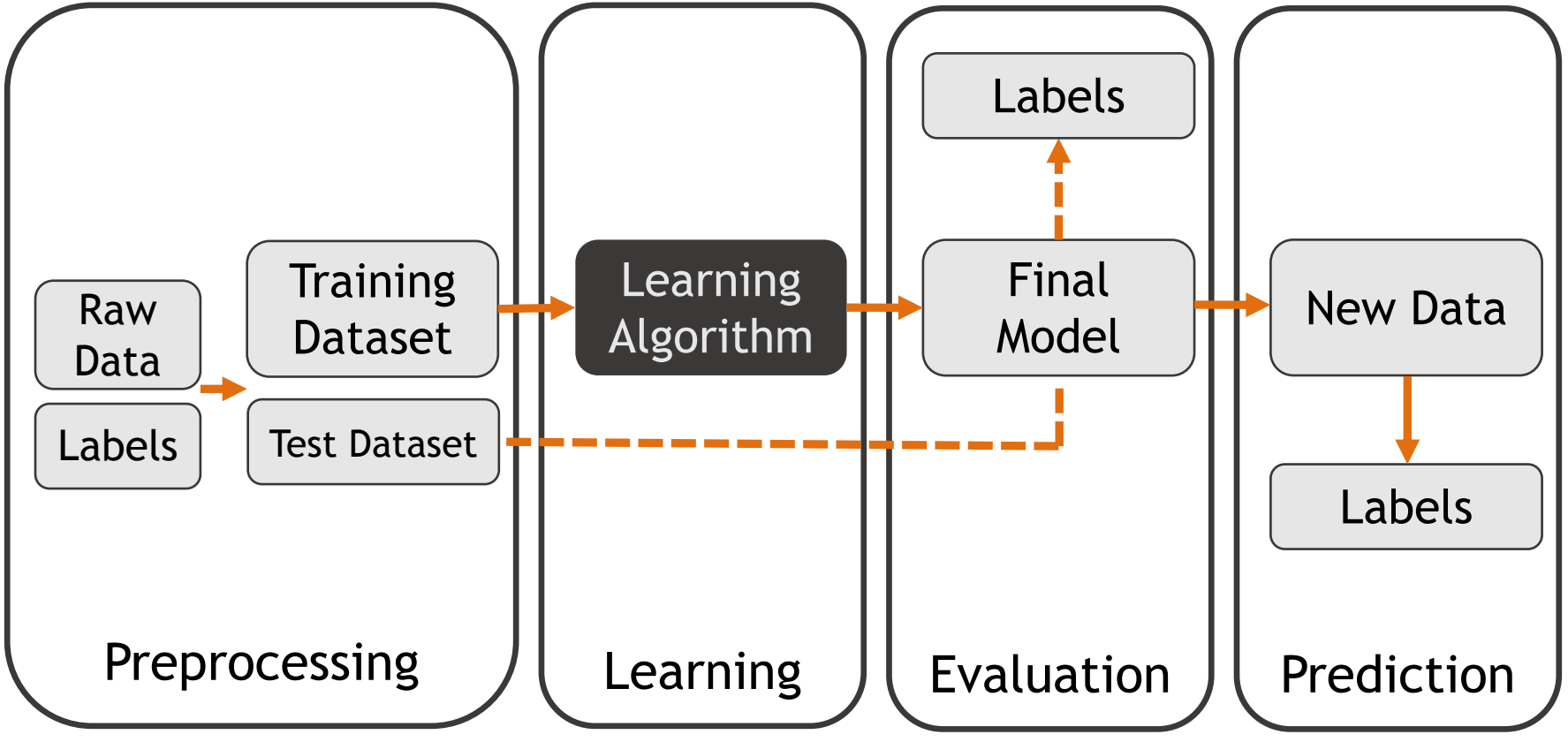


ML
Modeling Process

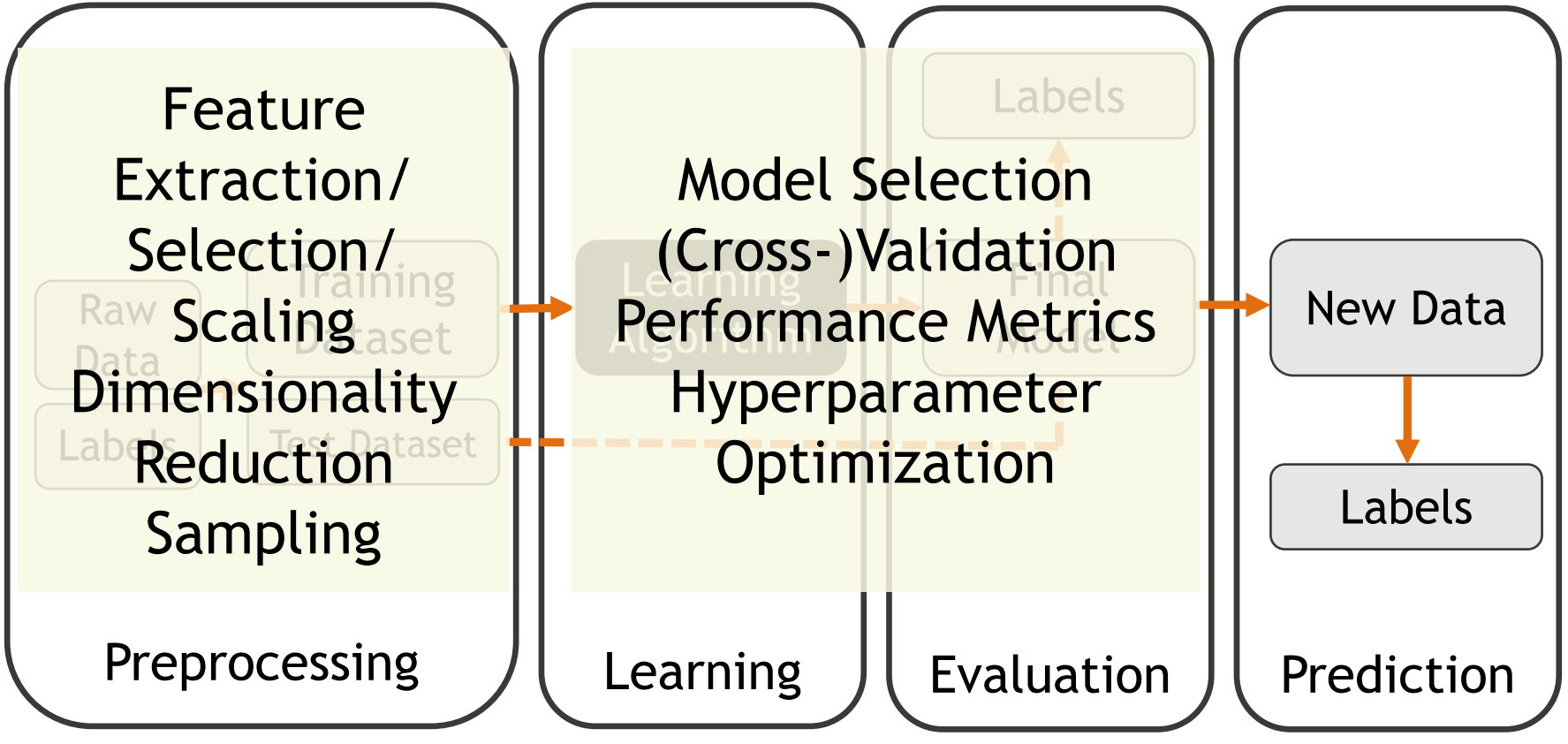
INTRO: Modeling Process



INTRO: Modeling Process



INTRO: Modeling Process

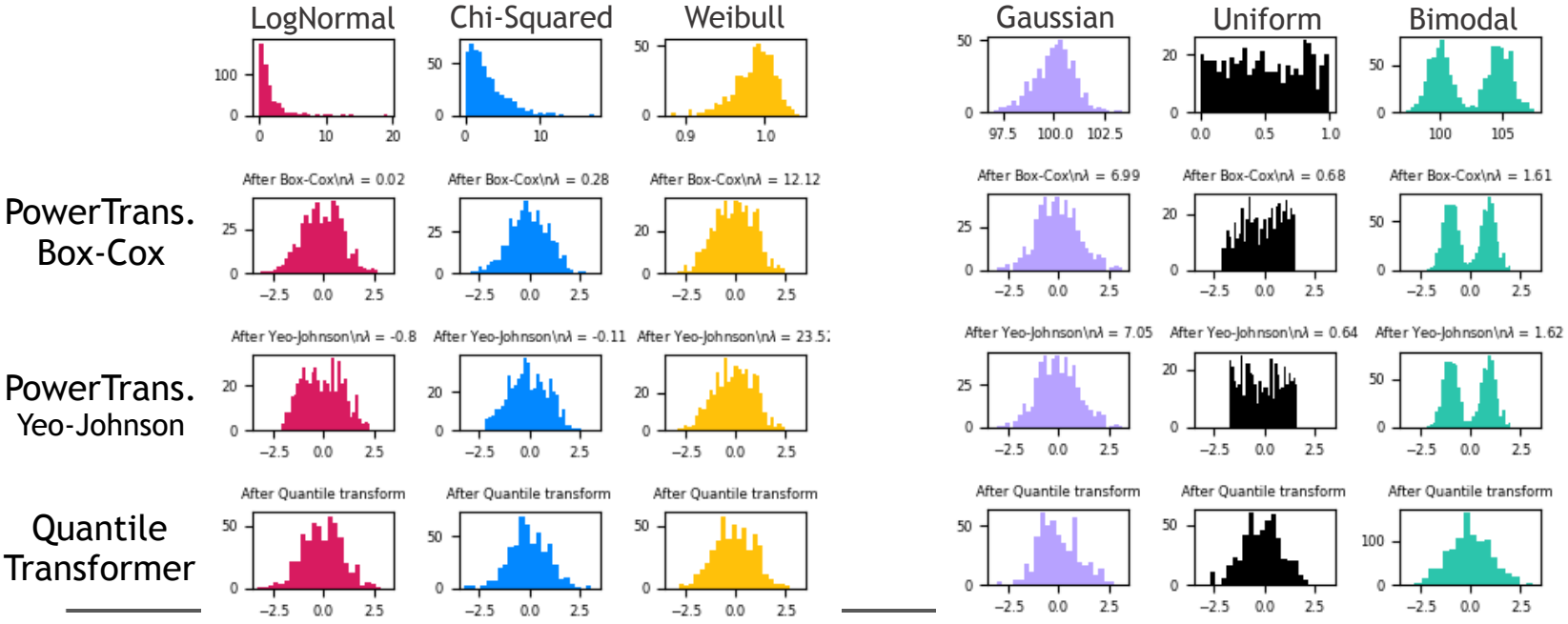


Preprocessing: Scaling

- What is scaling and Why need it?
  - Make sure features are on similar scale or range
  - Prevent one feature dominate others
  - Not required for every algorithm (e.g., decision tree), but for the most.
- Popular linear scaler
  - Min-Max Normalization, $\frac{x-\min(x)}{\max(x)-\min(x)}$
  - Standardization, $\frac{x-average(x)}{Std.Dev(x)}$

