Chapter 12

Scikit-Learn Workshop 2 :

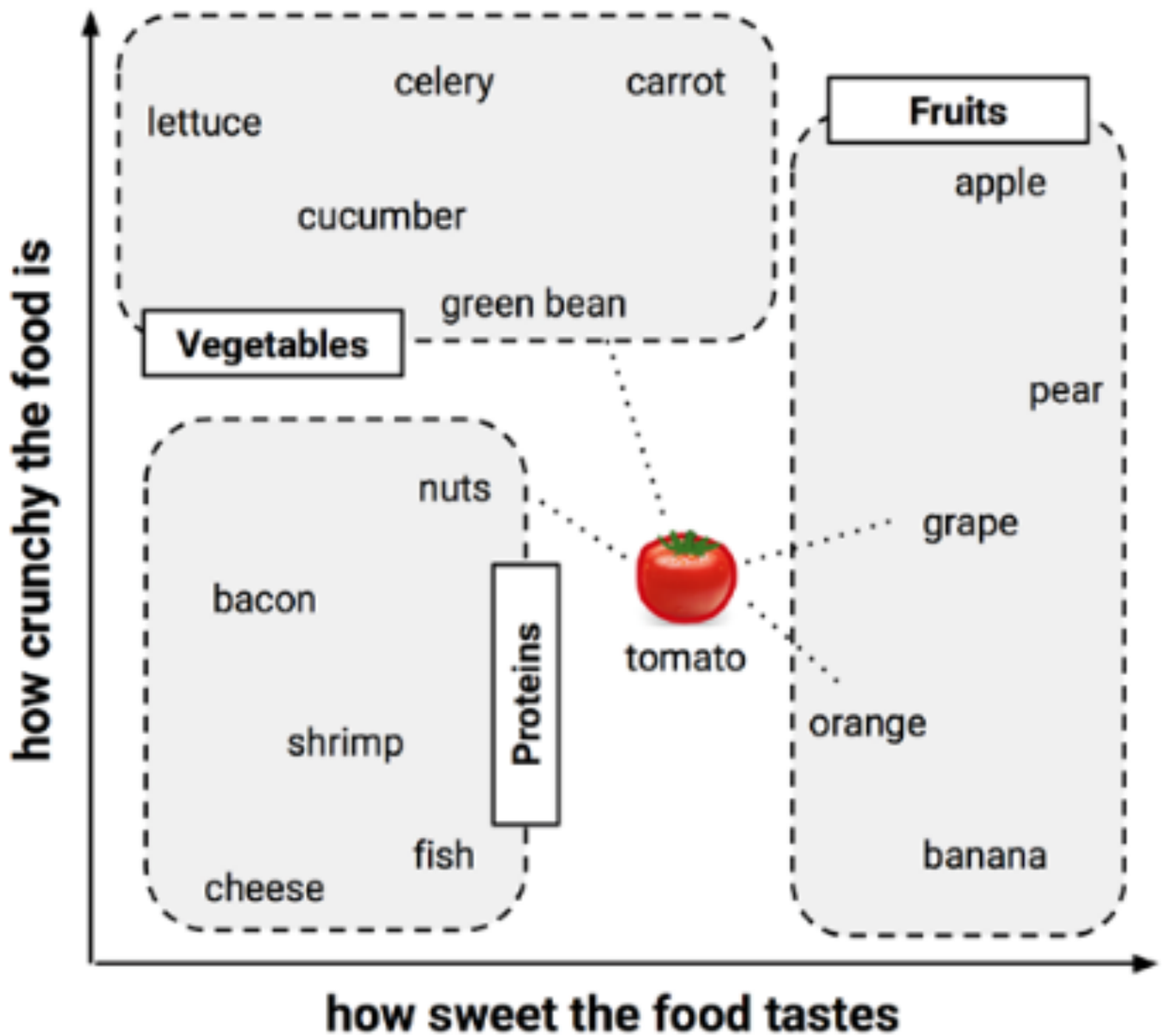
監督式學習 — 分類模型

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)

