

Classification 分類

Classification 分類問題

Supervised vs. Unsupervised Learning

- Supervised learning (**classification**分類)
 - Supervision[監督]: The training data (observations, measurements, etc.) are accompanied by **labels** [標籤|答案] indicating the class of the observations
 - New data is classified based on the training set
- Unsupervised learning (**clustering**叢集)
 - The class labels of training data is unknown
 - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Prediction Problems: Classification vs. Numeric Prediction

- Classification

- predicts categorical class labels (discrete or nominal)
- classifies data (constructs a model) based on the training set and the values (**class labels**) in a classifying attribute and uses it in classifying new data

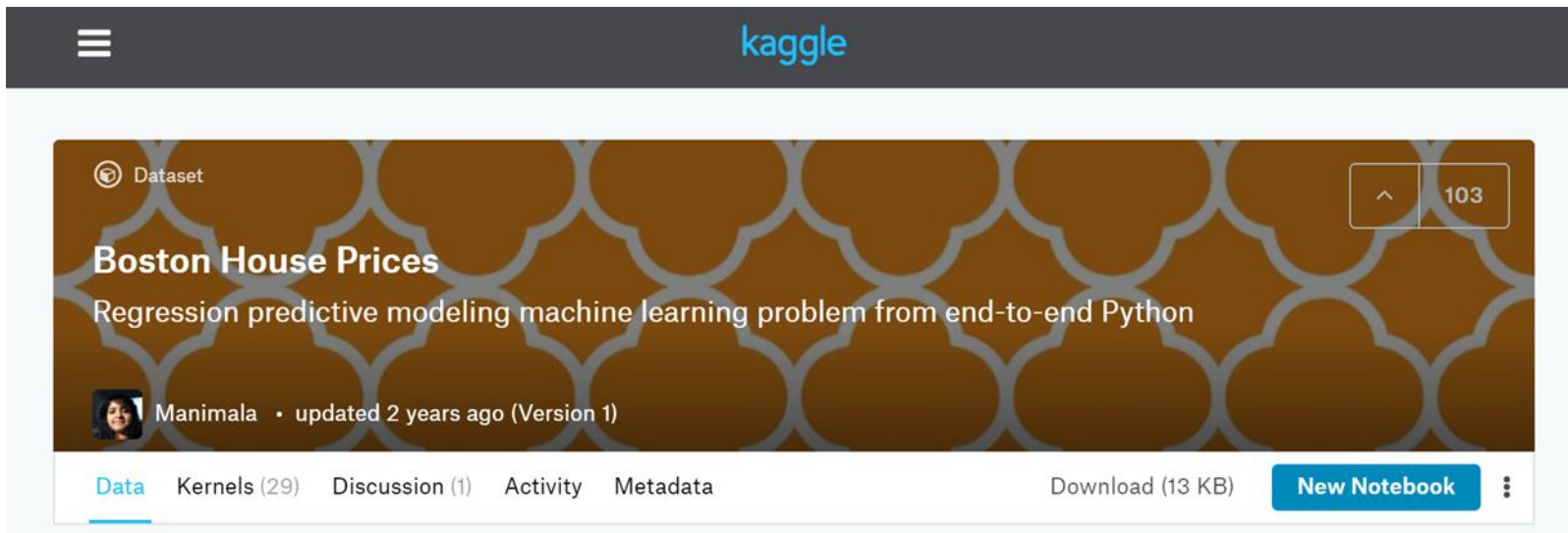
- Numeric Prediction

- models continuous-valued functions, i.e., predicts unknown or missing values

Numeric Prediction

房屋售價預測

Boston House Prices



The screenshot shows the Kaggle interface for the 'Boston House Prices' dataset. The header includes the Kaggle logo and a menu icon. The dataset title 'Boston House Prices' is prominently displayed, followed by a description: 'Regression predictive modeling machine learning problem from end-to-end Python'. Below this, the creator's name 'Manimala' and the update date 'updated 2 years ago (Version 1)' are shown. A navigation bar at the bottom contains links for 'Data', 'Kernels (29)', 'Discussion (1)', 'Activity', and 'Metadata'. On the right side of the navigation bar, there is a 'Download (13 KB)' link and a 'New Notebook' button. A small box in the top right corner of the dataset card shows an upward arrow and the number '103'.

Dataset

Boston House Prices

Regression predictive modeling machine learning problem from end-to-end Python

Manimala • updated 2 years ago (Version 1)

Data Kernels (29) Discussion (1) Activity Metadata

Download (13 KB) New Notebook

<https://www.kaggle.com/vikrishnan/boston-house-prices>

Prediction Problems: Classification vs. Numeric Prediction

- Classification

- predicts categorical class labels (discrete or nominal)
- classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Typical applications: Binary classification vs Multi-class classification
 - Credit/loan approval:信用評比與貸款通過
 - Medical diagnosis: if a tumor is cancerous or benign
是否有XX癌症
 - Fraud detection: if a transaction is fraudulent 詐騙或正常交易
 - 垃圾郵件 (SPAM) 或正常郵件
 - Web page categorization網站分類: which category it is

sklearn.datasets: Datasets

Loaders

<code>datasets.clear_data_home ([data_home])</code>	Delete all the content of the data home cache.
<code>datasets.dump_svmlight_file (X, y, f[, ...])</code>	Dump the dataset in svmlight / libsvm file format.
<code>datasets.fetch_20newsgroups ([data_home, ...])</code>	Load the filenames and data from the 20 newsgroups dataset (classification).
<code>datasets.fetch_20newsgroups_vectorized ([...])</code>	Load the 20 newsgroups dataset and vectorize it into token counts (classification).
<code>datasets.fetch_california_housing ([...])</code>	Load the California housing dataset (regression).
<code>datasets.fetch_covtype ([data_home, ...])</code>	Load the covtype dataset (classification).
<code>datasets.fetch_kddcup99 ([subset, data_home, ...])</code>	Load the kddcup99 dataset (classification).
<code>datasets.fetch_lfw_pairs ([subset, ...])</code>	Load the Labeled Faces in the Wild (LFW) pairs dataset (classification).
<code>datasets.fetch_lfw_people ([data_home, ...])</code>	Load the Labeled Faces in the Wild (LFW) people dataset (classification).
<code>datasets.fetch_olivetti_faces ([data_home, ...])</code>	Load the Olivetti faces data-set from AT&T (classification).
<code>datasets.fetch_openml ([name, version, ...])</code>	Fetch dataset from openml by name or dataset id.
<code>datasets.fetch_rcv1 ([data_home, subset, ...])</code>	Load the RCV1 multilabel dataset (classification).
<code>datasets.fetch_species_distributions ([...])</code>	Loader for species distribution dataset from Phillips et.
<code>datasets.get_data_home ([data_home])</code>	Return the path of the scikit-learn data dir.
<code>datasets.load_boston ([return_X_y])</code>	Load and return the boston house-prices dataset (regression).
<code>datasets.load_breast_cancer ([return_X_y])</code>	Load and return the breast cancer wisconsin dataset (classification).
<code>datasets.load_diabetes ([return_X_y])</code>	Load and return the diabetes dataset (regression).
<code>datasets.load_digits ([n_class, return_X_y])</code>	Load and return the digits dataset (classification).
<code>datasets.load_files (container_path[, ...])</code>	Load text files with categories as subfolder names.
<code>datasets.load_iris ([return_X_y])</code>	Load and return the iris dataset (classification).
<code>datasets.load_linnerud ([return_X_y])</code>	Load and return the linnerud dataset (multivariate regression).
<code>datasets.load_sample_image (image_name)</code>	Load the numpy array of a single sample image
<code>datasets.load_sample_images ()</code>	Load sample images for image manipulation.
<code>datasets.load_svmlight_file (f[, n_features, ...])</code>	Load datasets in the svmlight / libsvm format into sparse CSR matrix
<code>datasets.load_svmlight_files (files[, ...])</code>	Load dataset from multiple files in SVMlight format
<code>datasets.load_wine ([return_X_y])</code>	Load and return the wine dataset (classification).