Preprocessing: Scaling

- Nonlinear transformation
  - PowerTransformer / QuantileTransformer in Scikit-learn



https://scikit-learn.org/stable/auto_examples/preprocessing/plot_map_data_to_normal.html

Preprocessing: Label Encoding

- Categorical label can not be used for ML algorithm
  - E.g., [“red”, “blue”, “green”] ➔ [0, 1, 2]
  - Except decision tree model
- Sometimes, relationship is distorted by simple numeric transform
  - “0=red” is thought of as closer to “1=blue” than “2=green”
- **One-Hot Encoding** is used for this situation by vectorizing
  - “red” ➔ [1,0,0]
  - “blue” ➔ [0,1,0]
  - “green” ➔ [0,0,1]

https://scikit-learn.org/stable/auto_examples/preprocessing/plot_map_data_to_normal.html

Preprocessing: Sampling

- "Population **BALANCE**" is important (particularly for classification)
- Because "small population" means "less learning opportunity"
- An example of label population distribution for "Cloud type classification" model

Clear sky	Low stratiform	Low cumulus	Deep convection	Mid alto-clouds	High anvil	High cirrus	Low+High	Low+Mid	Mid+High
35%	12%	5%	5%	4%	3.5%	20%	10%	2.5%	3%

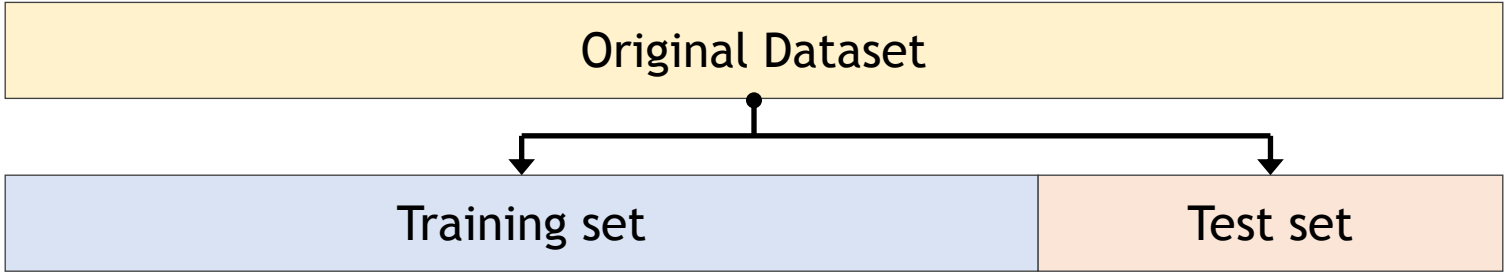
- How to do sampling with this data?

Preprocessing: Sampling

Clear sky	Low stratiform	Low cumulus	Deep convection	Mid alto-clouds	High anvil	High cirrus	Low+High	Low+Mid	Mid+High
35%	12%	5%	5%	4%	3.5%	20%	10%	2.5%	3%

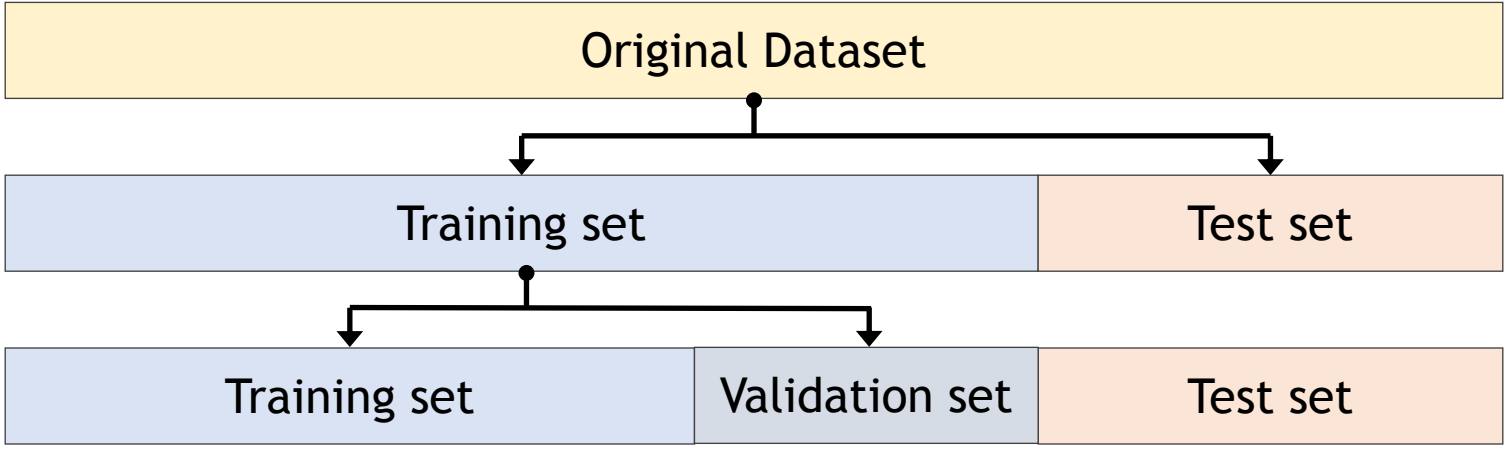
- First of all, filter out “Clear sky”
  - Because “Clear sky” can be easily classified by simple filtering
- Second, perform sampling for balance population distribution
  - Oversampling, e.g. random duplication of low population types
  - Undersampling, e.g. random choice for high population types
- Have to exactly same population among labels?
  - Small differences are not a big deal.
  - Depending on the designer’s weight on labels

Preprocessing: Data Dividing



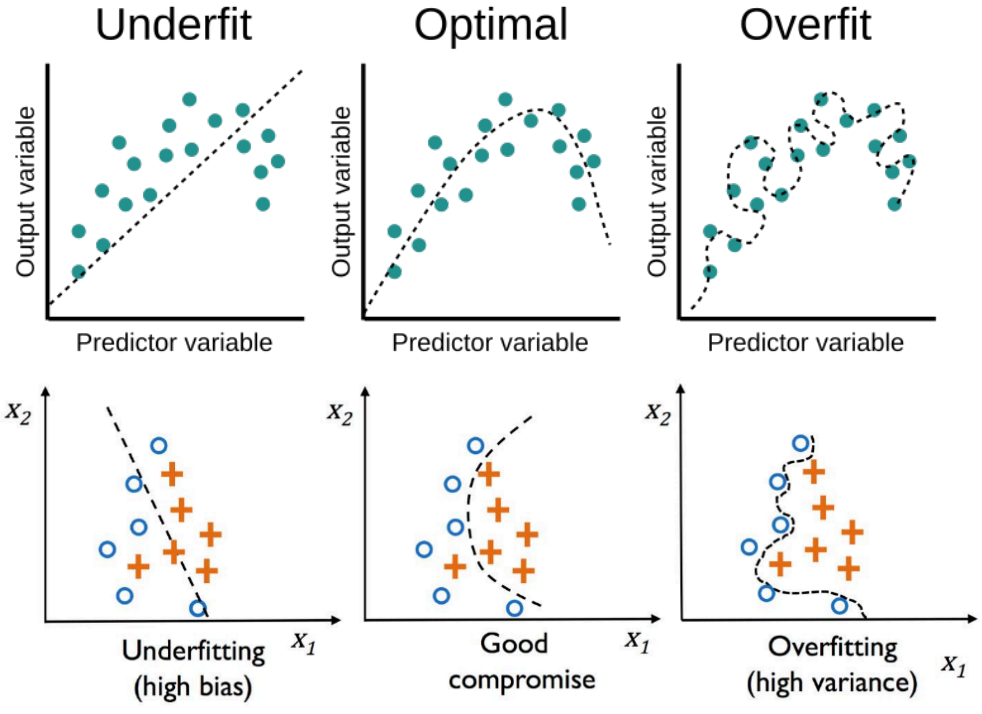
- Which ratio is good for dividing?
- Training : Test = 9:1? 7:3? 5:5?
- Depending on the size of original dataset
  - Usually the size of test dataset should be larger than some minimum level

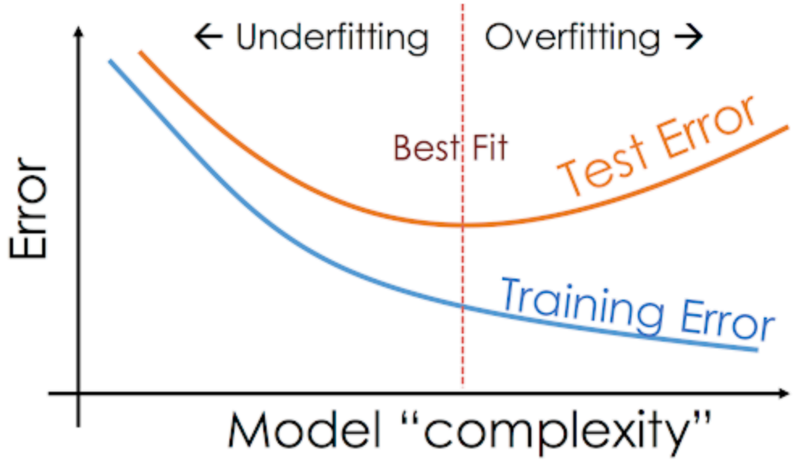
Learning: Validation



- Validation:
  - To tune the hyperparameters of the model
  - To prevent overfitting
- Hyperparameters: Pre-defined model characteristics
  - E.g., Regularization, Depth of network, Learning rate, etc.