CONCEPT



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

A. Nearest Neighbor Classifier

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

from sklearn.**neighbors** import **KNeighborsClassifier**

(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 10 — “Machine Learning Training & Testing Pipeline”*)

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] :

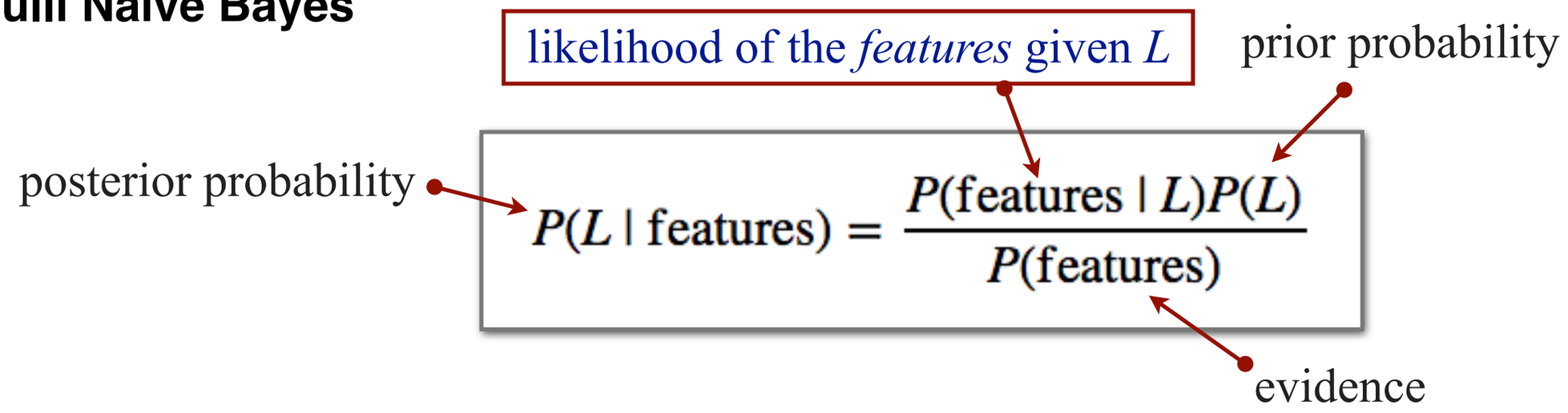
1. 可以分別撰寫 Python 函數，執行資料正規化(normalization) 和 標準化(standardization with z-score)。
2. 是否可以使用 sklearn 的 Pipeline 以及相關函式來執行資料轉換和建立建立模型呢？

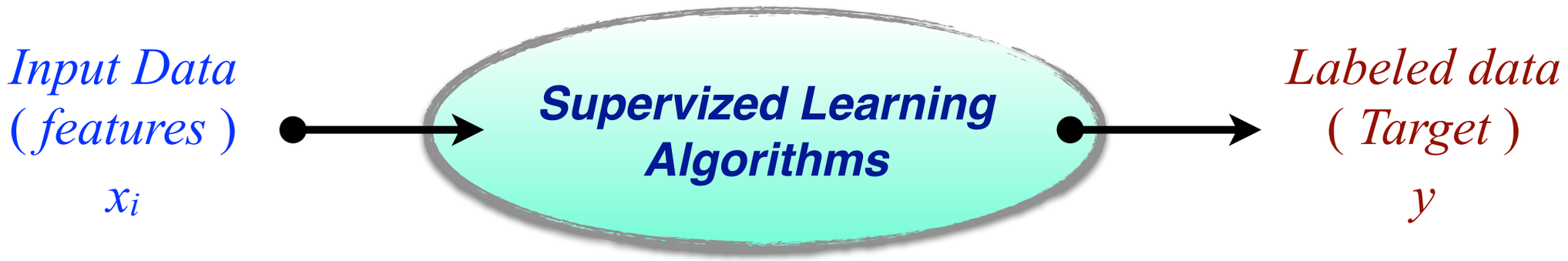
B. 同時，將上述兩項資料轉換後的 kNN 模型，分別進行 cross-validation！

