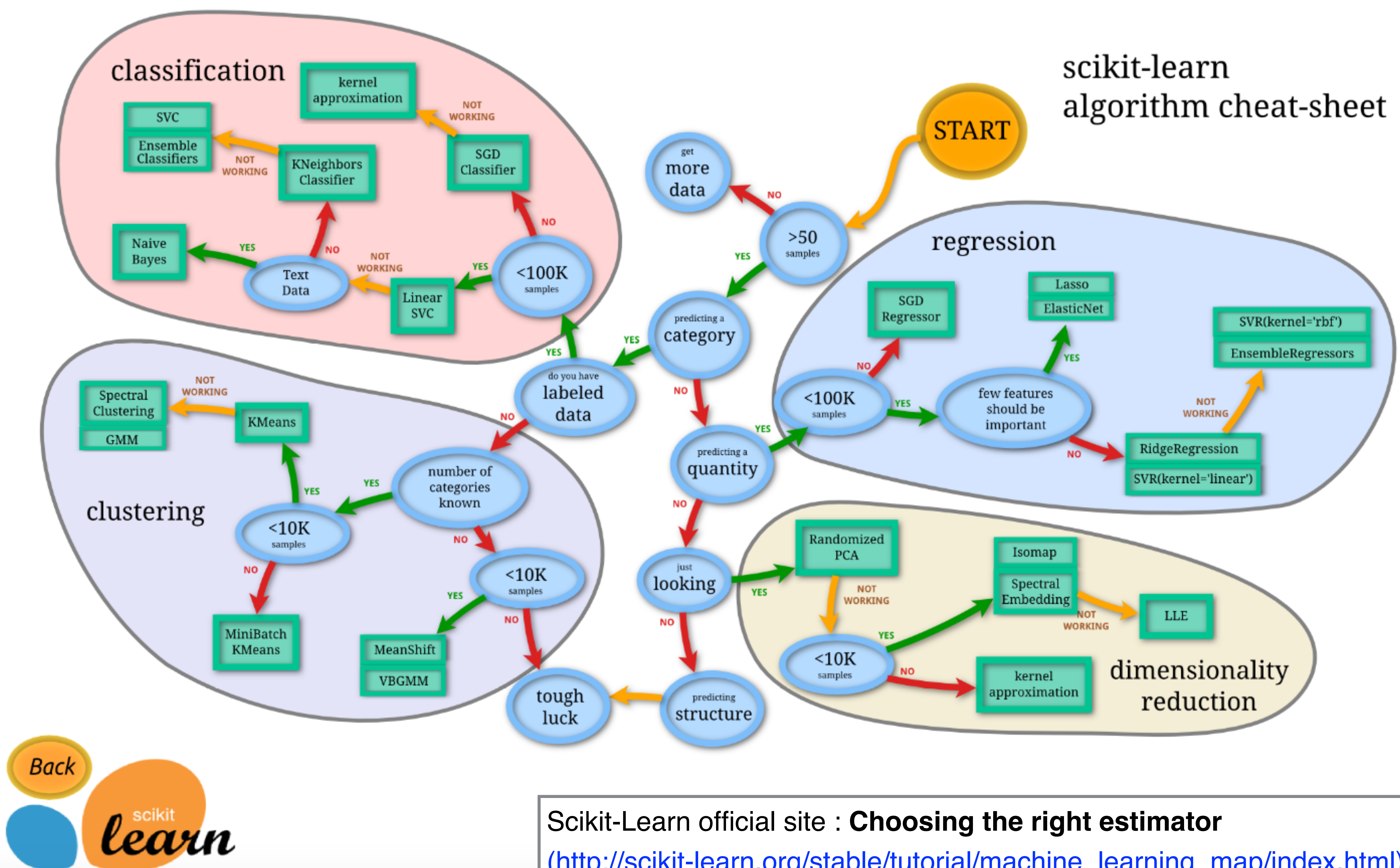
Machine Learning Workshops

< Scikit-Learn >

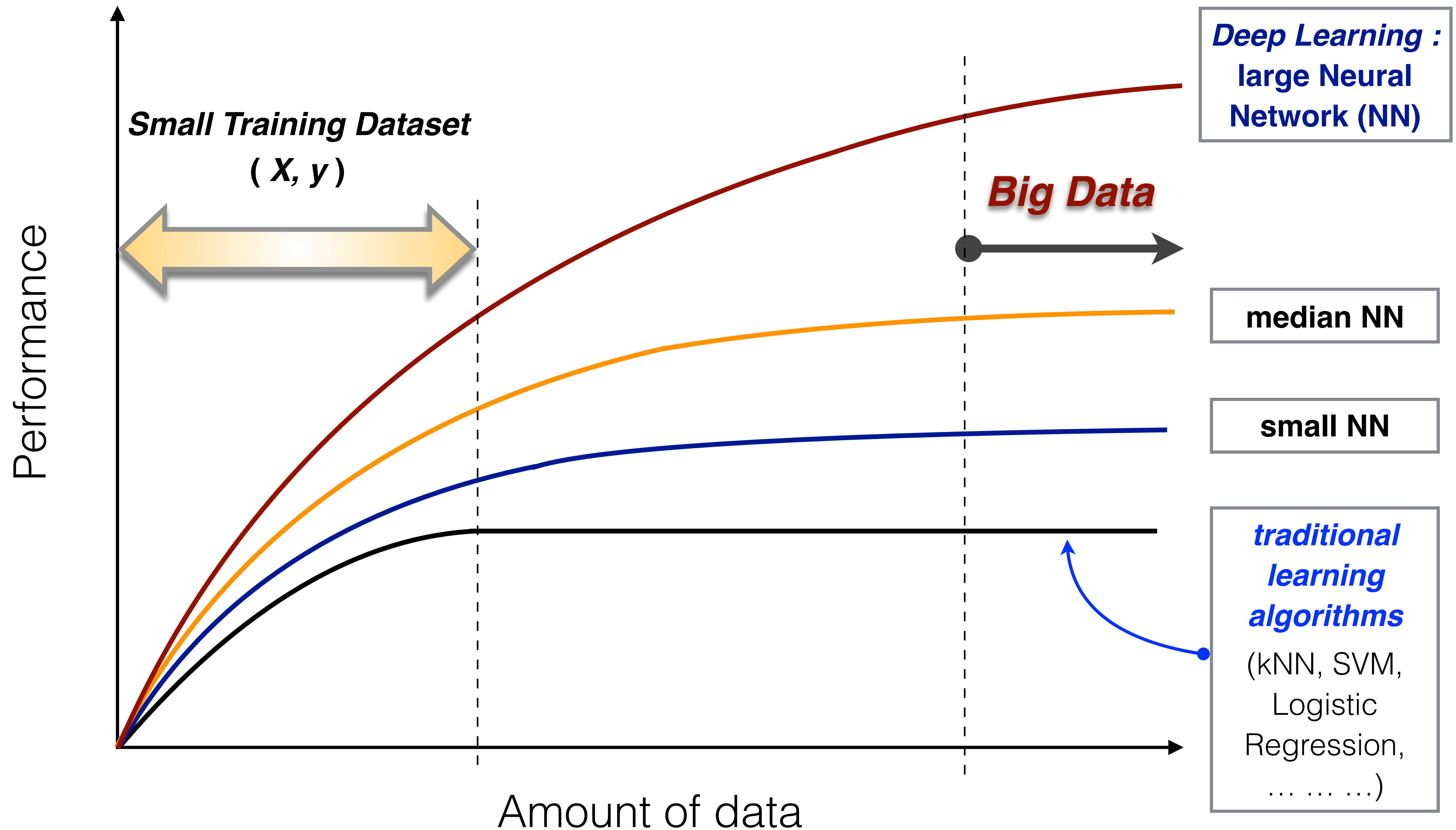
C. Alex Hu, PhD

Choosing the right estimator (選擇適當的估測器或演算法)



Scikit-Learn official site : **Choosing the right estimator**
[(http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