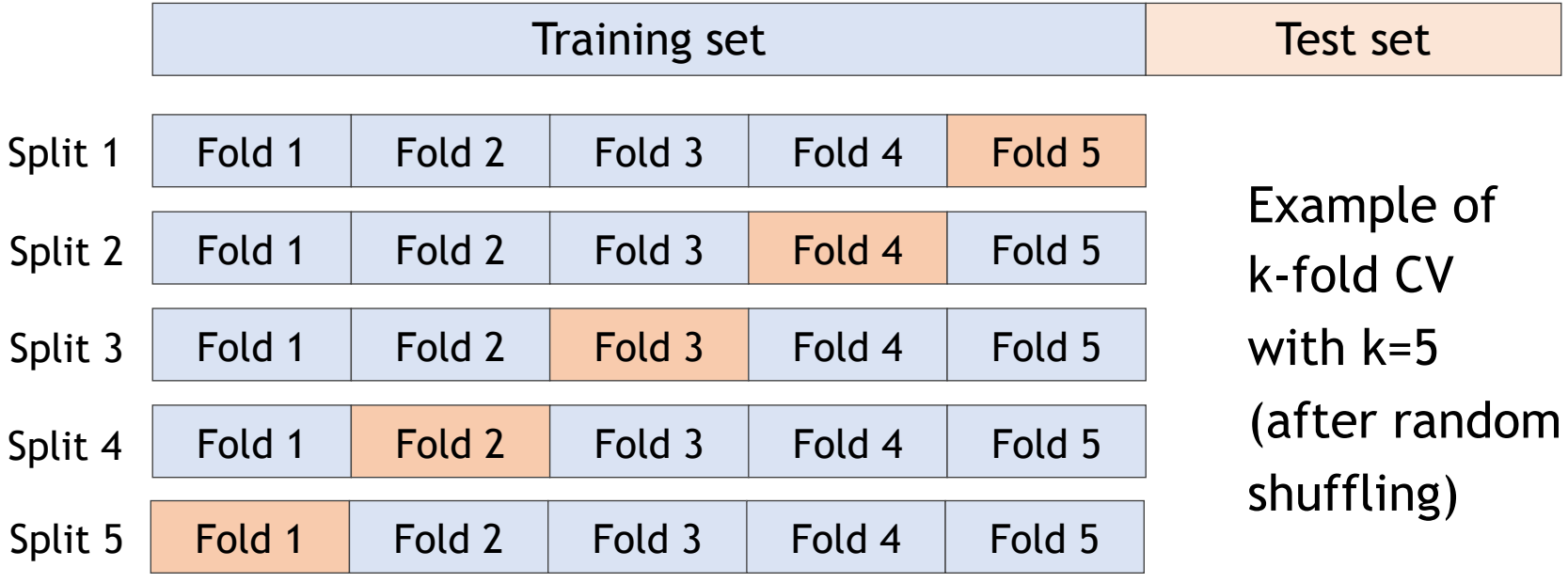
Learning: Underfit vs. Overfit





- Overfitting: Working very well with training set, but not with test set

Learning: Cross-Validation



Example of k-fold CV with k=5 (after random shuffling)

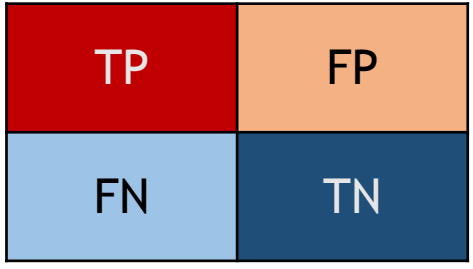
- Effective hyperparameter tuning for limited size of training set
  - Number of k varies depending on the size, usually 5 to 10

Evaluation: Performance Metrics

- **Confusion Matrix**: Stat. for model performance

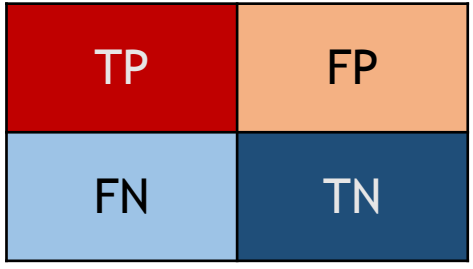
2x2 Simple confusion matrix		Event Observed (or Label)	
		Positive	Negative
Event Prediction by model	Positive	True Positive (hits)	False Positive (false alarms)
	Negative	False Negative (misses)	True Negative (correct negatives)

Evaluation: Performance Metrics



- **Accuracy**= True/All = (TP+TN)/All
- **Precision**= True Positive/Positive = TP/(TP+FP)
- **False Alarm Ratio (FAR)**= False Positive/Positive = FP/(TP+FP) = 1-Precision
- **Recall** or **Sensitivity** or **Probability of Dection(POD)** = hits when event occurred = TP/(TP+FN)

Evaluation: Performance Metrics



- **F1 score** = harmonic mean of Precision and Recall = $2\frac{Precision \times Recall}{Precision+Recall} = \frac{TP}{TP+(FP+FN)/2}$ ← Good for imbalanced data
- **Critical Success Index (CSI)** or **Threat Score** = $\frac{TP}{TP+FP+FN}$
- **Peirce Skill Score (PSS)** = $\frac{TP}{TP+FN} - \frac{FP}{FP+TN}$

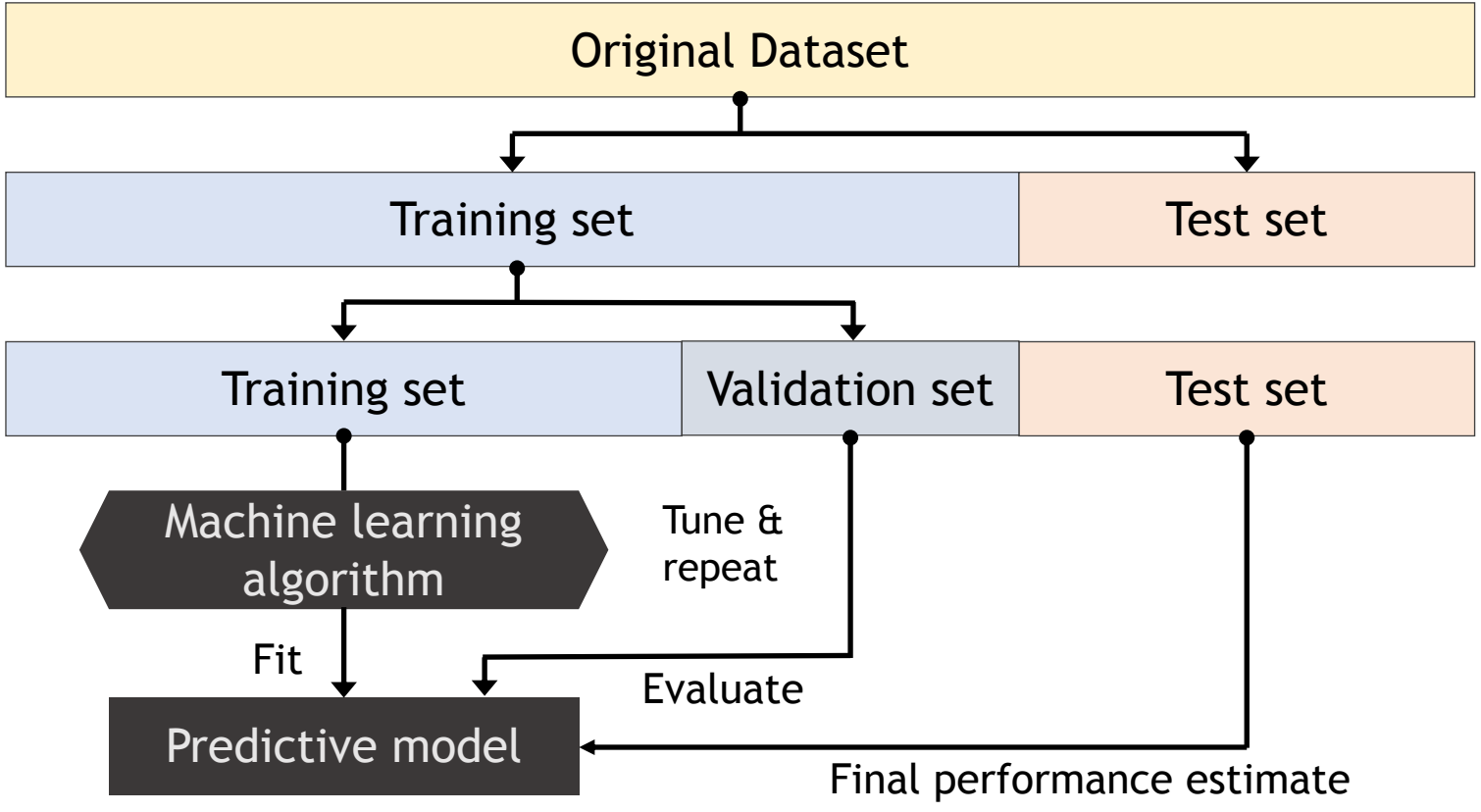
Evaluation: Performance Metrics

Case 1		Label (Truth)	
		Storm	No
Prediction by model	Storm	45	10
	No	5	940

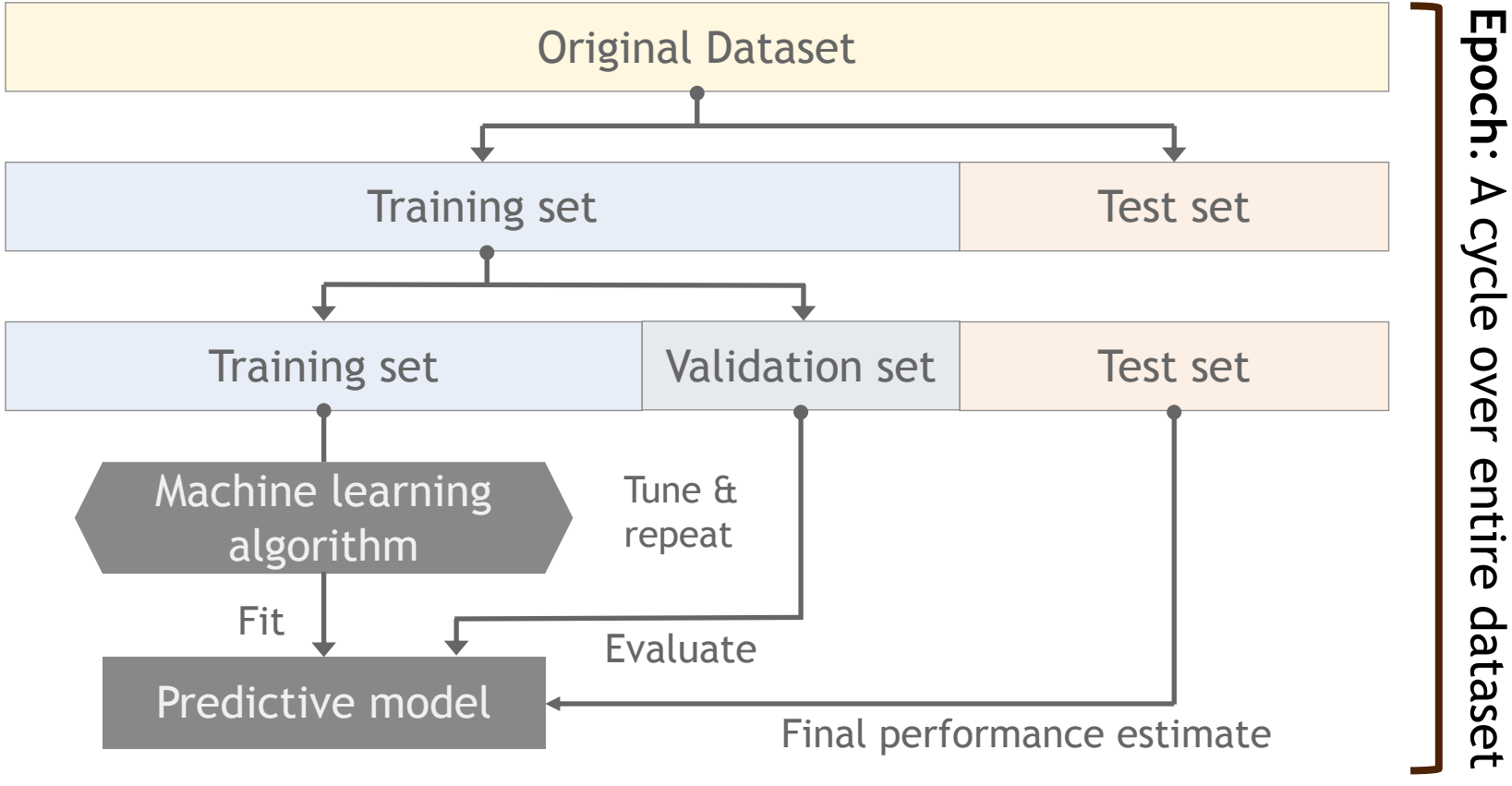
Case 2		Label (Truth)	
		Cat	Dog
Prediction by model	Cat	45	10
	Dog	5	40