分類演算法測試常用資料集

糖尿病預測:有沒有徵兆

Binary-class classification

Pima印第安人數據集

根據診斷措施預測糖尿病的發病

資料集簡介

- 該資料集最初來自國家糖尿病/消化/腎臟疾病研究所。
- 資料集的目標是基於資料集中包含的**某些診斷測量**來**診斷性的預測** **患者是否患有糖尿病**。
- 從較大的資料庫中選擇這些實例有幾個約束條件。尤其是，這裡的所有患者都是Pima印第安至少**21**歲的女性。
- 資料集由多個醫學預測變數和一個目標變數組成**Outcome**。預測變數包括患者的懷孕次數、BMI、胰島素水準、年齡等。

- 【1】 Pregnancies：懷孕次數
- 【2】 Glucose：葡萄糖
- 【3】 BloodPressure：血壓 (mm Hg)
- 【4】 SkinThickness：皮層厚度 (mm)
- 【5】 Insulin：胰島素 2小時血清胰島素 (mu U / ml)
- 【6】 BMI：體重指數 (體重/身高)^2
- 【7】 DiabetesPedigreeFunction：糖尿病譜系功能
- 【8】 Age：年齡 (歲)

【9】 Outcome：類標變數 (0或1)

```
X = pima.iloc[:, 0:8] # 特徵列 0-7列，不含第8列  
Y = pima.iloc[:, 8] # 目標列為第8列
```

```
!wget
```

```
https://raw.githubusercontent.com/MyDearGreat  
Teacher/AI201909/master/data/pima_data.csv
```

```
import pandas as pd
```

```
pima = pd.read_csv("pima_data.csv")
```

```
pima.head()
```

```
pima.describe()
```

```
# panda的describe描述屬性，展示了每一個欄位的
```

```
# 【count條目統計，mean平均值，std標準值，min最小值，25%，  
50%中位數，75%，max最大值】
```

```
pima.shape
```

Data Visualization - 數據視覺化

```
pima.hist(figsize=(16,14))
```

#查看每個欄位的資料分佈；

figsize的參數顯示的是每個子圖的長和寬

<https://blog.csdn.net/yizheyoye/article/details/79791473>

```
sklearn.datasets.load_diabetes
```

分類演算法測試常用資料集

威斯康辛乳腺癌資料集

:['malignant惡性', 'benign良性']

Binary-class classification

乳腺癌是世界範圍內婦女死亡的主要原因之一，
準確的診斷是乳腺癌治療中最重要的一步驟之一。

乳腺癌威斯康辛(診斷)資料集

該資料集分類標籤劃分為兩類(惡性、良性)

```
data = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-  
databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data",  
names=column_names)  
print(data)
```

1. Sample code number id number
2. Clump Thickness 1 – 10
3. Uniformity of Cell Size 1 – 10
4. Uniformity of Cell Shape 1 – 10
5. Marginal Adhesion 1 – 10
6. Single Epithelial Cell Size 1 – 10
7. Bare Nuclei 1 – 10
8. Bland Chromatin 1 – 10
9. Normal Nucleoli 1 – 10
10. Mitoses 1 – 10
11. Class: (2 for benign, 4 for malignant)

```
breast_cancer_data = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-  
databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data', header=None  
    , names = ['C_D', 'C_T', 'U_C_Si', 'U_C_Sh', 'M_A', 'S_E_C_S'  
    , 'B_N', 'B_C', 'N_N', 'M', 'Class'])
```

```
breast_cancer_data
```

```
print (breast_cancer_data.shape)
```

```
breast_cancer_data.info()
```

```
breast_cancer_data.head(25)
```

```
print(breast_cancer_data.describe())
```

<https://blog.csdn.net/ruoyunliufeng/article/details/79369142>


```
sklearn.datasets. load_breast_cancer (return_X_y=False)
```

[\[source\]](#)

Load and return the breast cancer wisconsin dataset (classification).

The breast cancer dataset is a classic and very easy binary classification dataset.

Classes	2
Samples per class	212(M),357(B)
Samples total	569
Dimensionality	30
Features	real, positive

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html

Read more in the [User Guide](#).

Parameters: `return_X_y` : *boolean, default=False*

If True, returns `(data, target)` instead of a Bunch object. See below for more information about the `data` and `target` object.

New in version 0.18.

Returns: `data` : *Bunch*

Dictionary-like object, the interesting attributes are: 'data', the data to learn, 'target', the classification labels, 'target_names', the meaning of the labels, 'feature_names', the meaning of the features, and 'DESCR', the full description of the dataset, 'filename', the physical location of breast cancer csv dataset (added in version 0.20).

(data, target) : *tuple if return_X_y is True*

New in version 0.18.

The copy of UCI ML Breast Cancer Wisconsin (Diagnostic) dataset is
downloaded from:

<https://goo.gl/U2Uwz2>

關鍵原始碼分析

<https://github.com/scikit-learn/scikit-learn/blob/1495f6924/sklearn/datasets/base.py#L402>

```
feature_names = np.array(['mean radius', 'mean texture',  
                          'mean perimeter', 'mean area',  
                          'mean smoothness', 'mean compactness',  
                          'mean concavity', 'mean concave points',  
                          'mean symmetry', 'mean fractal dimension',  
                          'radius error', 'texture error',  
                          'perimeter error', 'area error',  
                          'smoothness error', 'compactness error',  
                          'concavity error', 'concave points error',  
                          'symmetry error', 'fractal dimension error',  
                          'worst radius', 'worst texture',  
                          'worst perimeter', 'worst area',  
                          'worst smoothness', 'worst compactness',  
                          'worst concavity', 'worst concave points',  
                          'worst symmetry', 'worst fractal dimension'])
```

關鍵原始碼分析

<https://github.com/scikit-learn/scikit-learn/blob/1495f6924/sklearn/datasets/base.py#L402>

```
return Bunch(data=data, target=target,  
            target_names=target_names,  
            DESCR=fdescr,  
            feature_names=feature_names,  
            filename=csv_filename)
```