html)](http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html)

After Scikit-Learn … Next, **Deep Learning.**

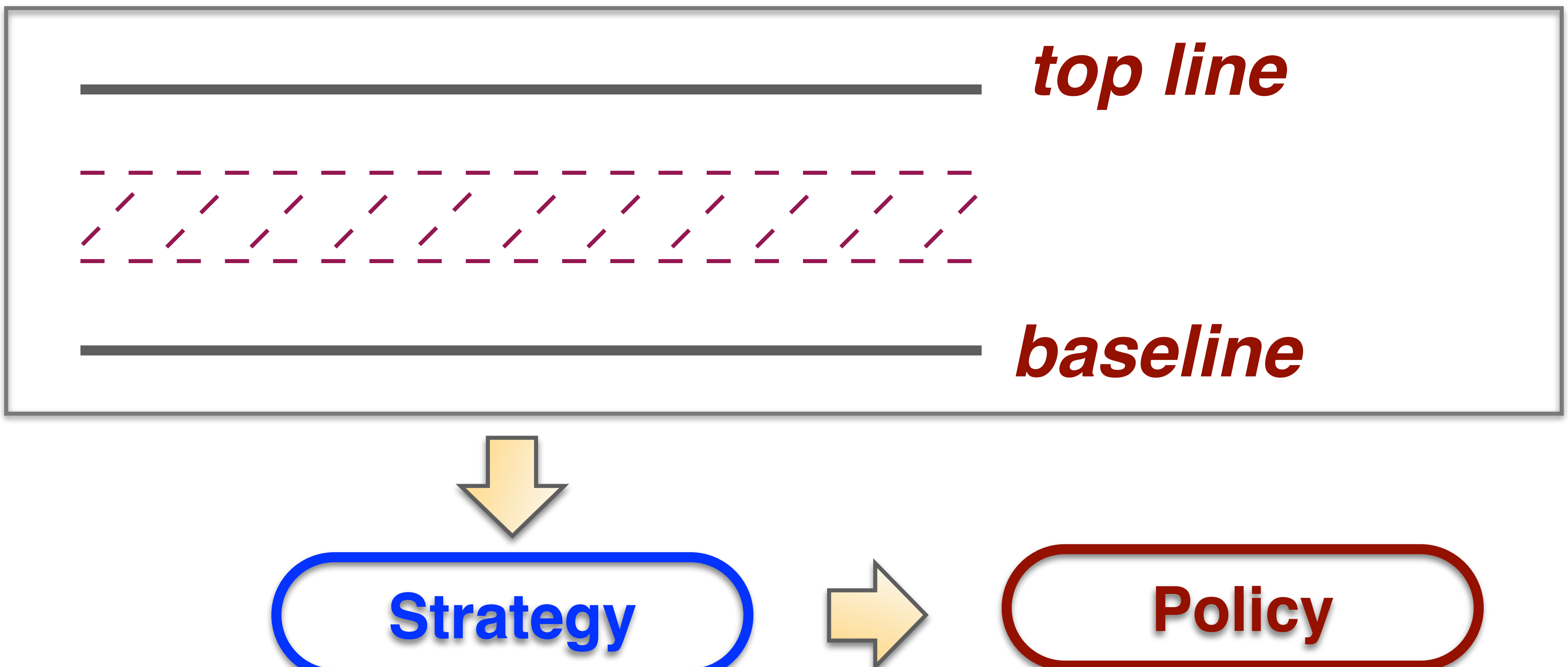


Deep Learning by Andrew Ng (吳恩達) [Full Course] — YouTube Video :
4. Drivers Behind the Rise of Deep Learning (https://youtu.be/j4-QFpTVCtM)