Metrics	Case 1	Cat	Dog
Accuracy	0.985	0.850	0.850
Precision	0.818	0.818	0.889
FAR	0.182	0.182	0.111
Recall (POD)	0.900	0.900	0.800
F1	0.857	0.857	0.842
CSI	0.750	0.750	0.727
PSS	0.889	0.700	0.700

Modeling Process

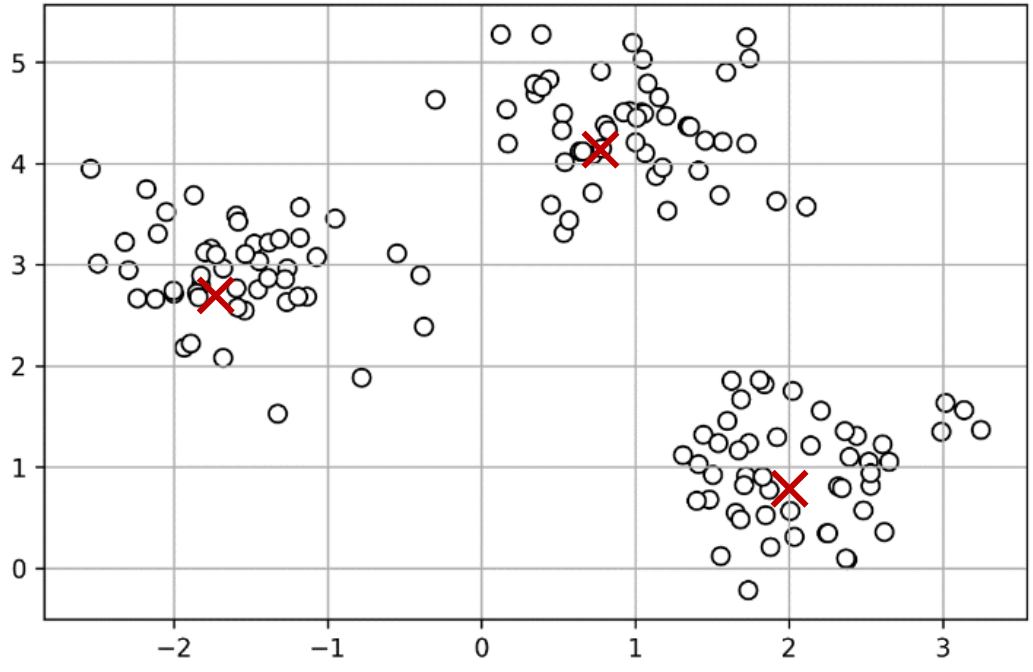


Evaluation: Epoch



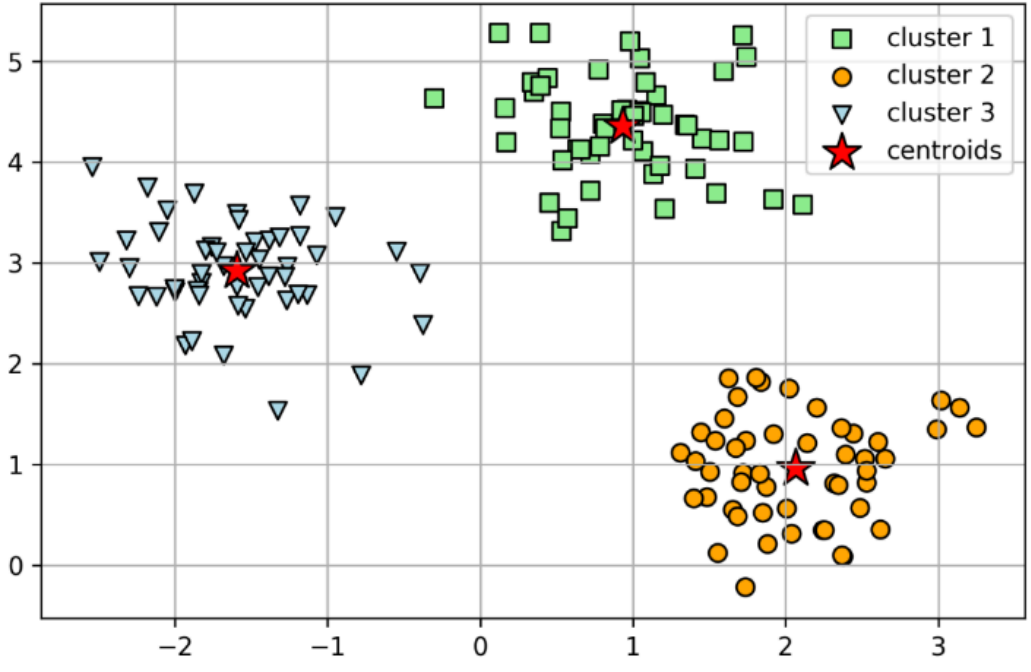
ML Algorithm
Clusturing

ML Algorithm: *k*-means



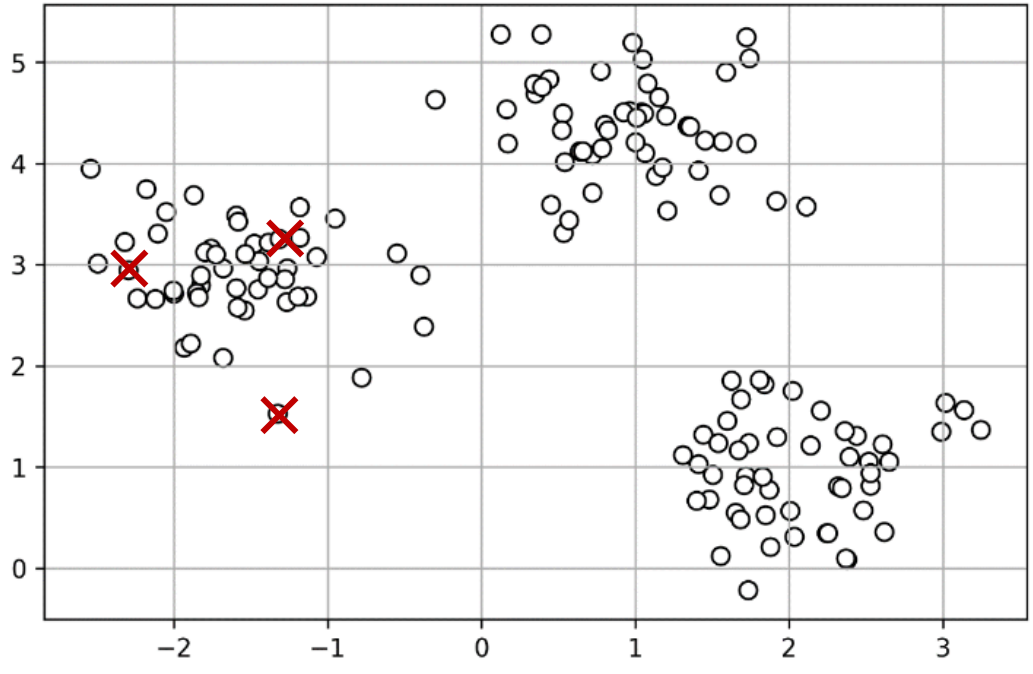
- Assign member to the closest centroid
- New centroid is the mean of all assigned members
- Check how much the new centroid moved
- Repeat until (nearly) no movement

ML Algorithm: *k*-means



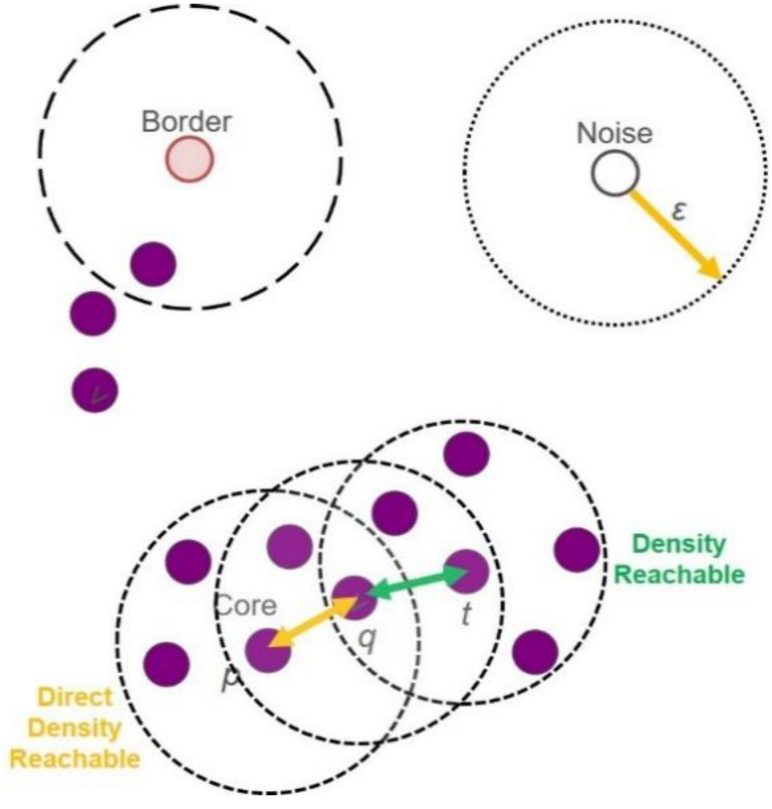
- Need to evaluate if the selected number, k is optimal.
- **Calinski-Harabasz** Index: weak for very large data
- **Davies-Bouldin** index: generally good performance