Let's say you are interested in the samples 10, 50, and 85, and want to know their class name.

```
from sklearn.datasets import load_breast_cancer  
data = load_breast_cancer()  
data.target[[10, 50, 85]]  
list(data.target_names)
```

['malignant惡性', 'benign良性']

分類演算法測試常用資料集

鳶尾花卉資料集Iris flower data set

Multi-class classification

<https://zh.wikipedia.org/wiki/安德森鳶尾花卉數據集>

安德森鳶尾花卉數據集（英文：Anderson's Iris data set），也稱鳶尾花卉數據集（英文：Iris flower data set）或

費雪鳶尾花卉數據集（英文：Fisher's Iris data set），是一類多重變量分析的數據集。

它最初是埃德加·安德森從加拿大加斯帕半島上的鳶尾屬花朵中提取的形態學變異數據，後由羅納德·費雪作為判別分析的一個例子，運用到統計學中。

其數據集包含了**150**個樣本，都屬於鳶尾屬下的三個亞屬，分別是山鳶尾、變色鳶尾和維吉尼亞鳶尾。

四個特徵被用作樣本的定量分析，它們分別是花萼和花瓣的長度和寬度。

基於這四個特徵的集合，費雪發展了一個**線性判別分析(linear discriminant analysis)**以確定其屬種。

Iris setosa

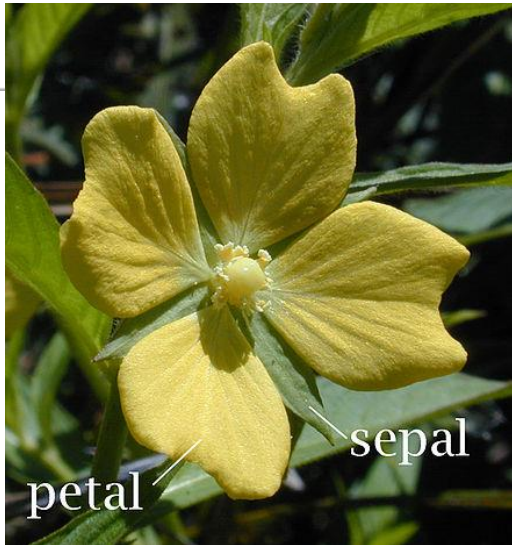
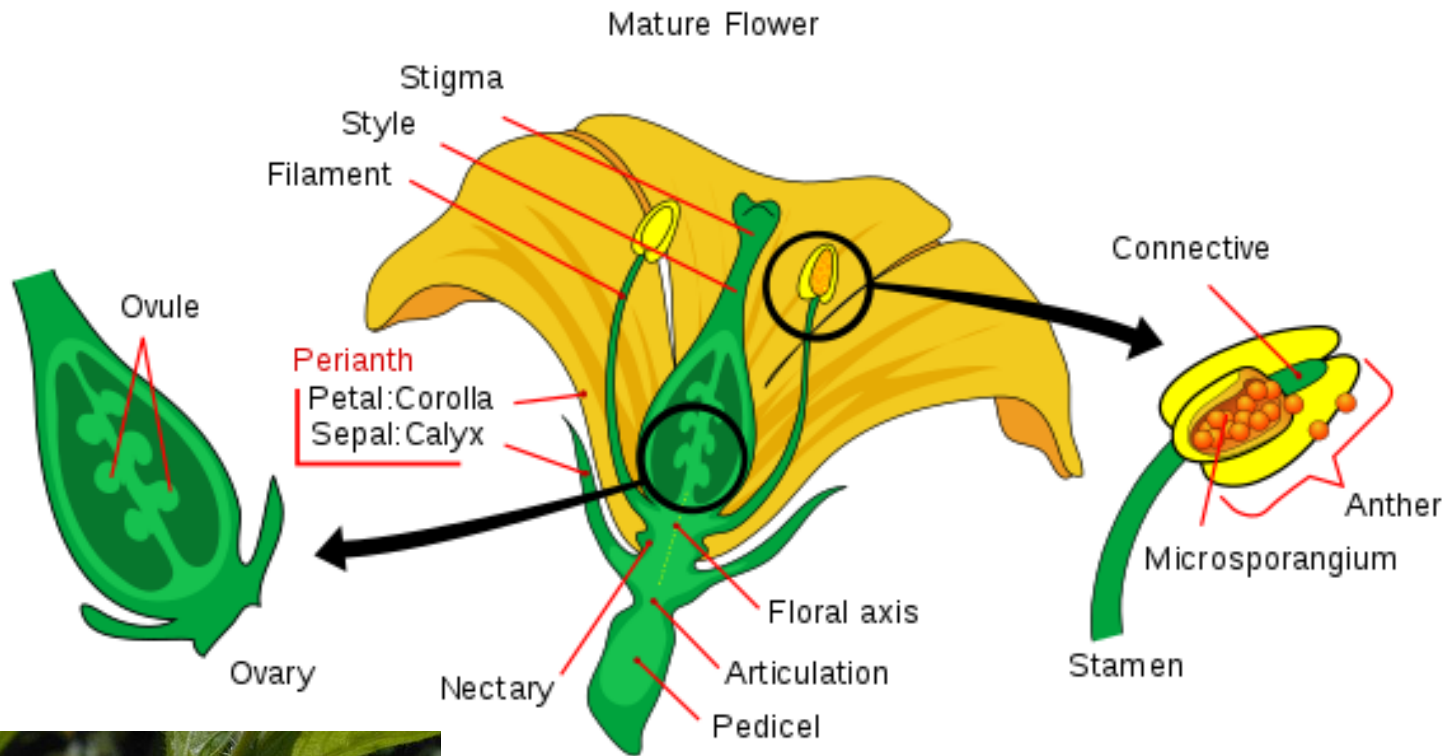