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**





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

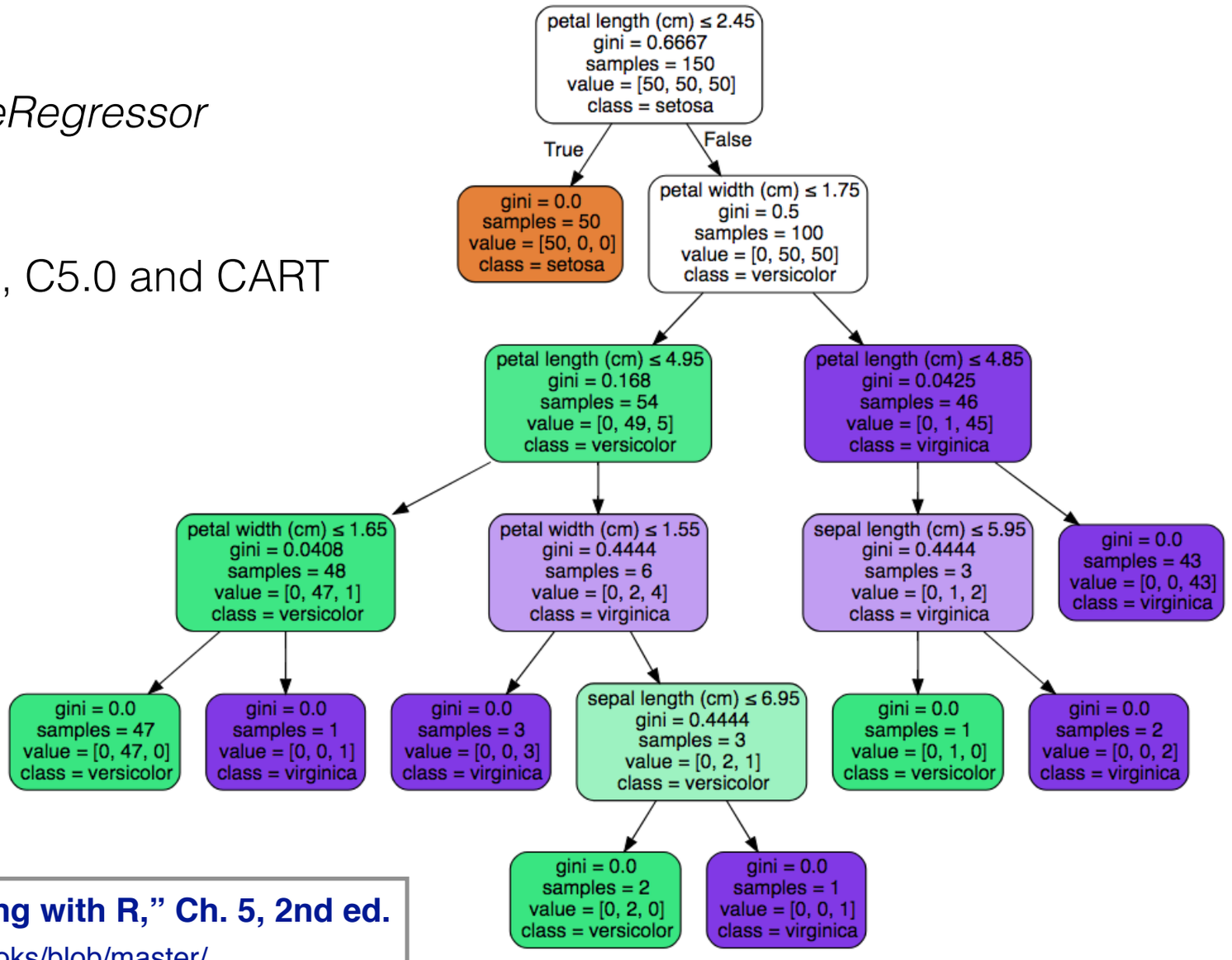
$$P(L \mid \text{features}) = \frac{P(\text{features} \mid L)P(L)}{P(\text{features})}$$