ML Algorithm: *k*-means



- If the initial starting points are in bad location?
- Cannot be sure how the final result looks like.
- **k-means++**: Optimal initiation strategy
- **More trials, better chance** for optimal results

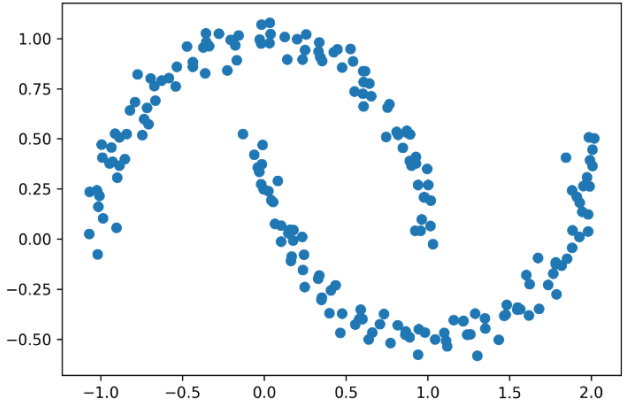
ML Algorithm: DBSCAN

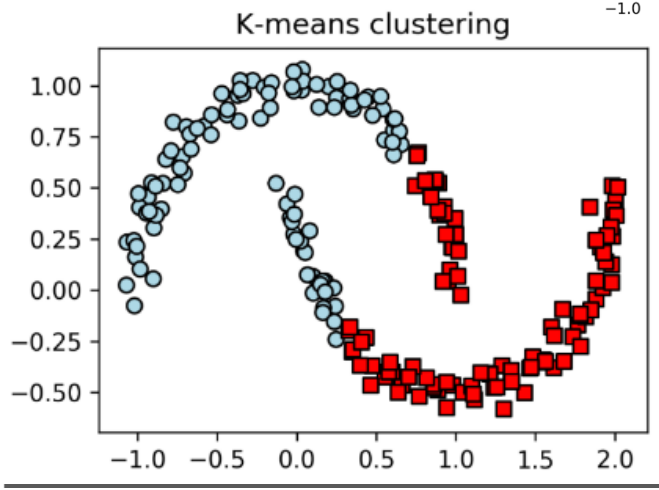


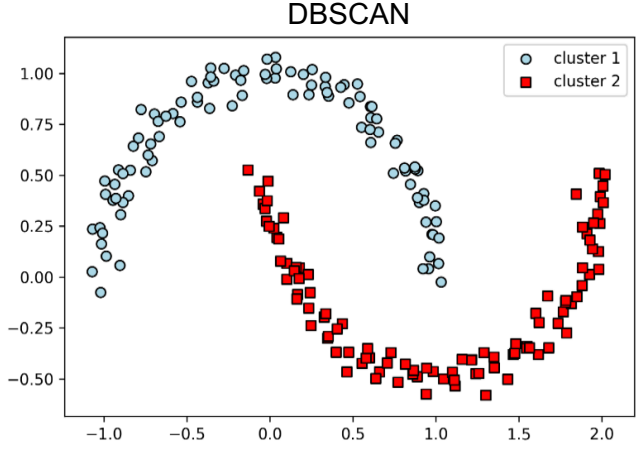
- Density-based spatial clustering of applications with noise (DBSCAN)
- Need to calculate distance between all available pairs
- It can be speed-up by building distance table, then need n(n-1)/2×m Byte memory space

https://www.mdpi.com/2076-3417/9/20/4398/htm

ML Algorithm: *k*-means vs. DBSCAN







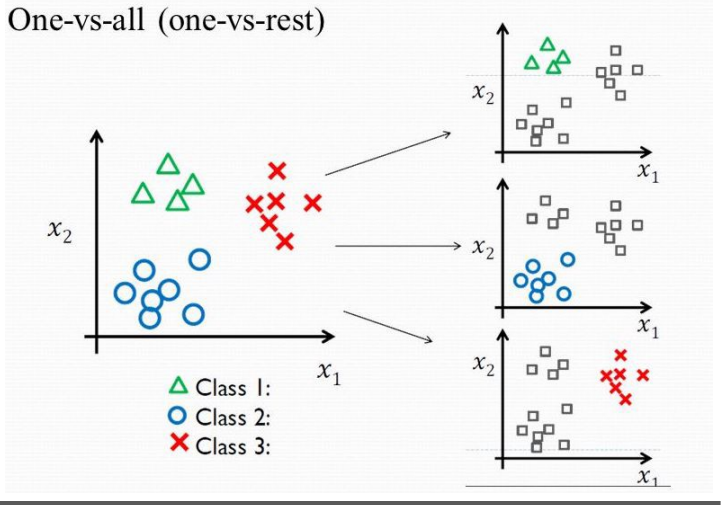
ML Algorithm: Clustering

- *k*-means
  - Top-down
  - Need to set the number of clustered group
  - Every member is included one of cluster
  - Deterministic and heavily depending on initiation
  - Be cautious for floating point overflow
- DBSCAN
  - Bottom-up
  - Need to set minimum distance criterion for boundary
  - Outliers may not be included in any cluster
  - Not entirely deterministic and less depending on the initiation
  - Be cautious for memory overflow

ML Algorithm
Multi-class Classification

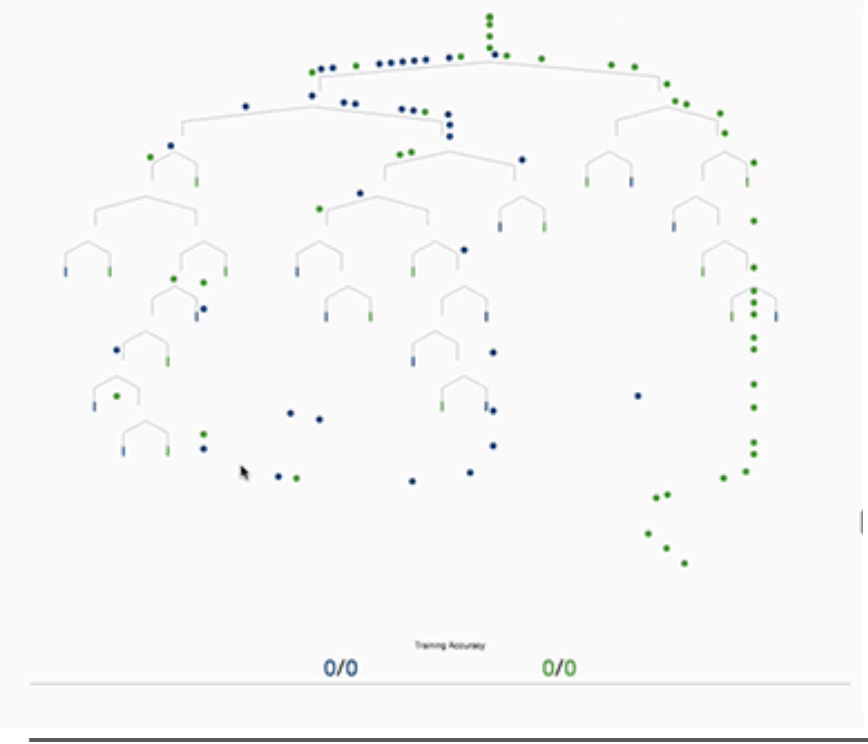
ML Algo: Multi-class Classification

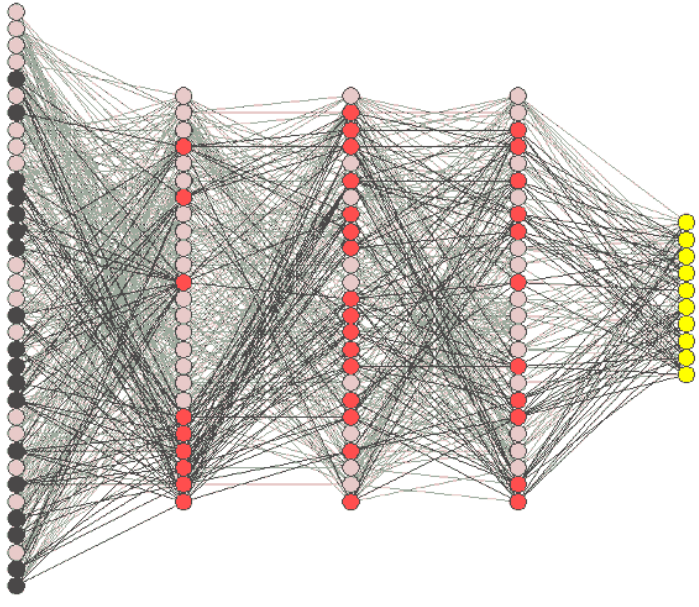
- Binary classification; e.g., Cat vs. Dog
- Some algorithms like SVM or Logistic Regression do not support multi-class classification; then how to do?
- Strategy 1: One-vs-One (OvO)
  - Perform binary classification for all available pairs of classes
- Strategy 2: One-vs-All (OvA; or One-vs-Rest [OvR])