Iris versicolor



Iris virginica

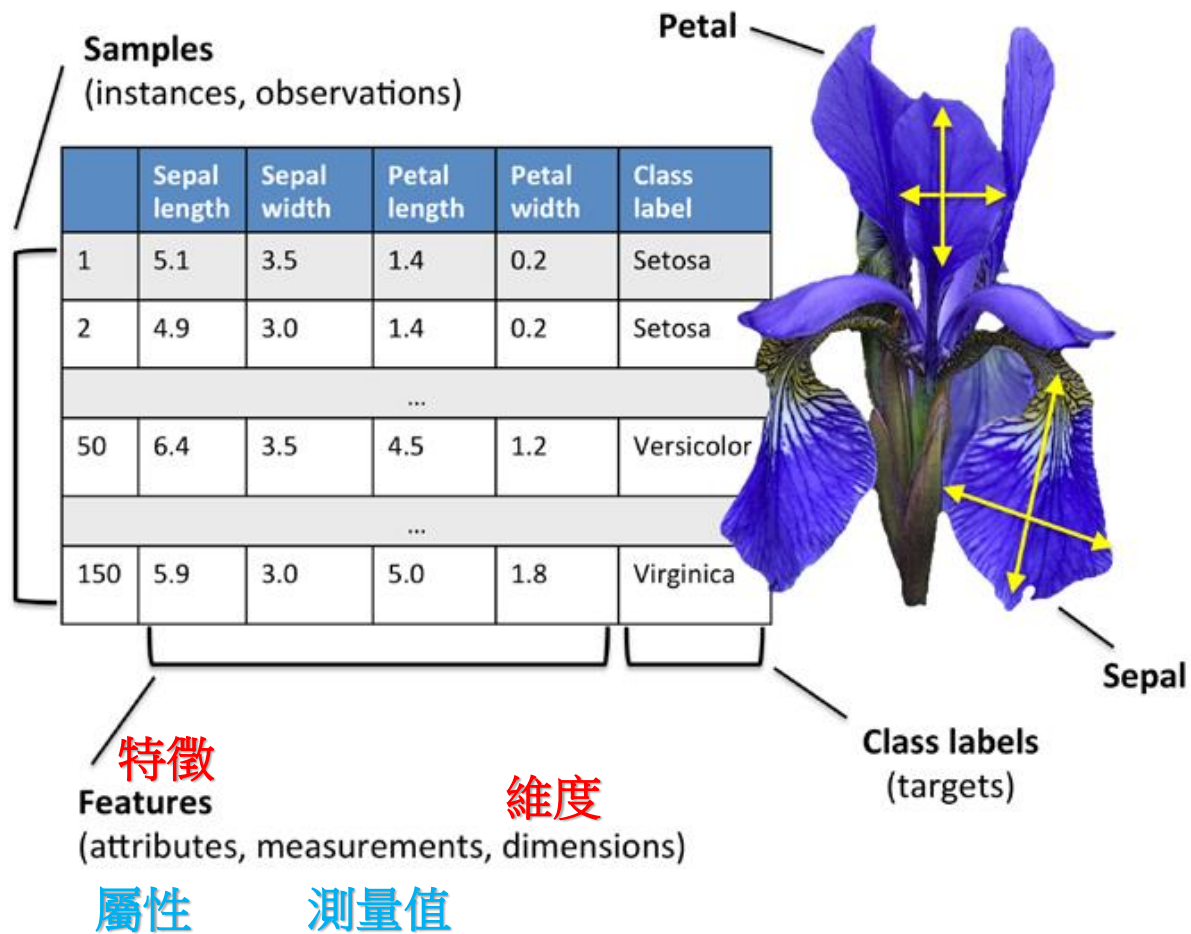




花萼是一朵花中所有萼片的總稱，位於花的最外層，一般是綠色

花瓣

花萼



使用matrix(矩陣)來表達資料

	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

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(150)} \end{bmatrix}$$

($y \in \{\text{Setosa, Versicolor, Virginica}\}$)

$$\mathbf{x}^{(i)} = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} & x_3^{(i)} & x_4^{(i)} \end{bmatrix}$$

$$\mathbf{x}_j = \begin{bmatrix} x_j^{(1)} \\ x_j^{(2)} \\ \vdots \\ x_j^{(150)} \end{bmatrix}$$

$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{bmatrix}$$

<https://scikit-learn.org/stable/datasets/index.html#iris-dataset>

5.2.2. Iris plants dataset

Data Set Characteristics:

Number of Instances:

150 (50 in each of three classes)

Number of Attributes:

4 numeric, predictive attributes and the class

Attribute Information:

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class:
 - Iris-Setosa
 - Iris-Versicolour
 - Iris-Virginica

Summary Statistics:

sepal length:	4.3	7.9	5.84	0.83	0.7826
sepal width:	2.0	4.4	3.05	0.43	-0.4194
petal length:	1.0	6.9	3.76	1.76	0.9490 (high)
petal width:	0.1	2.5	1.20	0.76	0.9565 (high)

Missing Attribute Values:

None

Class Distribution:

33.3% for each of 3 classes.

Creator: R.A. Fisher

Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)

Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken from Fisher's paper. Note that it's the same as in R, but not as in the UCI Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

References

- Fisher, R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to Mathematical Statistics" (John Wiley, NY, 1950).
- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis. (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.
- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System Structure and Classification Rule for Recognition in Partially Exposed Environments". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 1, 67-71.
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions on Information Theory, May 1972.

sklearn.datasets.load_iris()

sklearn.datasets.load_iris

```
sklearn.datasets.load_iris(return_X_y=False)
```

[\[source\]](#)

Load and return the iris dataset (classification).

The iris dataset is a classic and very easy multi-class classification dataset.

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

Read more in the [User Guide](#).

Parameters: `return_X_y` : *boolean, default=False*.

If True, returns `(data, target)` instead of a Bunch object. See below for more information about the data and target object.

New in version 0.18

Returns: `data` : *Bunch*

Dictionary-like object, the interesting attributes are: 'data', the data to learn, 'target', the classification labels, 'target_names', the meaning of the labels, 'feature_names', the meaning of the features, 'DESCR', the full description of the dataset, 'filename', the physical location of iris csv dataset (added in version 0.20).

`(data, target)` : *tuple if return_X_y is True*

New in version 0.18

sklearn.datasets.load_iris()

sklearn.datasets.load_iris

```
sklearn.datasets.load_iris(return_X_y=False)
```

[\[source\]](#)

Load and return the iris dataset (classification).

The iris dataset is a classic and very easy multi-class classification dataset.

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

Read more in the [User Guide](#).

Parameters: **return_X_y** : bool

If True, return
about the data

New in version

Returns: **data** : *Bunch*

Dictionary-like
classification labels, 'target_names', the meaning of the labels, 'feature_names', the meaning of
the features, 'DESCR', the full description of the dataset, 'filename', the physical location of iris
csv dataset (added in version 0.20).

(data, target) : tuple if *return_X_y* is True

New in version 0.18

L328-L400

```
return Bunch(data=data, target=target,  
             target_names=target_names,  
             DESCR=fdescr,  
             feature_names=['sepal length (cm)', 'sepal width (cm)',  
                           'petal length (cm)', 'petal width (cm)'],  
             filename=iris_csv_filename)
```

<https://github.com/scikit-learn/scikit-learn/blob/7b136e9/sklearn/datasets/base.py#L328>

```
from sklearn.datasets import load_iris
```

```
iris_dataset = load_iris()
```

```
print("Keys of iris_dataset:\n", iris_dataset.keys())
```

```
print(iris_dataset['DESCR'][:193] + "\n...")
```

```
print("Target names:", iris_dataset['target_names'])
```

```
print("Feature names:\n", iris_dataset['feature_names'])
```

```
print("Type of data:", type(iris_dataset['data']))
```

```
print("Shape of data:", iris_dataset['data'].shape)
```

```
print("First five rows of data:\n", iris_dataset['data'][:5])
```