Data Analysis for Finding **Strategy to Resolving Problems**

1. Small Dataset => Sklearn Models => Baseline

2. Big Data => Predictive Models => Risk Analysis

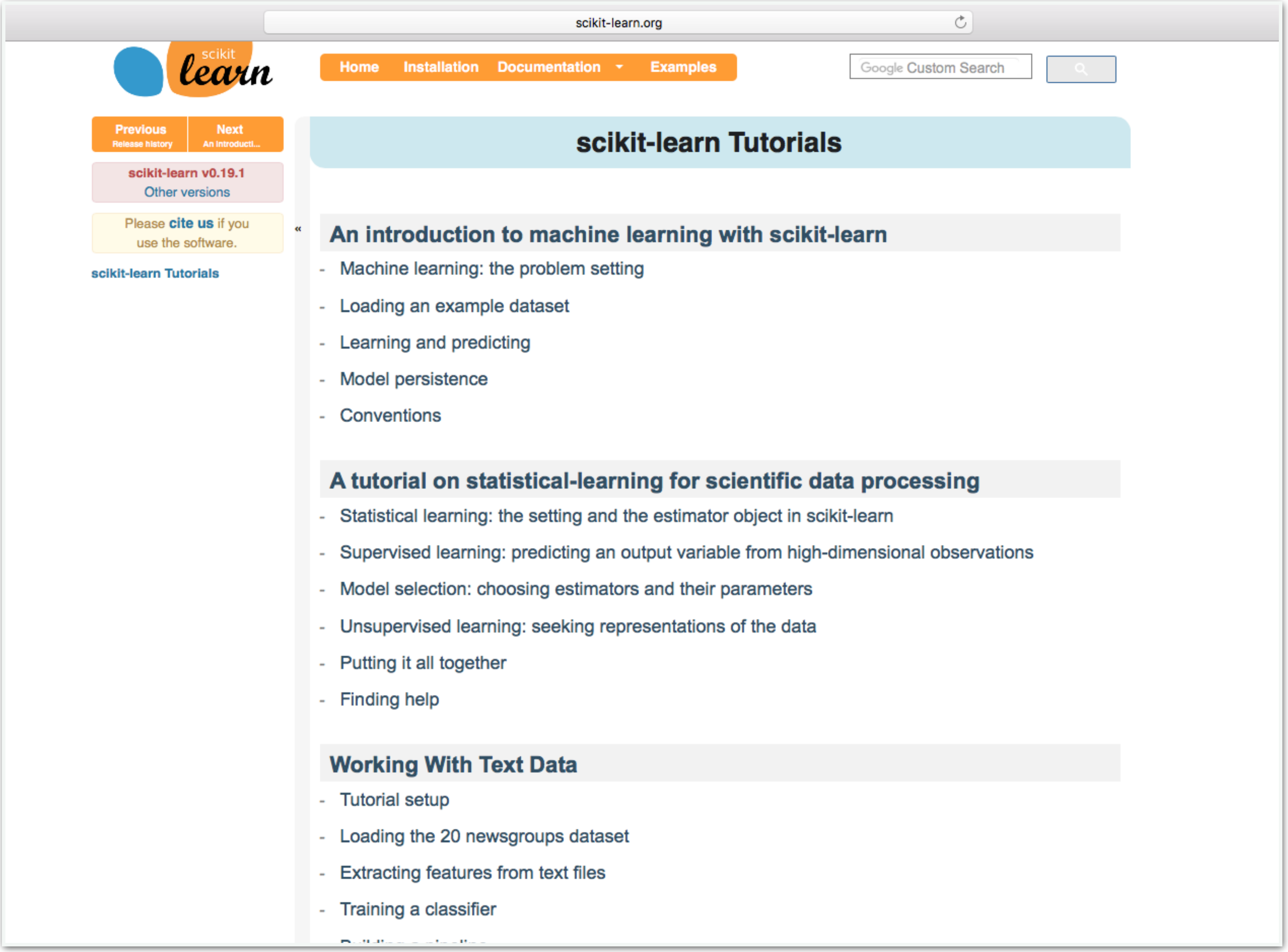


3. Policy => Execution => Risk Management

scikit-learn Tutorials

Sklearn Tutorials : http://scikit-learn.org/stable/tutorial/index.html

Code downloading : https://github.com/scikit-learn/scikit-learn



Scikit-Learn Workshops

[Scikit-Learn Workshop 1] : 監督式學習 — 迴歸模型

[Scikit-Learn Workshop 2] : 監督式學習 — 分類模型

[Scikit-Learn Workshop 3] : Validation Curves & Learning Curves

[Scikit-Learn Workshop 4] : 非監督式學習 — 集群分析

[Scikit-Learn Workshop 5] : 支持向量機 **(for Both)**

[Scikit-Learn Workshop 6] : Neural Networks **(for Both)**

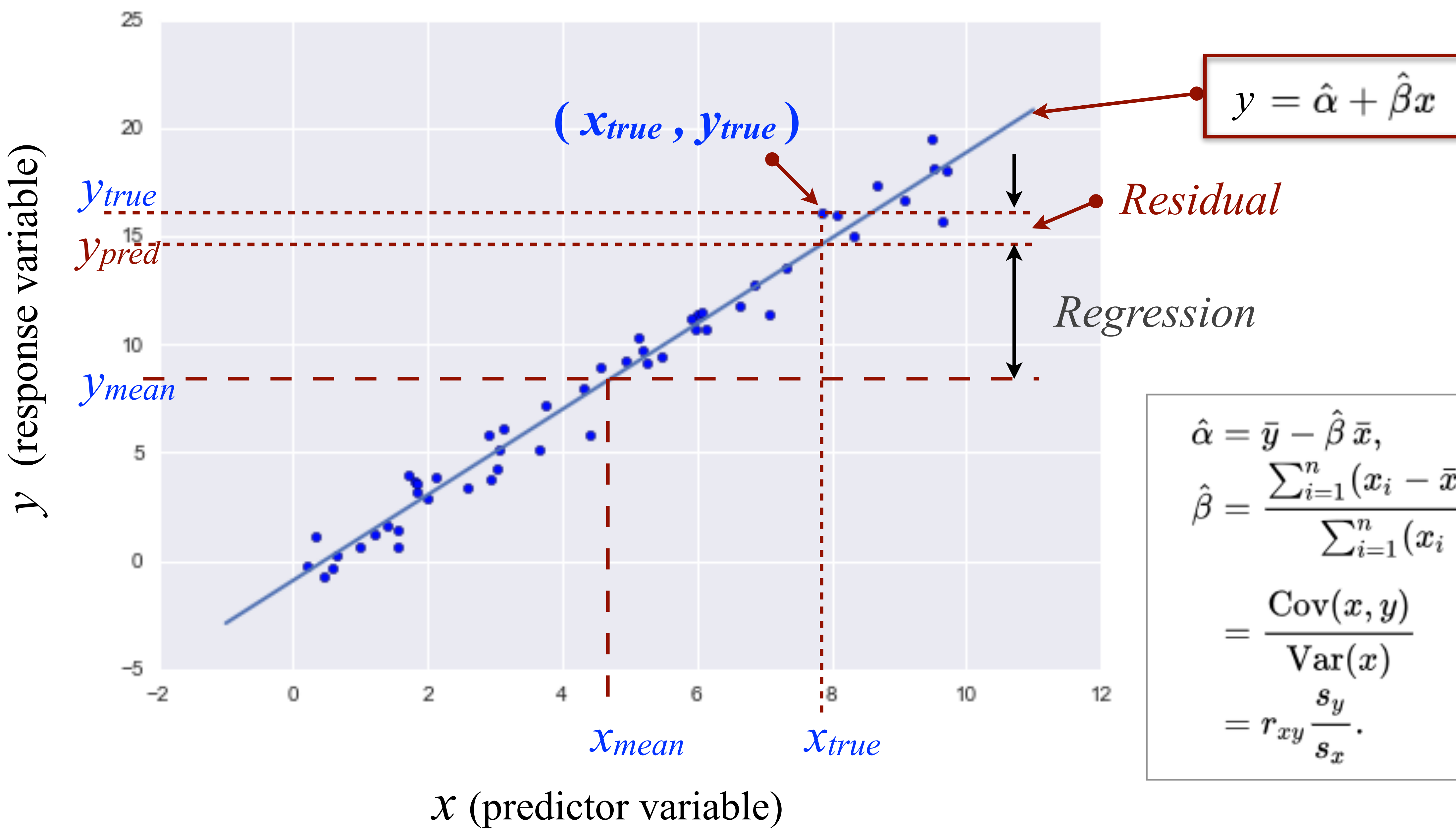
[Scikit-Learn Workshop 7] : Meta-Learner - Random Forests

Scikit-Learn Workshop 1：監督式學習 — 迴歸模型

A. Simple Linear Regression

(https://en.wikipedia.org/wiki/Simple_linear_regression)

from sklearn.**linear_model** import **LinearRegression**



$$\hat{\alpha} = \bar{y} - \hat{\beta}\,\bar{x},$$

$$\hat{\beta} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$= \frac{\mathrm{Cov}(x, y)}{\mathrm{Var}(x)}$$

$$= r_{xy}\frac{s_y}{s_x}.$$

B. Basis Function Regression

(Ref : "05.06-Linear-Regression" from the Python Data Science Handbook by Jake VanderPlas)

— One trick you can use to adapt linear regression to nonlinear relationships between variables is to transform the data according to ***basis functions***.

— **The idea** is to take our multidimensional linear model:

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 + \cdots$$

and build the x_1 , x_2 , x_3 , and so on, from our single-dimensional input x . That is, we let $x_n = f_n(x)$, where $f_n()$ is some function that transforms our data; for examples,

1. **Polynomial basis functions**
2. **Gaussian basis functions**

Scikit-Learn Workshop 1 : 監督式學習 — 迴歸模型 (cont'd)

1. Polynomial basis functions

(Ref : "05.06-Linear-Regression" from the Python Data Science Handbook by Jake VanderPlas)

If $f_n(x) = x^n$, our model becomes a ***polynomial regression*** :

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots$$