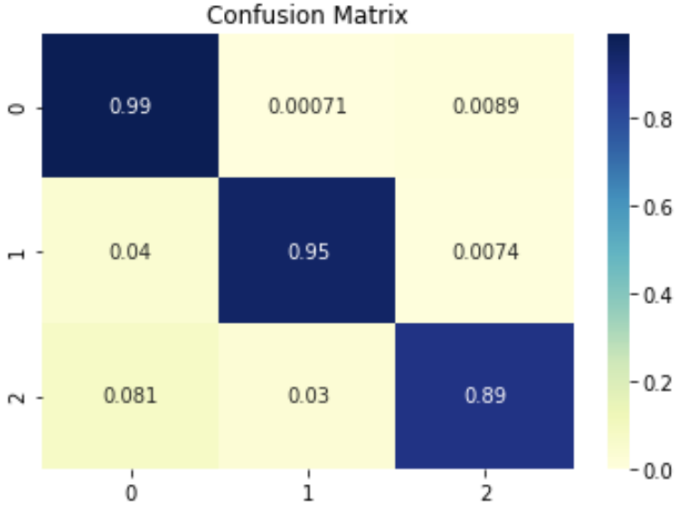
ML Algo: Multi-class Classification

- Of course, some algorithms naturally support multi-class





ML Algo: Multi-class Classification

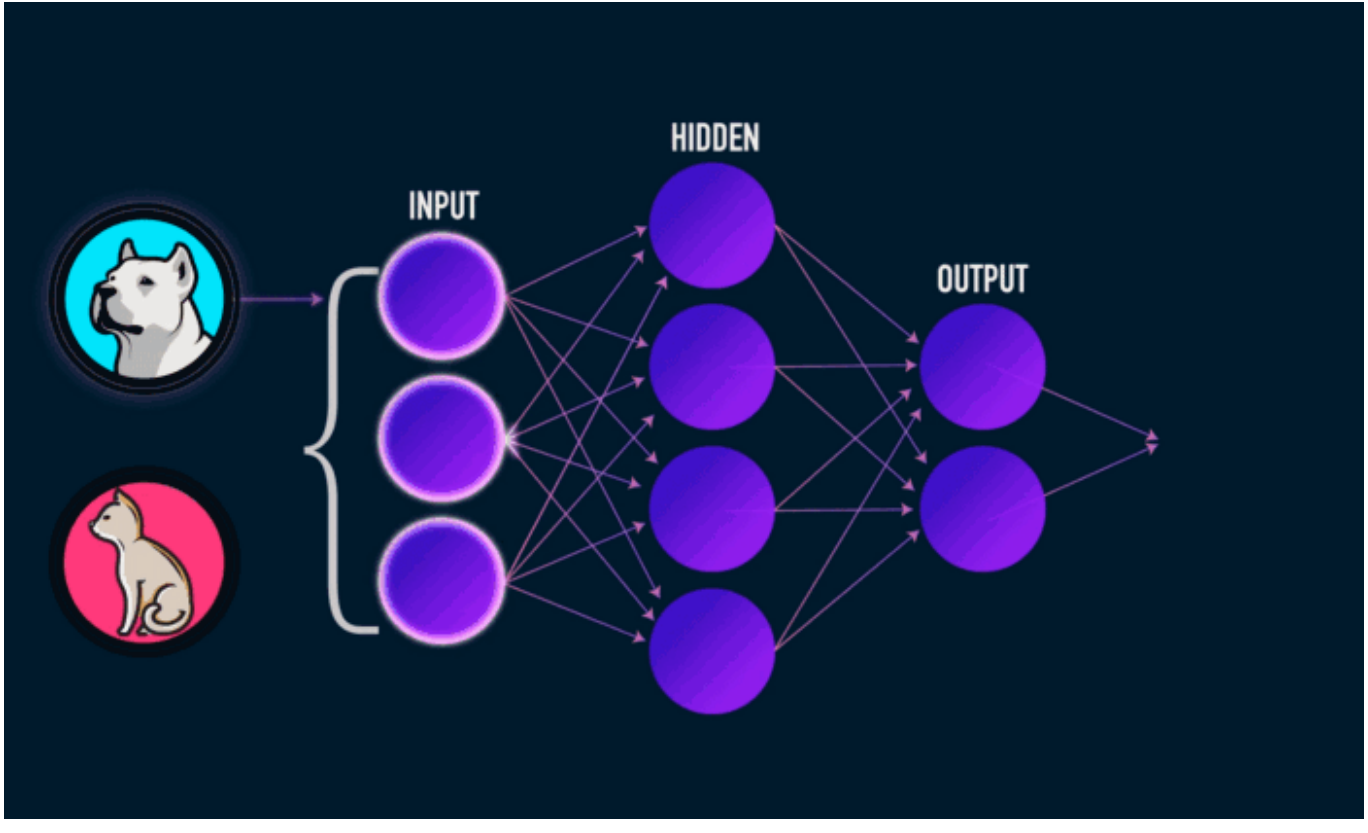


	precision	recall	f1-score	support
0	0.97	0.99	0.98	5608
1	0.99	0.95	0.97	2698
2	0.90	0.89	0.89	694
micro avg	0.97	0.97	0.97	9000
macro avg	0.95	0.94	0.95	9000
weighted avg	0.97	0.97	0.97	9000

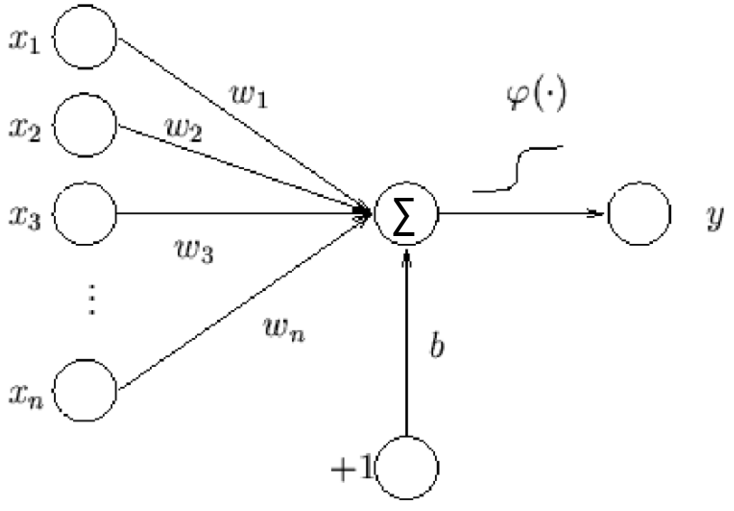
- Micro average: Not by class, but for each data entry
- Macro average: Simple average for each class
- Weighted average: Weighted average of each class score by fraction of each label

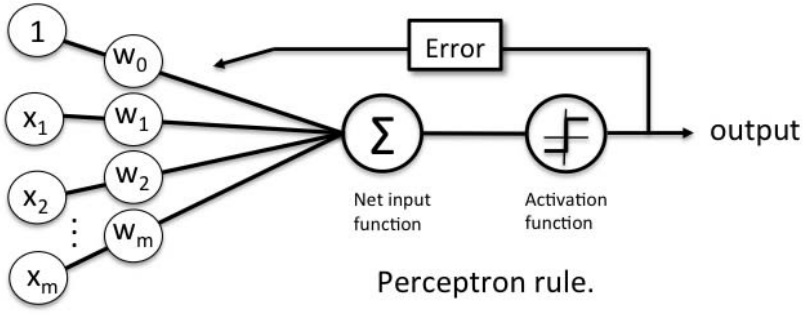
ML Algorithm
(Artificial) Neural Network

ML Algorithm: (Artrifitial)Neural Net



ML Algorithm: NN-Perceptron



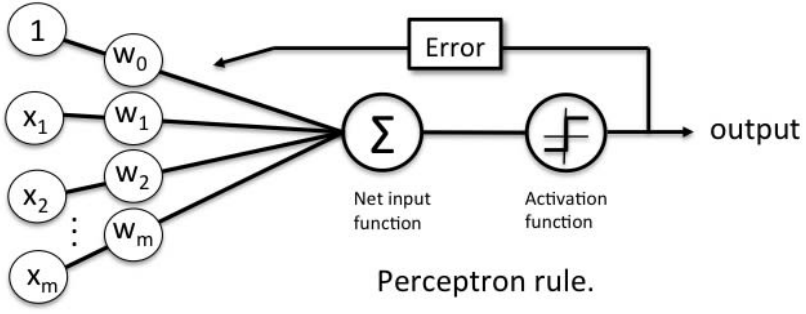


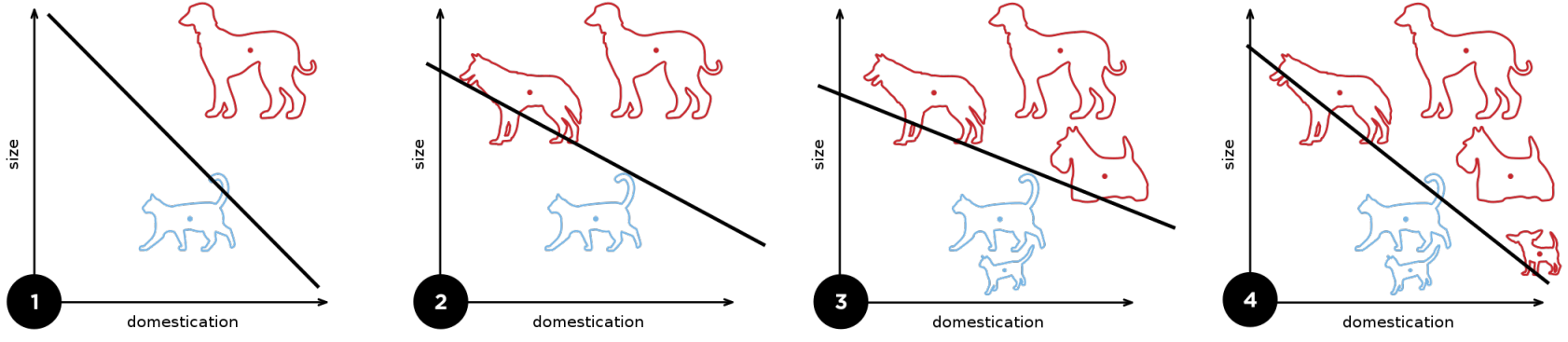
- Input
- Weight
- Bias

- Activation Function
- Back-Propagation

ML Algorithm: NN-Perceptron

- Perceptron is basically "Linear" regression
- $y = \sum_{i=1}^{n} w_i x_i + w_0$
- Bias, w_0: the role of y-intercept