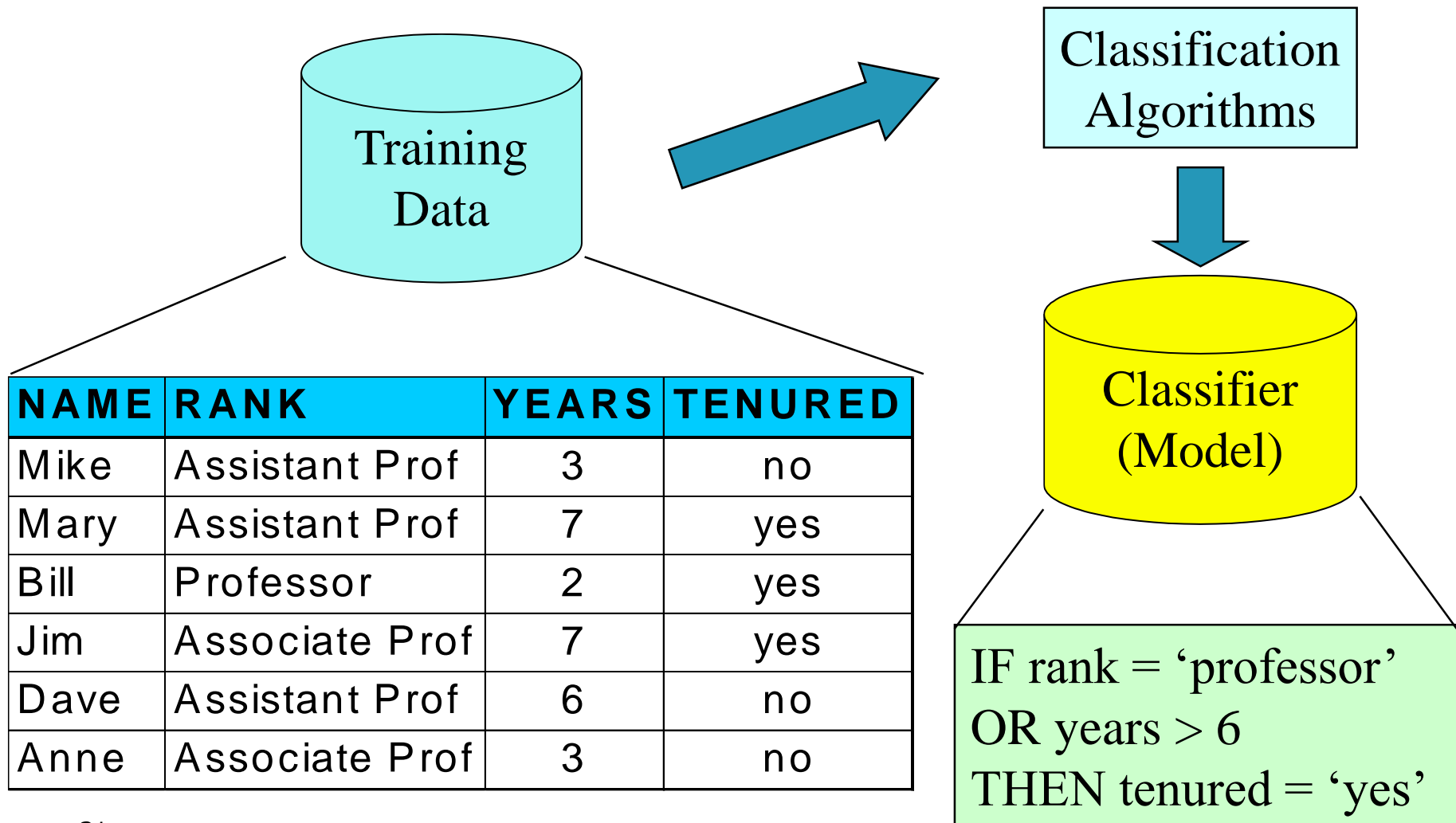
Classification 分類流程

Classification—A Two-Step Process

- **Model construction**: describing a set of predetermined classes
 - Each tuple/sample is assumed to belong to a predefined class, as determined by the **class label attribute**
 - The set of tuples used for model construction is **training set**
 - The model is represented as classification rules, decision trees, or mathematical formulae
- **Model usage**: for classifying future or unknown objects
 - **Estimate accuracy** of the model
 - The known label of test sample is compared with the classified result from the model
 - **Accuracy** rate is the percentage of test set samples that are correctly classified by the model
 - **Test set** is independent of training set (otherwise overfitting)
 - If the accuracy is acceptable, use the model to **classify new data**
- Note: If *the test set* is used to select models, it is called **validation (test) set**