Notice that this is still a linear model — the ***linearity*** refers to the fact that the coefficients a_n never multiply or divide each other.

What we have effectively done is taken our one-dimensional x values and projected them into a higher dimension, so that a linear fit can fit more complicated relationships between x and y.

2. Gaussian basis functions

(Ref： "05.06-Linear-Regression" from the Python Data Science Handbook by Jake VanderPlas)

Other basis functions are possible. For example, one useful pattern is to fit a model that is not a sum of polynomial bases, but a sum of Gaussian bases.

If

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

,

our model becomes a ***Gaussian regression*** :

$$y = a_0 + a_1\, f_1(x) + a_2\, f_2(x) + a_3\, f_3(x) + \cdots$$

C. The Issue of "Regularization"

(Ref : "05.06-Linear-Regression" from the Python Data Science Handbook by Jake VanderPlas)

— The introduction of basis functions into our linear regression makes the model much more flexible, but it also can very quickly lead to ***over-fitting.***

— For example, if we choose too many Gaussian basis functions, we end up with results that don't look so good.

— This is typical over-fitting behavior when basis functions overlap: the coefficients of adjacent basis functions blow up and cancel each other out.

— We could limit such spikes explicitly in the model by ***penalizing*** *large values of the model parameters*.

- **Ridge regression (** L_2 **regularization)**
- **Lasso regression (** L_1 **regularization)**

[Ref. "**A Complete Tutorial on Ridge and Lasso Regression in Python**," AARSHAY JAIN, JANUARY 28, 2016 https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/]

Scikit-Learn Workshop 1：監督式學習 — 迴歸模型 (cont'd)

[EXAMPLE] : Predicting Bicycle Traffic

(Ref : "05.06-Linear-Regression" from the Python Data Science Handbook by Jake VanderPlas)

— We can ***predict*** *the number of bicycle trips across Seattle's Fremont Bridge based on weather, season, and other factors*.

— Join the bike data with another dataset, and try to determine the extent to which weather and seasonal factors—temperature, precipitation, and daylight hours—affect the volume of bicycle traffic through this corridor.