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

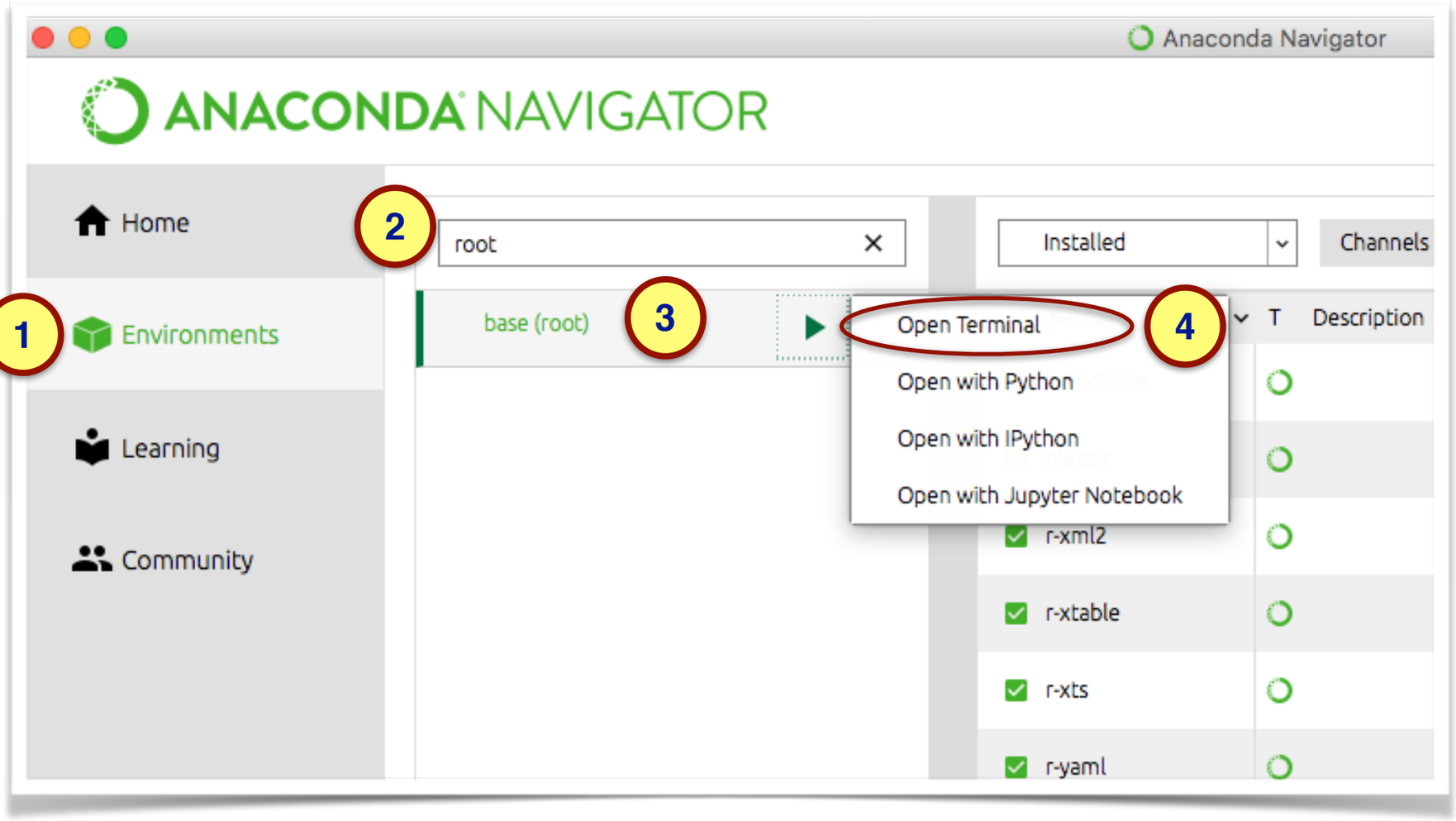


[Ref] Brett Lantz, "Machine Learning with R," Ch. 5, 2nd ed.
(https://github.com/devharsh/Technical-eBooks/blob/master/Machine%20Learning%20with%20R%2C%202nd%20Edition.pdf**)**