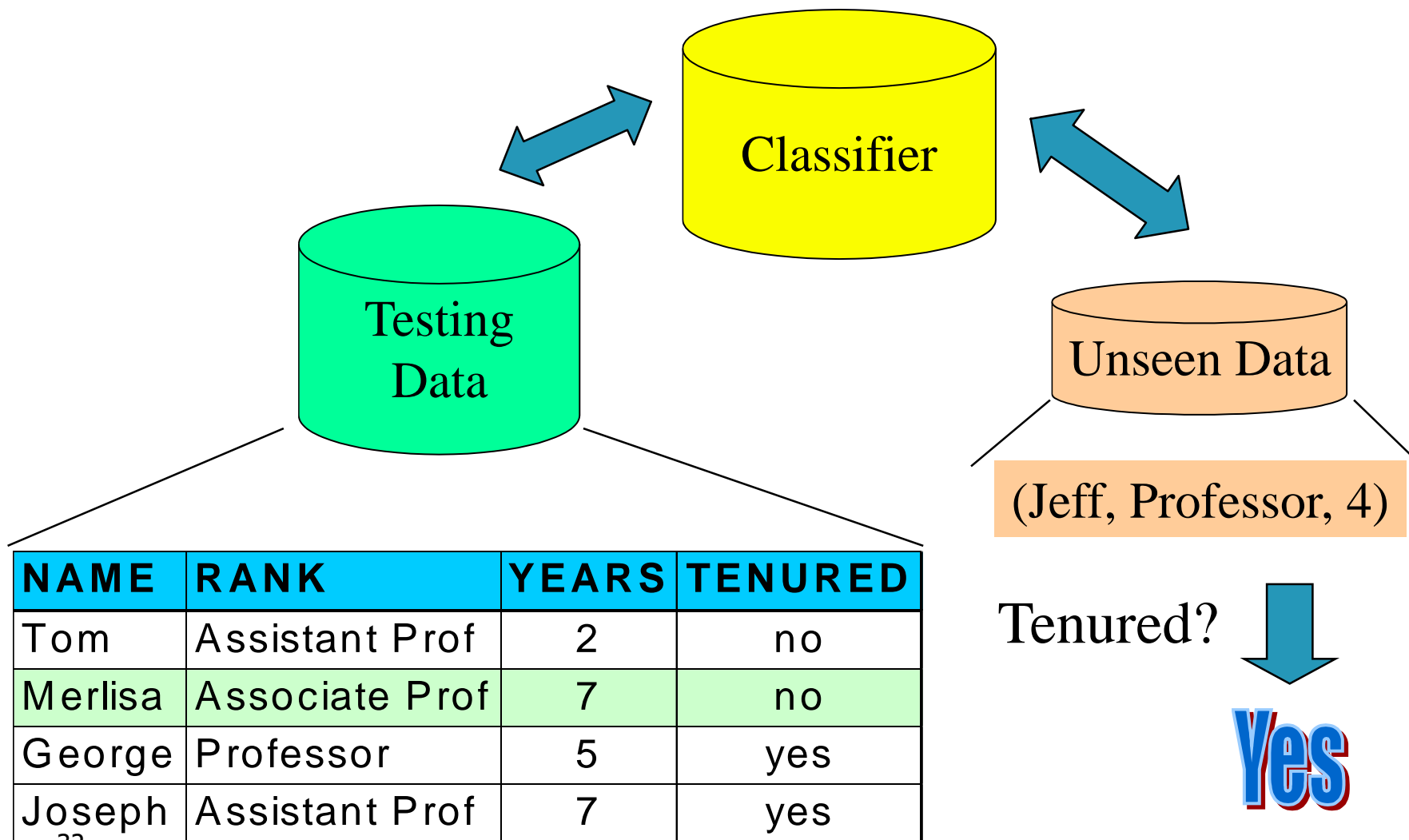
Process (1): Model Construction

訓練與模型建立



Process (2): Using the Model in Prediction

使用模型進行預測



分類演算法

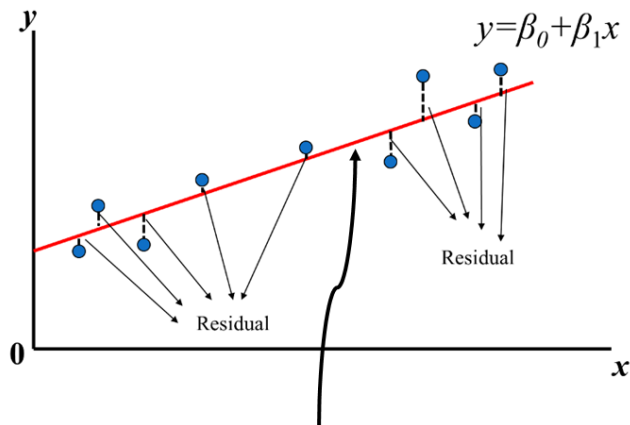
Supervised learning:分類演算法

- Linear Models :羅吉斯回歸(Logistic regression)
- k-Nearest Neighbors (KNN)
- Naive Bayes Classifiers
- **Decision Trees**
- **Ensembles of Decision Trees**
- Kernelized Support Vector Machines
- Neural Networks (Deep Learning)

羅吉斯回歸(Logistic regression) VS 線性回歸(Linear regression)

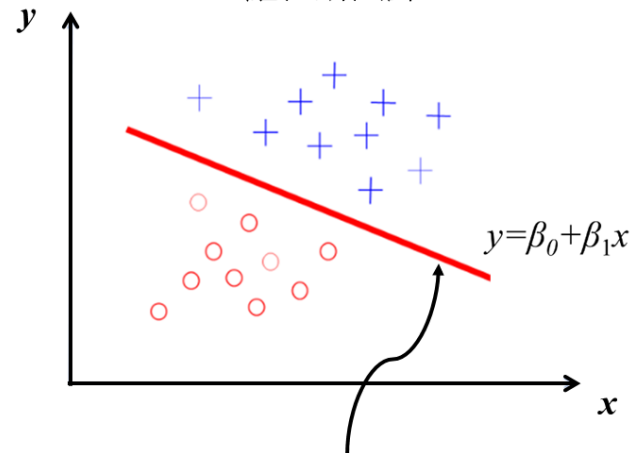
線性回歸是用來預測一個連續的值，羅吉斯回歸是用來分類。

線性回歸



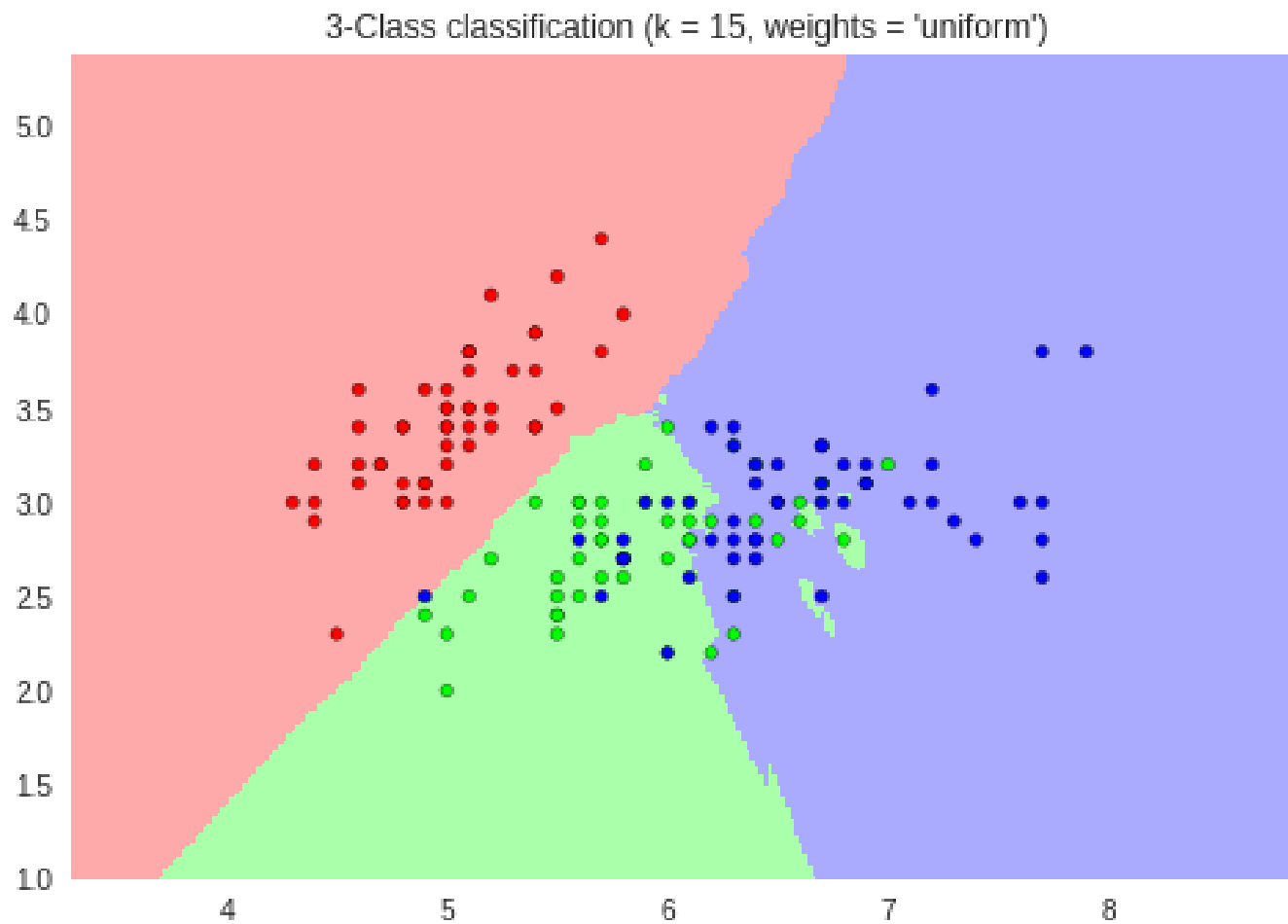
線性回歸是希望
「找到資料都可以盡量fix的那條紅線」

羅吉斯回歸




羅吉斯回歸希望
「找到那條紅線，讓資料可以區隔開來」

Nearest Neighbors Classification



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



WIKIPEDIA
The Free Encyclopedia

- Main page
- Contents
- Featured content
- Current events
- Random article
- Donate to Wikipedia
- Wikipedia store