Scikit-Learn Workshop 2 : 監督式學習 — 分類模型

A. Nearest Neighbor Classifier

(http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)

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

CONCEPT

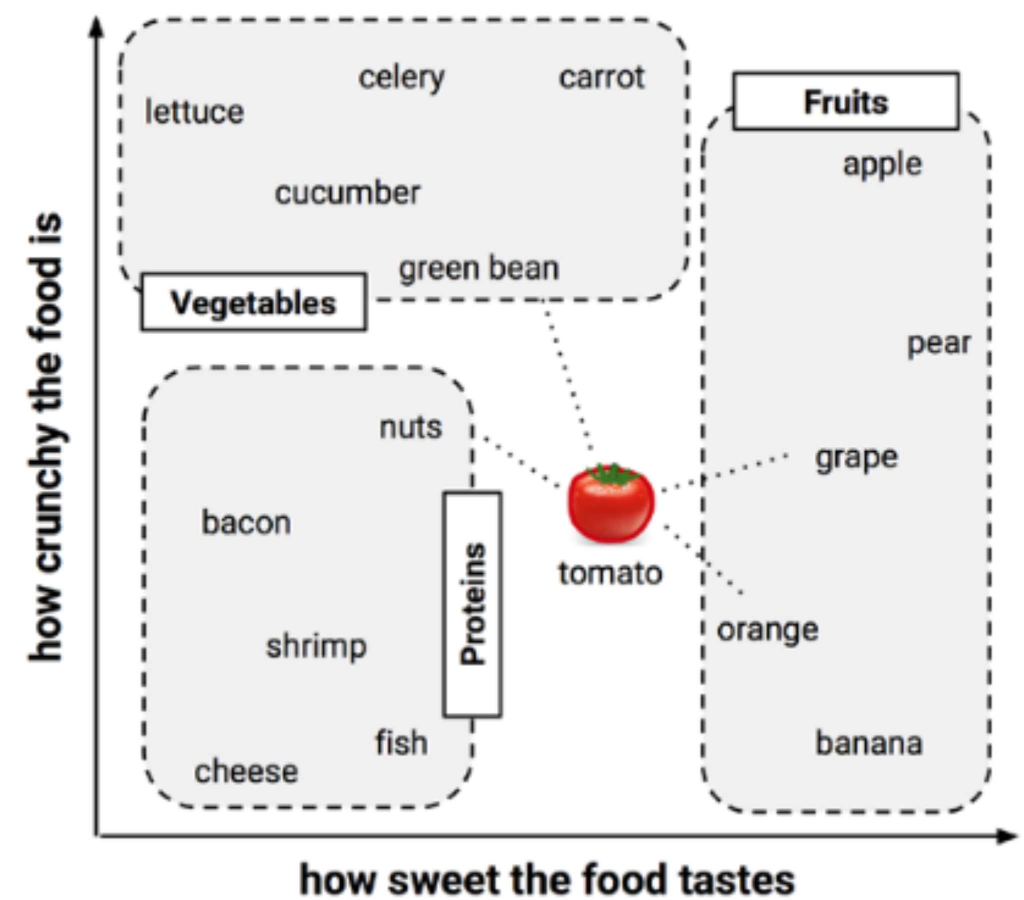
(Ref : "05.03-Hyperparameters-and-Model-Validation" from the Python Data Science Handbook by Jake VanderPlas)

— *iris* dataset

— **Model validation the right way**

- **Cross-validation**

(Refer to *Chapter 7 — "Machine Learning Training & Testing Pipeline"*)



Brett Lantz, "Machine Learning with R," Ch. 3, 2nd ed.

(https://github.com/devharsh/Technical-eBooks/blob/master/Machine%20Learning%20with%20R%2C%202nd%20Edition.pdf**)**

Scikit-Learn Workshop 2 : 監督式學習 — 分類模型 (cont'd)

[EXAMPLE 1] : Predicting Breast Cancer Diagnosis

(ML_Case_Study-Sklearn-with_Breast_Cancer_Wisconsin_Dataset-20180510.ipynb)

— Breast Cancer Wisconsin (Original) Dataset - UCI : ***wisc_bc_data.csv***

— Sklearn **Nearest Neighbor Classifier**

[EXERCISE 1] : 如何改進其預測準確率 (accuracy score) 呢 ?

[HINT] :

1. Re-evaluate the model by changing the parameter n_neighbors in the `KNeighborsClassifier()` model. (e.g., *n_neighbors=3, 5, 21, …*)
2. Re-evaluate the model by changing the parameter settings in the train_test_split(), such as random_state, train_size, test_size, etc. (e.g., *random_state=1, train_size=0.8, test_size=0.2*)

Scikit-Learn Workshop 2 : 監督式學習 — 分類模型 (cont'd)

[EXAMPLE 2] : Predicting Breast Cancer Diagnosis - PART 2

(ML_Case_Study-Sklearn-with_Breast_Cancer_Wisconsin_Dataset-20180510.ipynb)

— Breast Cancer Wisconsin (Original) Dataset - UCI : ***wisc_bc_data.csv***

— **Cross Validation** with Sklearn Nearest Neighbor Classifier

Scikit-Learn Workshop 2 : 監督式學習 – 分類模型 (cont'd)

- **Data Transformation - Normalization/Standardization**

  一般而言，當資料中的各特徵變數(feature variable)數據範圍差異過大時，通常會先行將所有特徵變數重新正規化(normalization，使其範圍介於 0 與 1 之間)，或者透過計算其 z-score 來重新進行資料的標準化(standardization)。

[EXERCISE 2] : 如何改進其預測準確率呢？ PART 2

A. 請依據上列敘述，重新將 wisc_bc_data.csv 的資料，先分別

(1) 正規化(normalization，使其範圍介於 0 與 1 之間)

(2) 標準化(standardization with z-score)

之後，分別計算其預測準確率(accuracy score)結果，並比較其差異。

[HINT] :

1. 可以分別撰寫 Python 函數，執行資料正規化(normalization) 和 標準化(standardization with z-score)。
2. 是否可以使用 sklearn 的 Pipeline 以及相關函式來執行資料轉換和建立建立模型呢？

B. 同時，將上述兩項資料轉換後的 kNN 模型，分別進行 cross-validation！

Scikit-Learn Workshop 2 : 監督式學習 — 分類模型 (cont'd)

B. Naive Bayes Classifiers (http://scikit-learn.org/stable/modules/naive_bayes.html)

- **Gaussian Naive Bayes**
- **Multinomial Naive Bayes**
- **Bernoulli Naive Bayes**