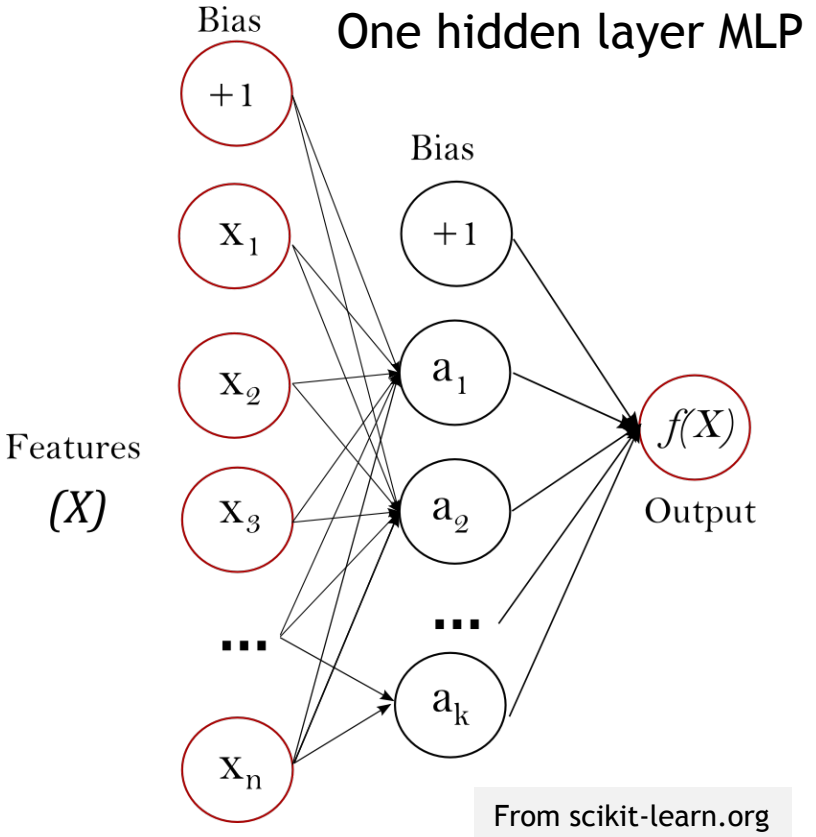




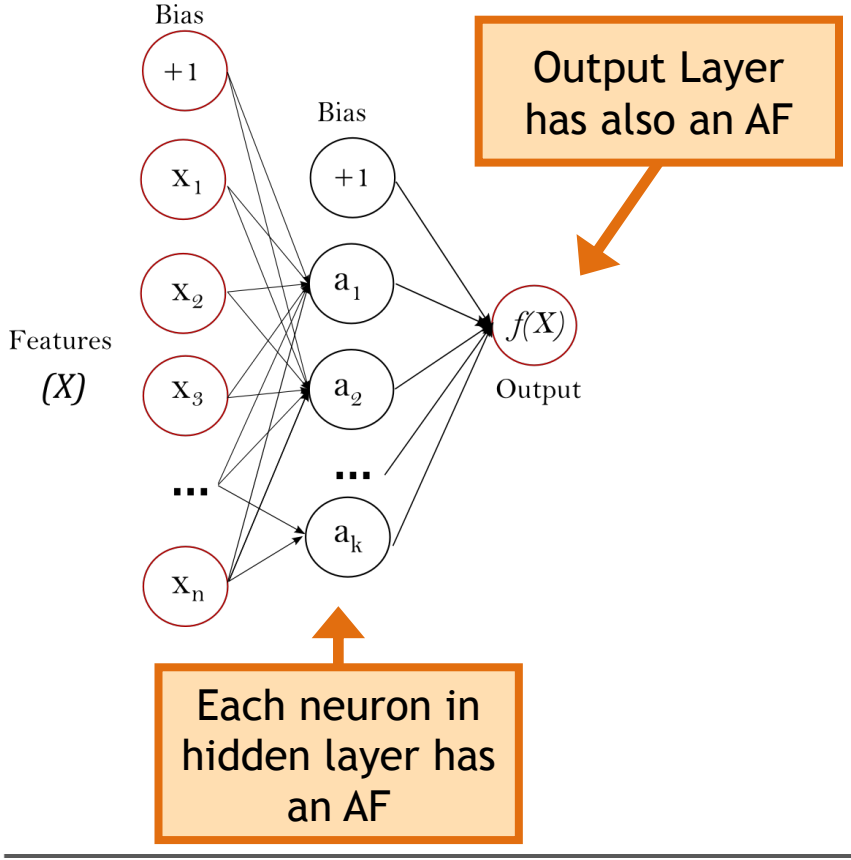
By Elizabeth Goodspeed - Own work, CC BY-SA 4.0
https://commons.wikimedia.org/w/index.php?curid=40188333

ML Algorithm: NN-MLP

- Multi-layer Perceptron (MLP) is "**Non-linear**" regression with Activation Function
- Also called Feed-forward Neural Net
- Basic learning algorithm, "**Back-Propagation**," is same to the simple Perceptron



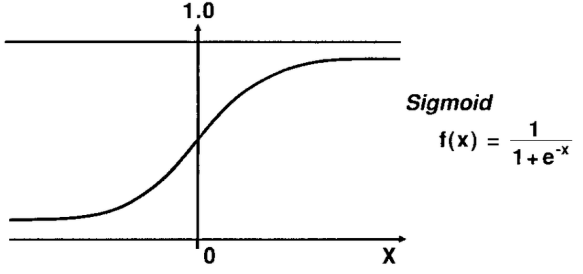
ML Algorithm: NN-MLP

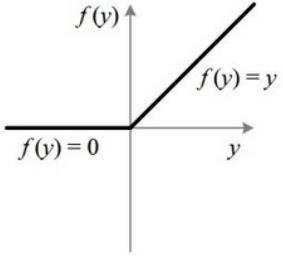


- Activation Function (AF)
- Collect information from previous layer, and transfer a value to next layer in **manageable range**
- **Derivative of AF** is used for back-propagation procedure

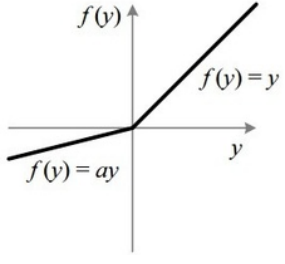
ML Algorithm: Activation Function

- Step function: can not be used for back-propagation
- Linear function: Used for regression output layer
- Sigmoid: Good for binary classification output layer
- Softmax: Good for multi-class classification output layer
- (Leaky) ReLU: Good for hidden layer with computational efficiency.
- Sigmoid and Tanh have vanishing gradient problem (not good for hidden layer)





ReLU

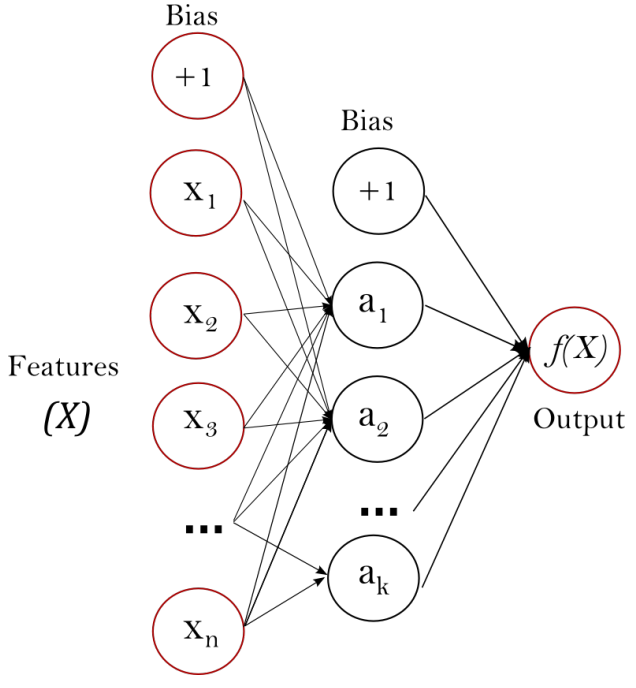


Leaky ReLU

ML Algorithm
Convolution Neural Network

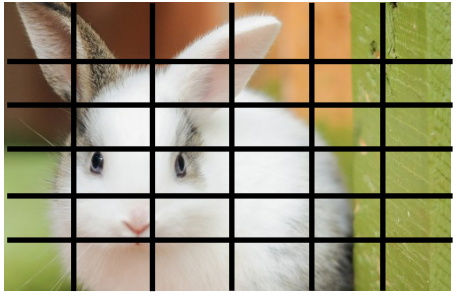
ML Algorithm: Convolution NN

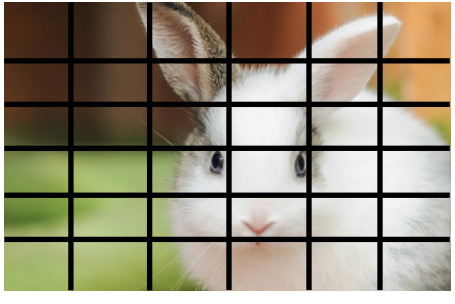
- Why need “Convolution Neural Network (CNN)”?
- For the classification of image…

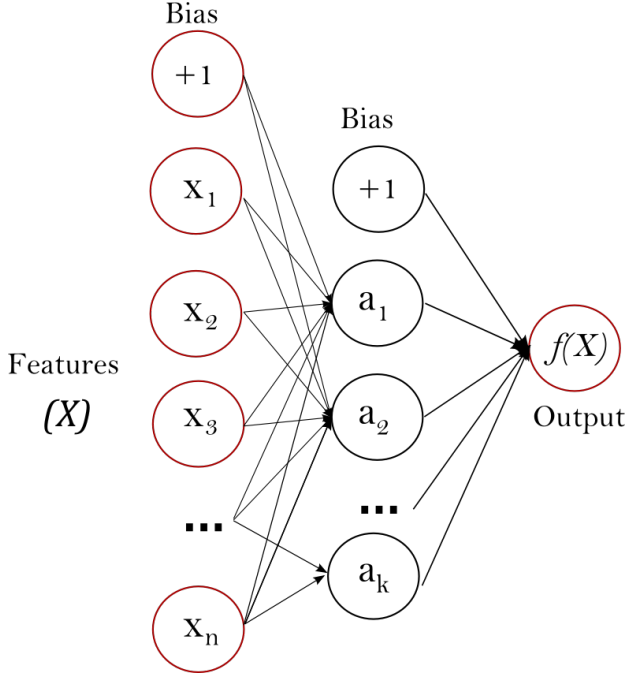


ML Algorithm: Convolution NN

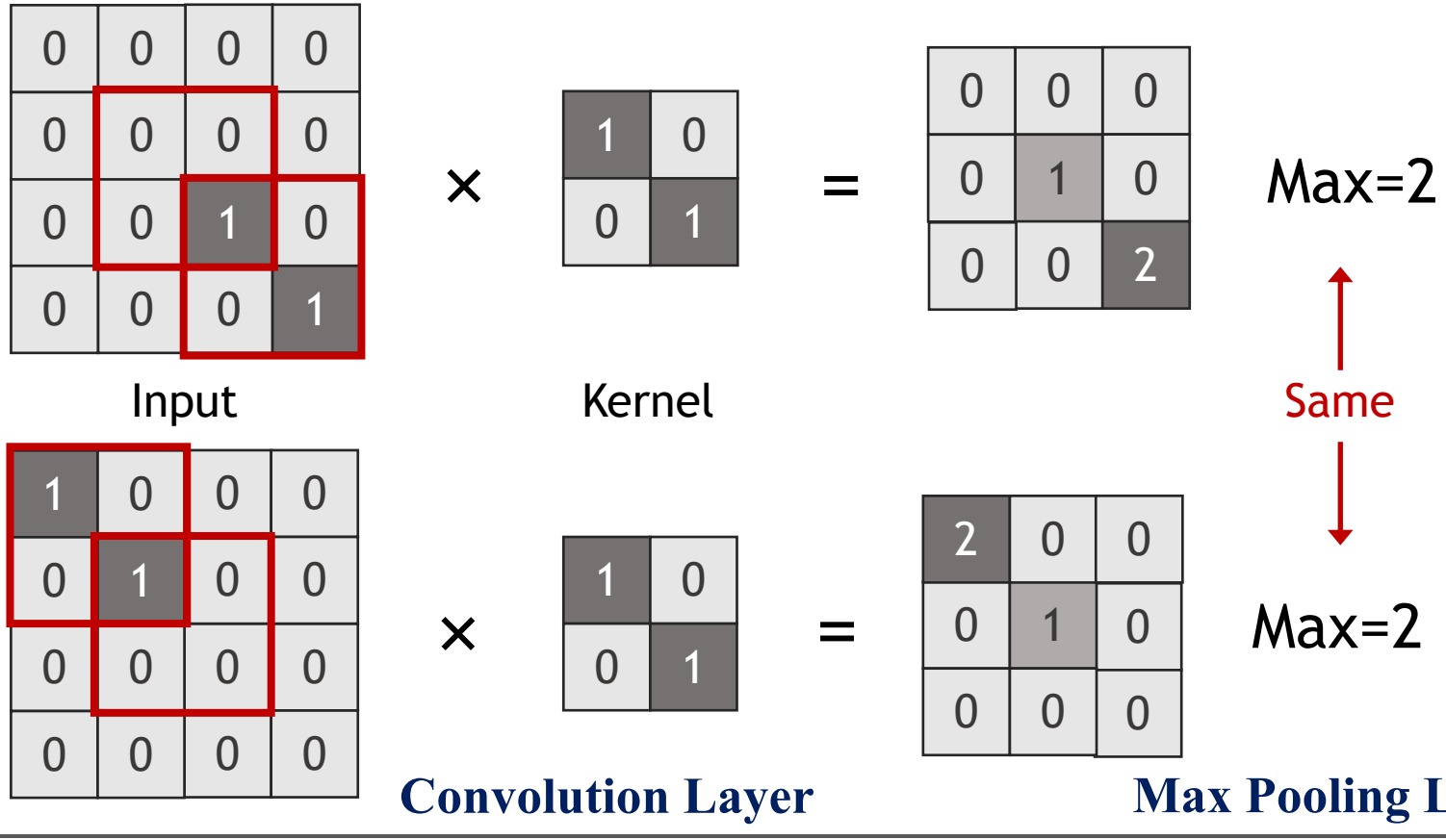
- Why need “Convolution Neural Network (CNN)”?
- For the classification of image…
- Problem with MLP:
  - Need too large input dimension
  - Cannot detect the subject
    if it is in different location