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`

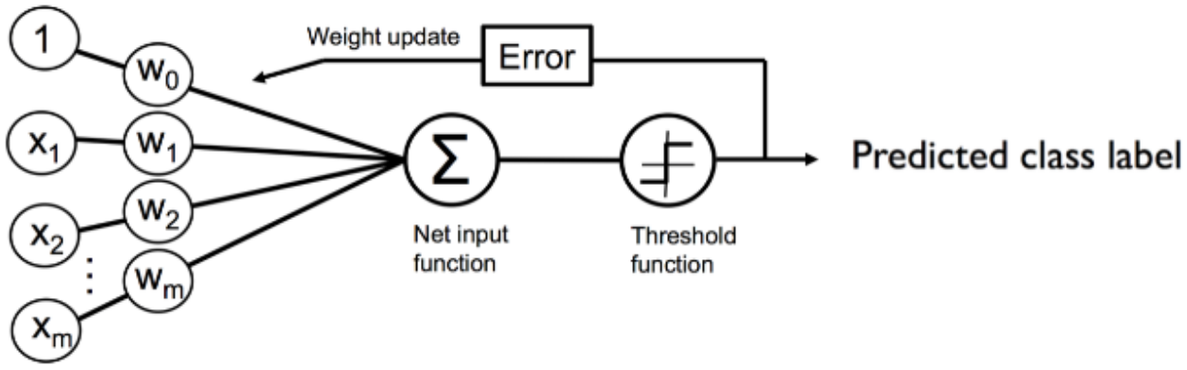
Scikit-Learn Workshop 2：監督式學習 — 分類模型 (cont'd)

D. Logistic Regression

(http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

Perceptron *vs.* Logistic Regression

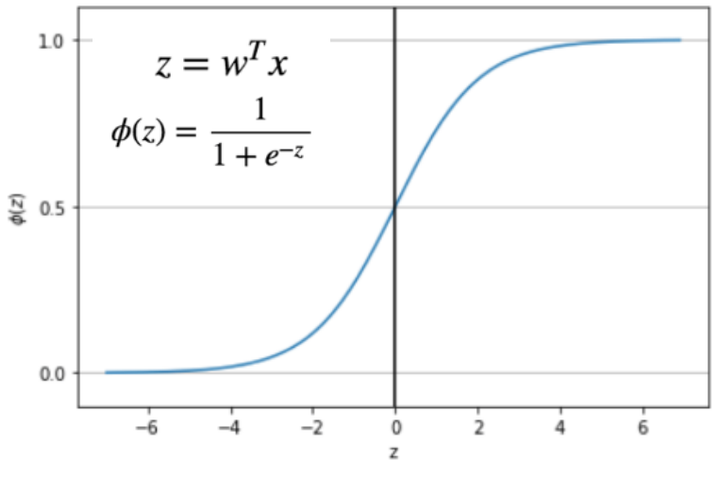
— Activation functions : Step function *vs. Sigmoid function* (Logistic function)