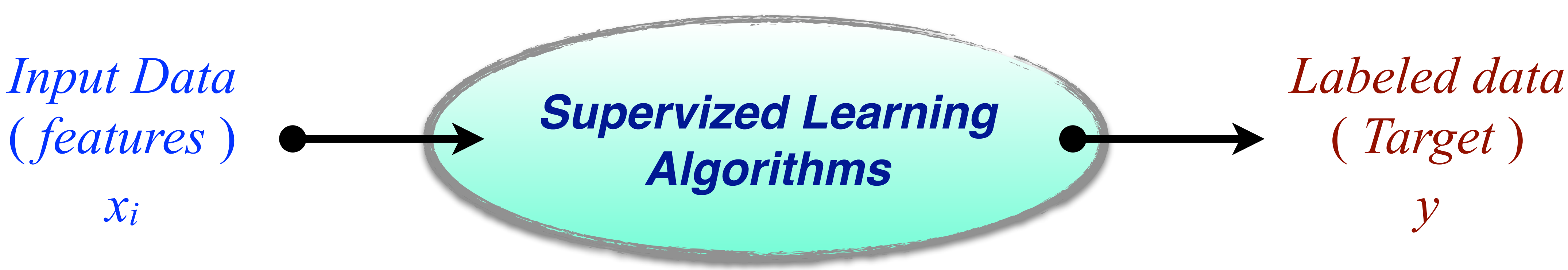
likelihood of the *features* given *L*

prior probability

posterior probability

$$\boxed{P(L \mid \text{features}) = \frac{P(\text{features} \mid L)P(L)}{P(\text{features})}}$$

evidence



Scikit-Learn Workshop 2 : 監督式學習 — 分類模型 (cont'd)

[EXAMPLE B.1] : Gaussian Naive Bayes Classifier

```
from sklearn.naive_bayes import GaussianNB
```

GaussianNB implements the Gaussian Naive Bayes algorithm for classification.

The ***likelihood of the features*** is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

likelihood of the *features* given L

prior probability

posterior probability

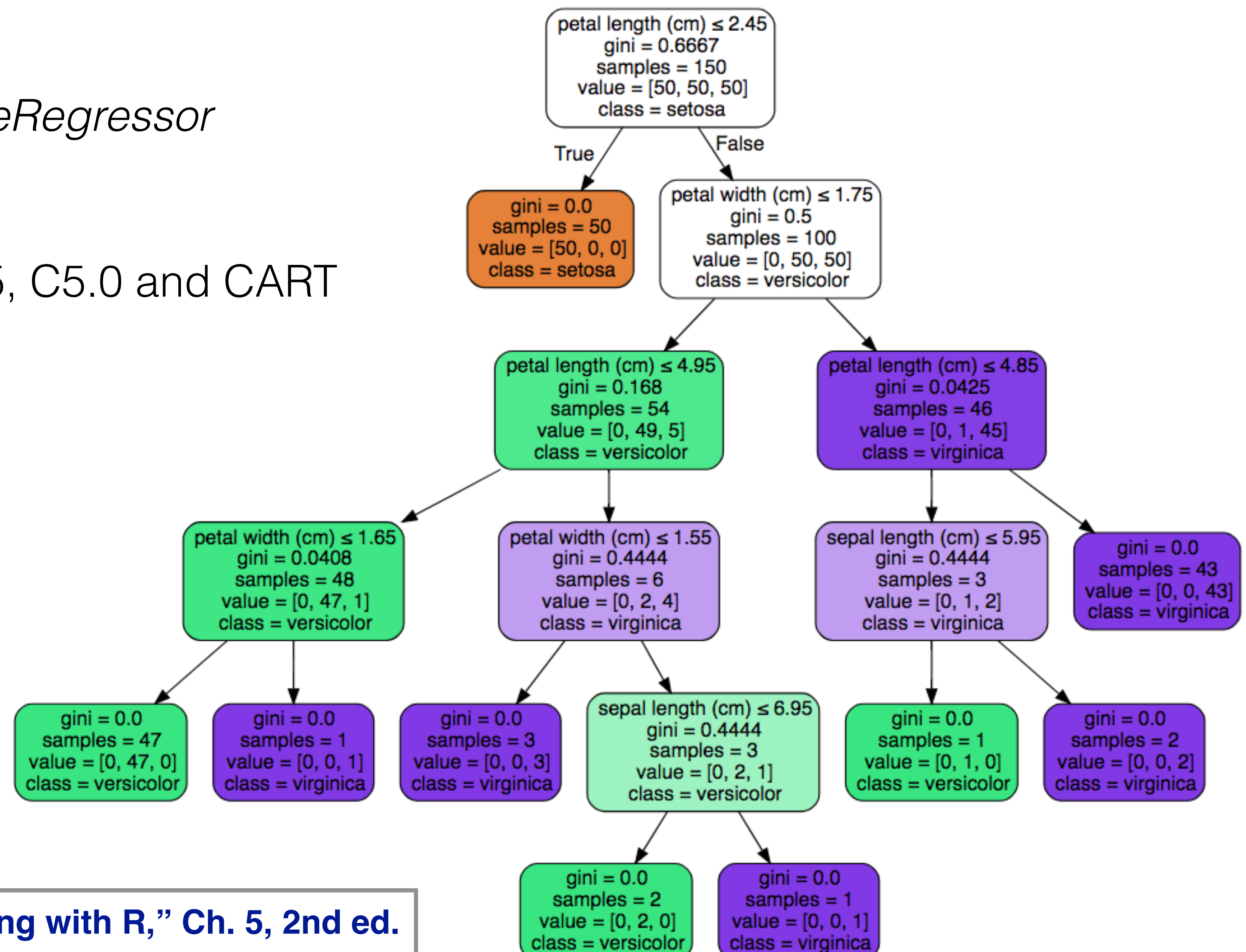
$$\boxed{P(L \mid \text{features}) = \frac{P(\text{features} \mid L)P(L)}{P(\text{features})}}$$

evidence

Scikit-Learn Workshop 2 : 監督式學習 – 分類模型 (cont'd)

C. Decision Trees Classifiers (http://scikit-learn.org/stable/modules/tree.html)

- **Classification**
- Regression - *DecisionTreeRegressor*
- Multi-output problems
- Tree algorithms: ID3, C4.5, C5.0 and CART