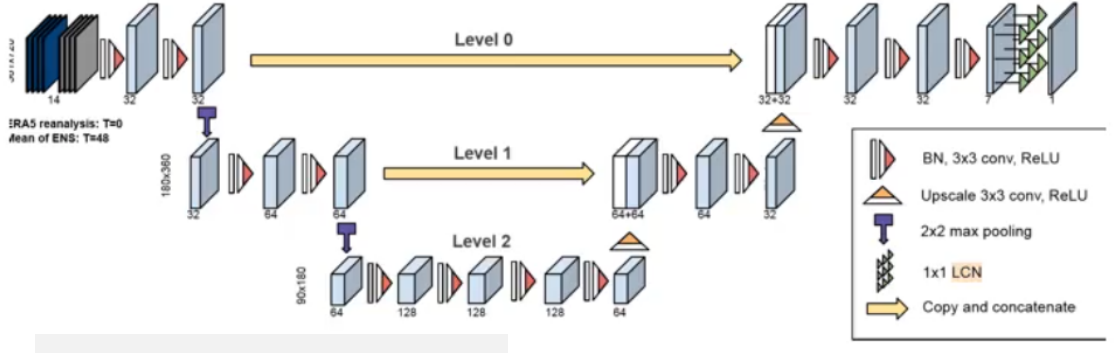


ML Algorithm: Convolution NN



ML Algorithm: Other NN

- For time series
  - Recurrent Neural Network(RNN)
  - Long Short-Term Memory(LSTM)
  - …
- For 2-D to 2-D
  - U-Net

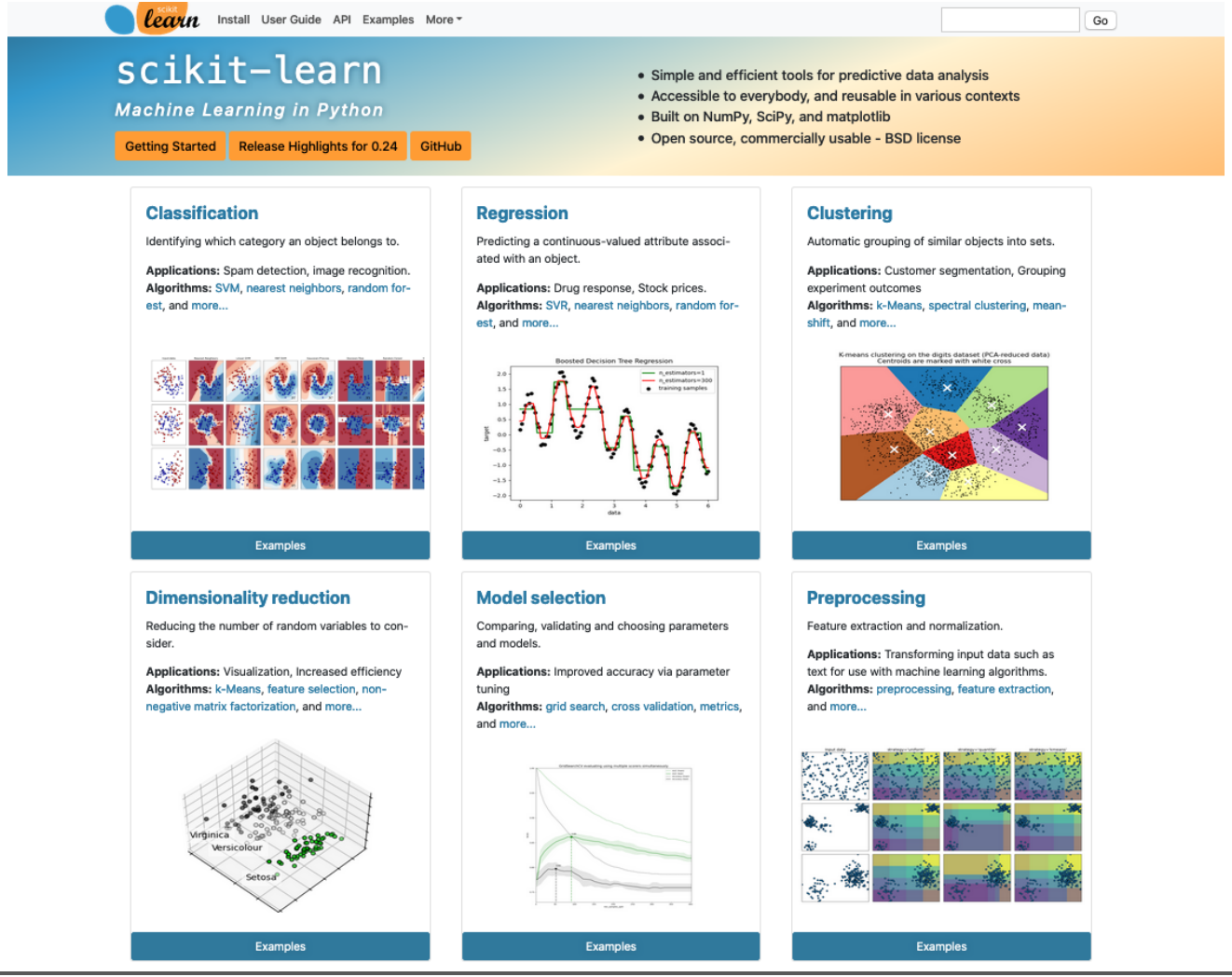


arxiv.org/abs/2005.08748

Coding Practice

Data and Problem Summary

- Input Data
  - Outgoing Longwave Radiation(OLR) from NOAA CDR
  - Monthly, 2.5° resolution, 1979.01-2020.12
- Label
  - Monthly Niño3.4 index
- Problem
  - 3-month forecast of [El Niño / Neutral / La Niña] using OLR data (tropics only for convenience)
- Strategy
  - Training MLP model with 12 values of 30°×30° box mean of OLR
  - Last 5 years (60 months) are reserved for testing.


Install
User Guide
API
Examples
More
Go
scikit-learn
Machine Learning in Python
Getting Started
Release Highlights for 0.24
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...
Examples
Regression
Predicting a continuous-valued attribute associated with an object.
Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...
Boosted Decision Tree Regression
Examples
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
Dimensionality reduction
Reducing the number of random variables to consider.
Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...
Virginica
Versicolour
Setosa
Examples
Model selection
Comparing, validating and choosing parameters and models.
Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...
Examples
Preprocessing
Feature extraction and normalization.
Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...
Examples

MLPClassifier in Scikit-learn

sklearn.neural_network.MLPClassifier

class `sklearn.neural_network.` **`MLPClassifier`**(*hidden_layer_sizes=100, activation='relu', *, solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000*) [source]

Multi-layer Perceptron classifier.

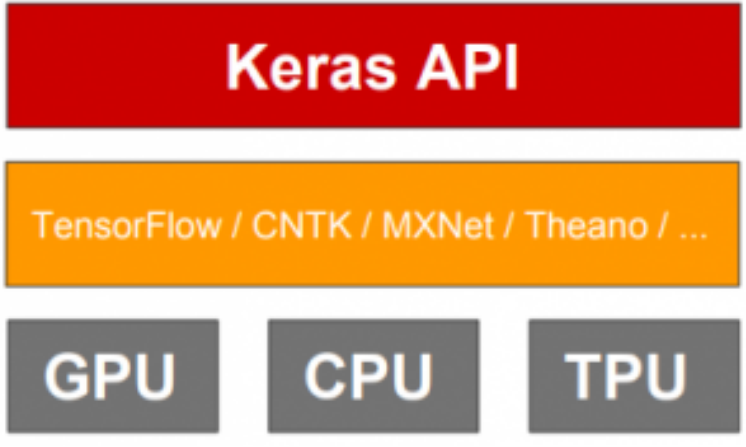
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Programs

1. Simple example
2. Simple version + One-Hot encoding
3. Simple version + Balancing samples
4. GridSearchCV
5. Ensemble with Kfold
6. Save and Load model settings


TensorFlow


K
Keras

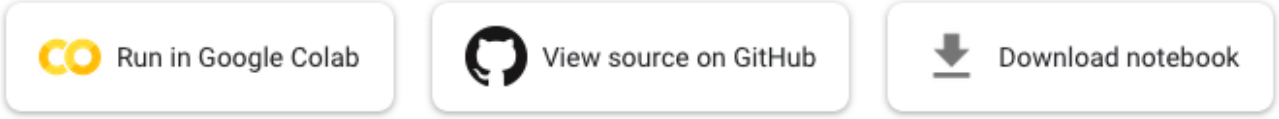

Keras API
TensorFlow / CNTK / MXNet / Theano / ...
GPU
CPU
TPU

Sequential model in TensorFlow

TensorFlow > Learn > TensorFlow Core > Guide

☆☆☆☆☆

The Sequential model



Setup

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

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

Programs

- Simple MLP with TensorFlow+Keras
- Simple CNN with TensorFlow+Keras