Interaction


- Help
- About Wikipedia
- Community portal
- Recent changes
- Contact page

Tools

- What links here
- Related changes
- Upload file
- Special pages
- Permanent link

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)



Wiki Loves Monuments: Photograph a monument, help Wikipedia and win!
[Learn more](#)

Naive Bayes classifier

From Wikipedia, the free encyclopedia

In **machine learning**, **naïve Bayes classifiers** are a family of simple "**probabilistic classifiers**" based on applying **Bayes' theorem** with strong (naïve) **independence** assumptions between the features.

Naïve Bayes has been studied extensively since the 1960s. It was introduced (though not under that name) into the **text retrieval** community in the early 1960s,^[1] and remains a popular (baseline) method for **text categorization**, the problem of judging documents as belonging to one category or the other (such as **spam or legitimate**, sports or politics, etc.) with **word frequencies** as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including **support vector machines**.^[2] It also finds application in automatic **medical diagnosis**.^[3]

Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. **Maximum-likelihood** training can be done by evaluating a **closed-form expression**,^[4]⁷¹⁸ which takes **linear time**, rather than by expensive **iterative approximation** as used for many other types of classifiers.

In the **statistics** and **computer science** literature, naive Bayes models are known under a variety of names, including **simple Bayes** and **independence Bayes**.^[5] All these names reference the use of Bayes' theorem in the classifier's decision rule, but naïve Bayes is not (necessarily) a Bayesian method.^{[4][5]}

Support-vector machine

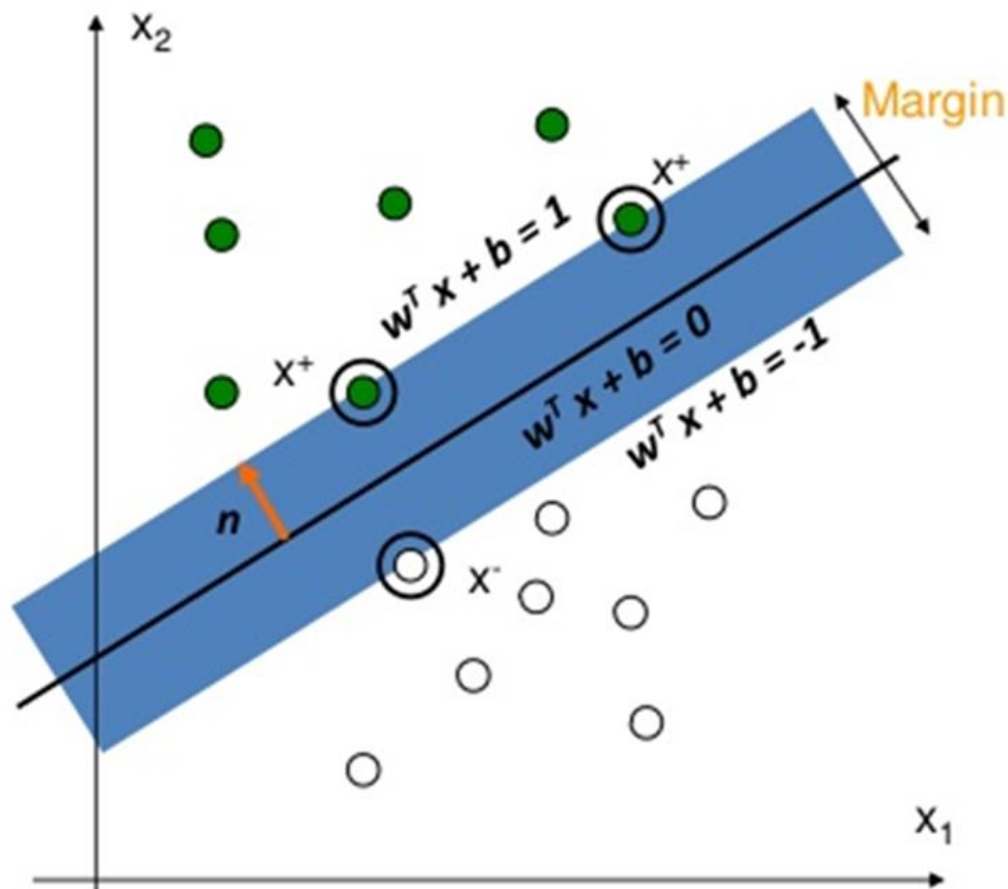
支援 向量 機

Formulation:

$$\text{minimize } \frac{1}{2} \|\mathbf{w}\|^2$$

such that

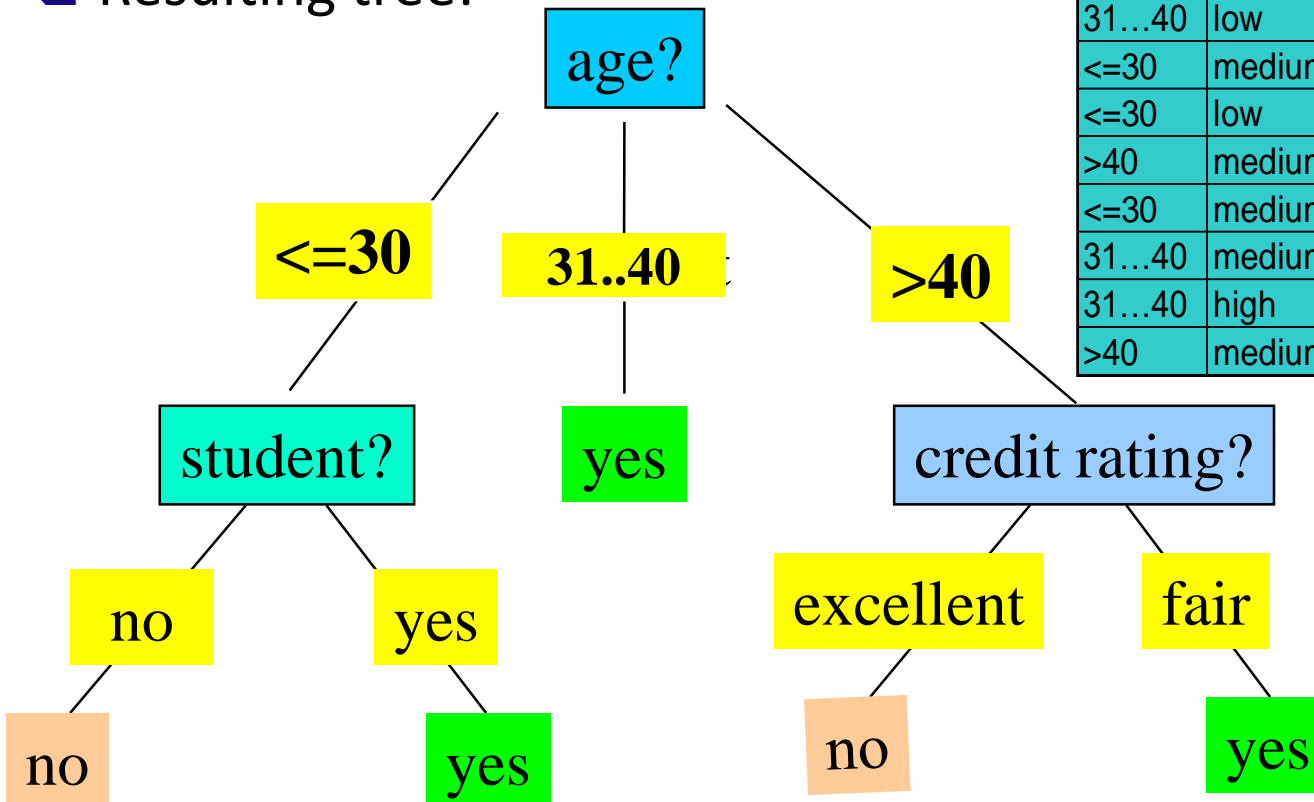
$$y_i(\mathbf{W} \mathbf{X} + \mathbf{b}) \geq 1$$