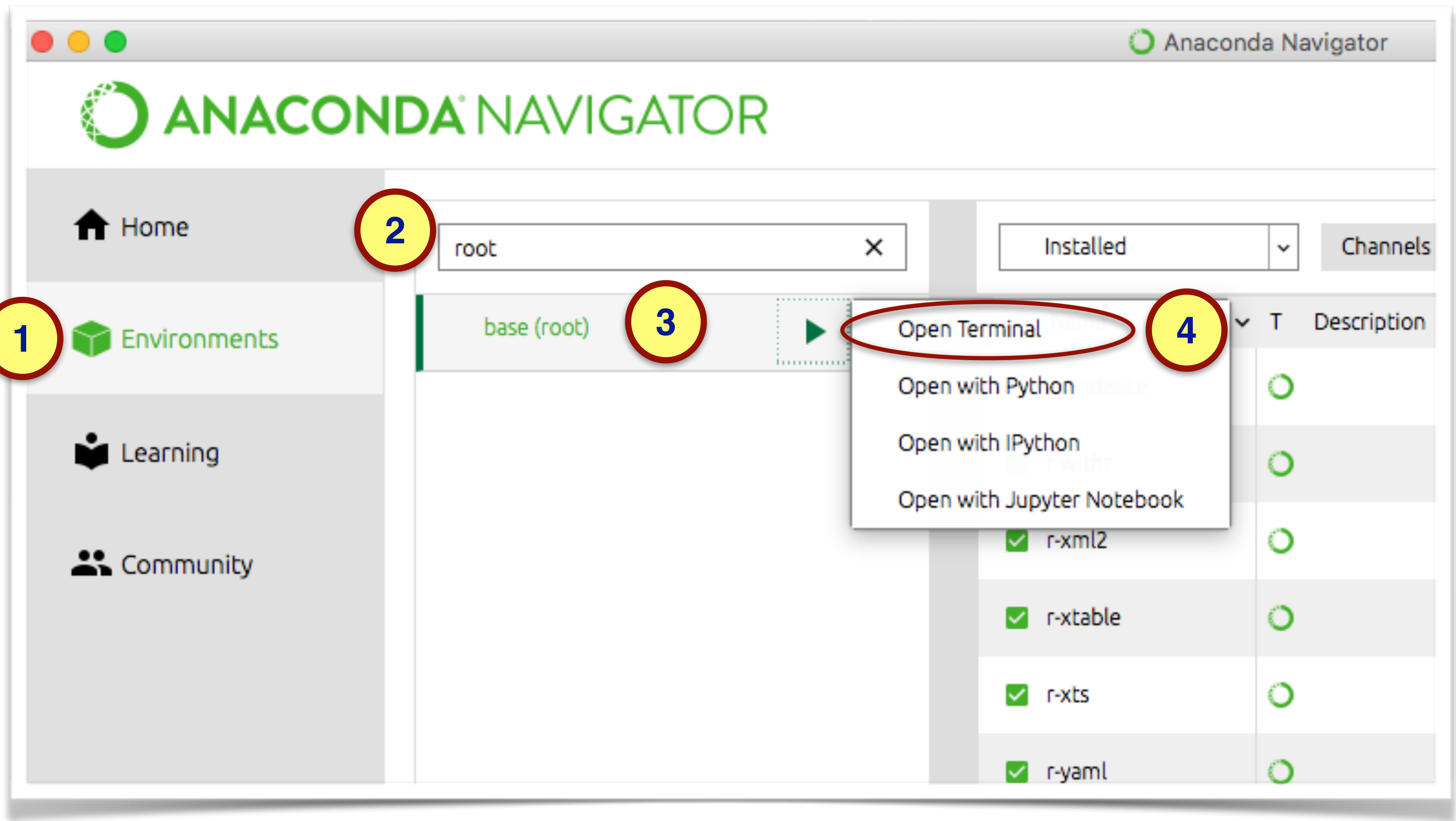


[Ref] Brett Lantz, "Machine Learning with R," Ch. 5, 2nd ed.
(https://github.com/devharsh/Technical-eBooks/blob/master/Machine%20Learning%20with%20R%2C%202nd%20Edition.pdf**)**

Scikit-Learn Workshop 2 : 監督式學習 — 分類模型 (cont'd)

Q : How to install Graphviz library on Anaconda?

STEP 1 : From the **Anaconda NAVIGATOR**



STEP 2 : On the **Terminal** prompt, key in : `conda install python-graphviz`

Scikit-Learn Workshop 3 : Validation Curves & Learning Curves

A. Validation Curves & Learning Curves — for Regression Models

Ref : "05.03-Hyperparameters-and-Model-Validation"

from the Python Data Science Handbook by Jake VanderPlas

B. Validation Curves & Learning Curves — for Classification Models

Scikit-Learn_Workshop_3-validation_curves_&_learning_curves-Classification_Models.ipynb

Scikit-Learn Workshop 4 : 非監督式學習 — 集群分析

Scikit-Learn_Workshop_4-UL-Clustering_Analysis.ipynb

[EXAMPLE] : with Breast Cancer Wisconsin (Original) Data Set : wisc_bc_data.csv

A. Unsupervised learning — Dimensionality Reduction

(Sklearn : http://scikit-learn.org/stable/modules/unsupervised_reduction.html)

- **PCA** (Principal Component Analysis)

  Ref : "05.09-Principal-Component-Analysis.ipynb"
  from the Python Data Science Handbook by Jake VanderPlas

B. Unsupervised learning — Gaussian Clustering

(Sklearn : http://scikit-learn.org/stable/modules/mixture.html)