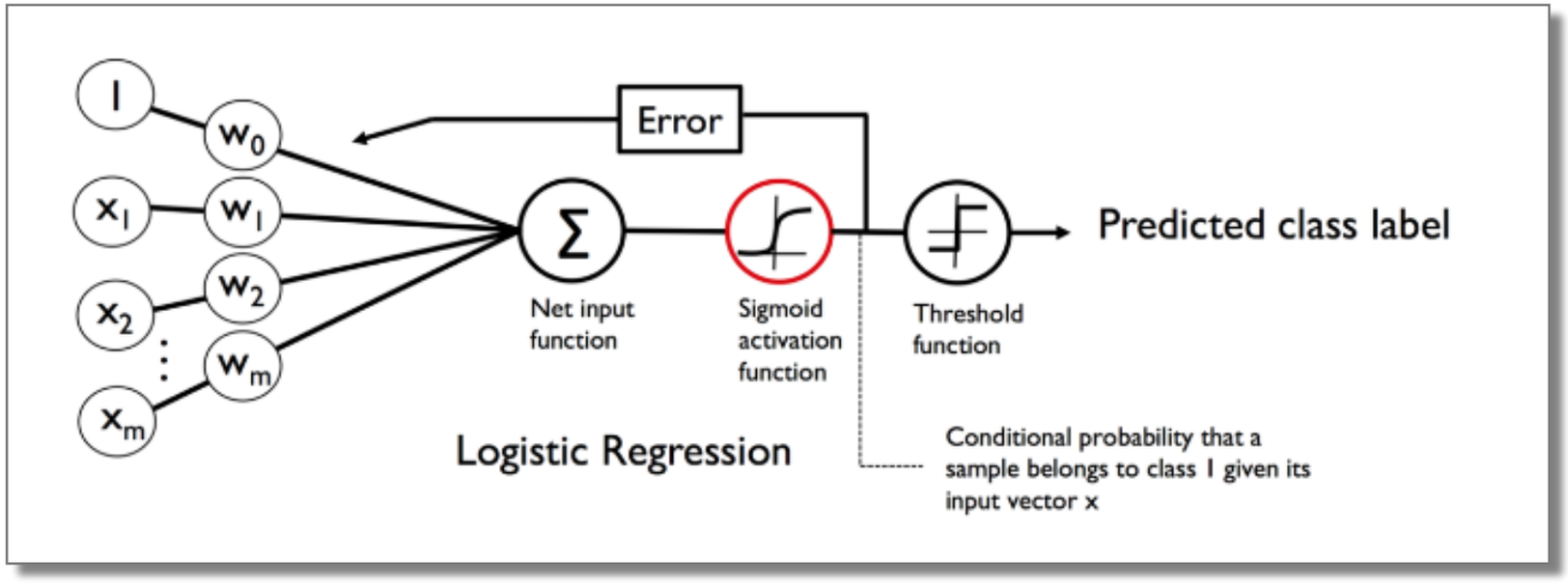


Perceptron

***hyperplane* (a linear divisor)**

$$\hat{y}_i = \sum_{j=0}^{M} w_j * x_{ij}$$





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

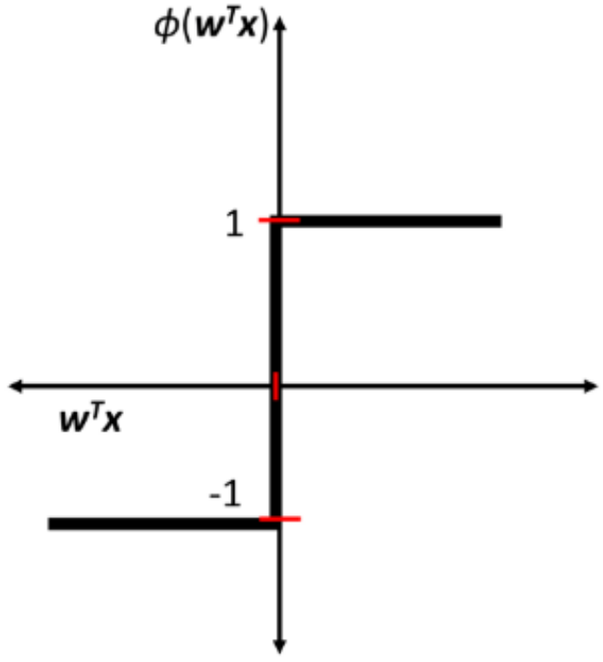
D. Logistic Regression

(http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

Perceptron *vs.* Logistic Regression

— **Activation functions** : Step function *vs. Sigmoid function* (Logistic function)

Perceptron



Logistic Regression