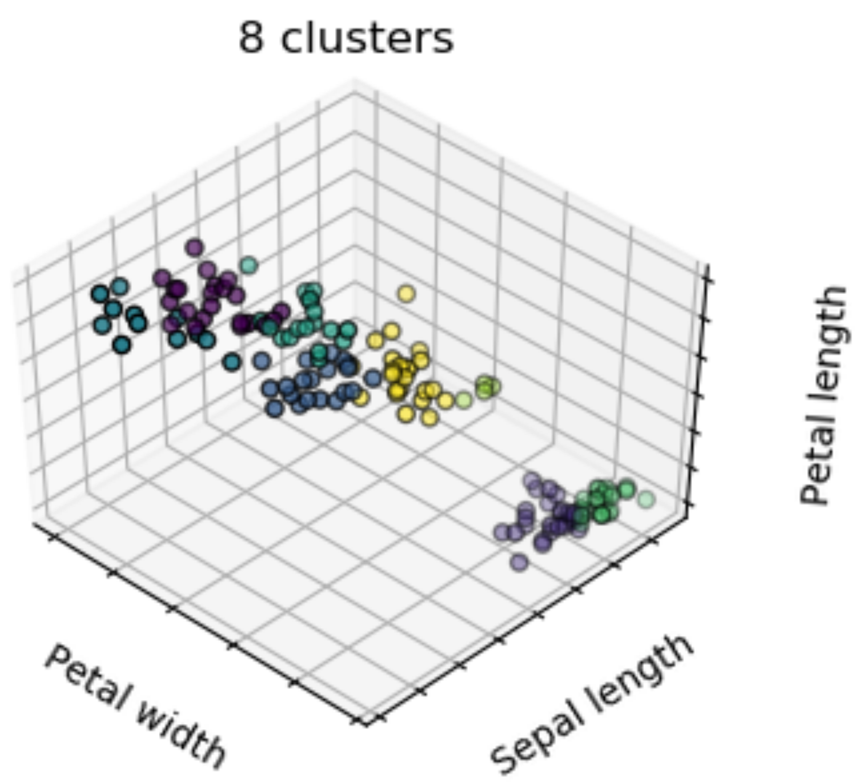
- **Gaussian mixture models**

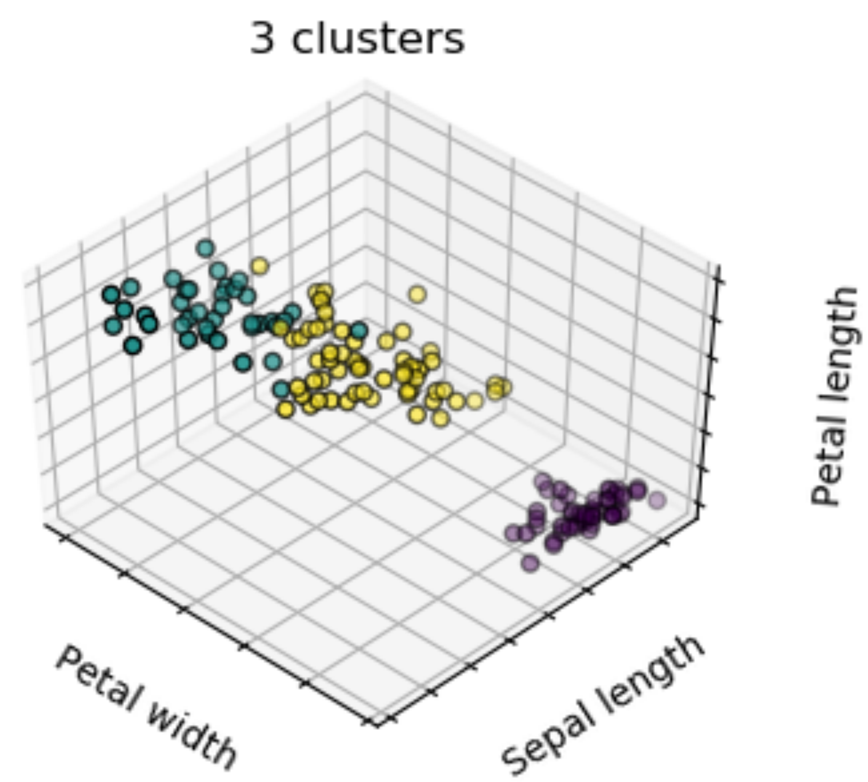
  Ref : "05.12-Gaussian mixture.ipynb"
  from the Python Data Science Handbook by Jake VanderPlas

Scikit-Learn Workshop 4 : 非監督式學習 — 集群分析 (cont'd)

C. Unsupervised learning — K-means Clustering

(Sklearn : http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html)





- **K-means models**

Ref : "05.11-K-Means.ipynb"
from the Python Data Science Handbook by Jake VanderPlas

Scikit-Learn Workshop 5 : 支持向量機 SVM (for Both)

Sklearn — Support Vector Machines

http://scikit-learn.org/stable/modules/svm.html

A. SVC & Kernel Methods — for Classification

- **SVC & Kernel Methods**

  Ref : "05.07-Support-Vector-Machines.ipynb"
  from the Python Data Science Handbook by Jake VanderPlas

B. SVR — for Regression

- The method of Support Vector Classification can be extended to solve regression problems. This method is called **Support Vector Regression**.

  Ref : "**Support Vector Regression (SVR) using linear and non-linear kernels**"
  http://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html#sphx-glr-auto-examples-svm-plot-svm-regression-py

Scikit-Learn Workshop 6 : Neural Networks (for Both)

- **Kaggle — www.kaggle.com**
- **Sklearn — Neural Networks**

  http://scikit-learn.org/stable/modules/neural_networks_supervised.html

A. Neural Networks — for Classification

- **`sklearn.neural_network.MLPClassifier`**

  Ref : http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier

B. Neural Networks — for Regression

- **`sklearn.neural_network.MLPRegressor`**

  Ref : http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html#sklearn.neural_network.MLPRegressor

Scikit-Learn Workshop 7 : Meta-Learner - Random Forests

Meta-Learners — Ensemble methods

http://scikit-learn.org/stable/modules/ensemble.html

A. Random Forests — for Classification

- **RandomForestClassifier**

Ref : "05.08-Random-Forests.ipynb"
from the Python Data Science Handbook by Jake VanderPlas

B. Random Forests — for Regression

- **RandomForestRegressor**

Ref : "05.08-Random-Forests.ipynb"
from the Python Data Science Handbook by Jake VanderPlas