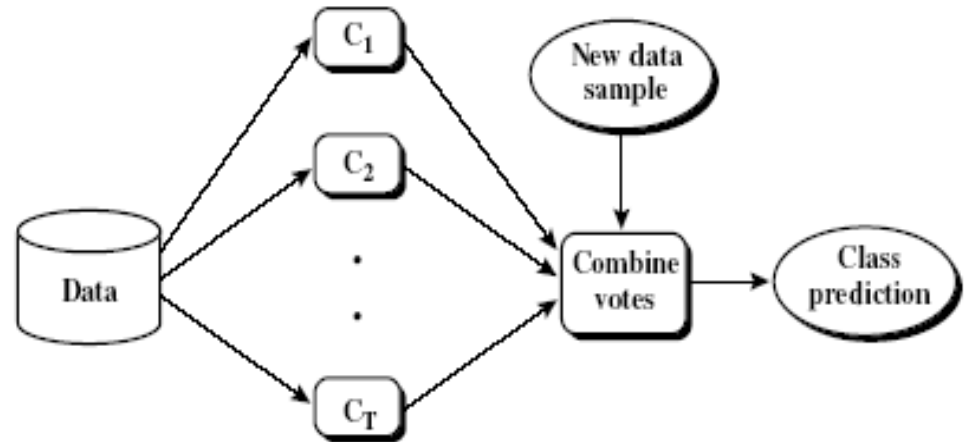
Decision Tree : An Example

- ❑ Training data set: Buys_computer
- ❑ The data set follows an example of Quinlan's ID3 (Playing Tennis)
- ❑ Resulting tree:

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no



Ensemble Methods: Increasing the Accuracy



- Ensemble methods
 - Use a combination of models to increase accuracy
 - Combine a series of k learned models, M_1, M_2, \dots, M_k , with the aim of creating an improved model M^*
- Popular ensemble methods
 - Bagging[裝袋模型]: averaging the prediction over a collection of classifiers
 - Boosting: weighted vote with a collection of classifiers
 - Stacking Ensemble: combining a set of heterogeneous classifiers[不同異質的分類器]

Ensemble Methods 整合方法|集成方法

Bagging

Random Forest
(Breiman 2001)

Extra-Trees

Extremely randomized trees
極端隨機樹
PierreGeurts 2006

Boosting

Adaboost

(Freund and Schapire, 1997)

GradientBoosting

Gradient Boosted Decision Tree (GBDT)

XGBoost 陳天奇 (2014)

Extreme Gradient Boosting

LightGBM(2017)

Light Gradient Boosting Machine

CatBoost (2017)

categorical boosting

Stacking

sklearn不直接支援stacking演算法



分類演算法 評估指標

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

Classification metrics

See the [Classification metrics](#) section of the user guide for further details.

<code>metrics.accuracy_score(y_true, y_pred[, ...])</code>	Accuracy classification score.
<code>metrics.auc(x, y[, reorder])</code>	Compute Area Under the Curve (AUC) using the trapezoidal rule
<code>metrics.average_precision_score(y_true, y_score)</code>	Compute average precision (AP) from prediction scores
<code>metrics.balanced_accuracy_score(y_true, y_pred)</code>	Compute the balanced accuracy
<code>metrics.brier_score_loss(y_true, y_prob[, ...])</code>	Compute the Brier score.
<code>metrics.classification_report(y_true, y_pred)</code>	Build a text report showing the main classification metrics
<code>metrics.cohen_kappa_score(y1, y2[, labels, ...])</code>	Cohen's kappa: a statistic that measures inter-annotator agreement.
<code>metrics.confusion_matrix(y_true, y_pred[, ...])</code>	Compute confusion matrix to evaluate the accuracy of a classification
<code>metrics.f1_score(y_true, y_pred[, labels, ...])</code>	Compute the F1 score, also known as balanced F-score or F-measure
<code>metrics.fbeta_score(y_true, y_pred, beta[, ...])</code>	Compute the F-beta score
<code>metrics.hamming_loss(y_true, y_pred[, ...])</code>	Compute the average Hamming loss.
<code>metrics.hinge_loss(y_true, pred_decision[, ...])</code>	Average hinge loss (non-regularized)
<code>metrics.jaccard_score(y_true, y_pred[, ...])</code>	Jaccard similarity coefficient score
<code>metrics.log_loss(y_true, y_pred[, eps, ...])</code>	Log loss, aka logistic loss or cross-entropy loss.
<code>metrics.matthews_corrcoef(y_true, y_pred[, ...])</code>	Compute the Matthews correlation coefficient (MCC)
<code>metrics.multilabel_confusion_matrix(y_true, ...)</code>	Compute a confusion matrix for each class or sample
<code>metrics.precision_recall_curve(y_true, ...)</code>	Compute precision-recall pairs for different probability thresholds
<code>metrics.precision_recall_fscore_support(...)</code>	Compute precision, recall, F-measure and support for each class
<code>metrics.precision_score(y_true, y_pred[, ...])</code>	Compute the precision
<code>metrics.recall_score(y_true, y_pred[, ...])</code>	Compute the recall
<code>metrics.roc_auc_score(y_true, y_score[, ...])</code>	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
<code>metrics.roc_curve(y_true, y_score[, ...])</code>	Compute Receiver operating characteristic (ROC)
<code>metrics.zero_one_loss(y_true, y_pred[, ...])</code>	Zero-one classification loss.

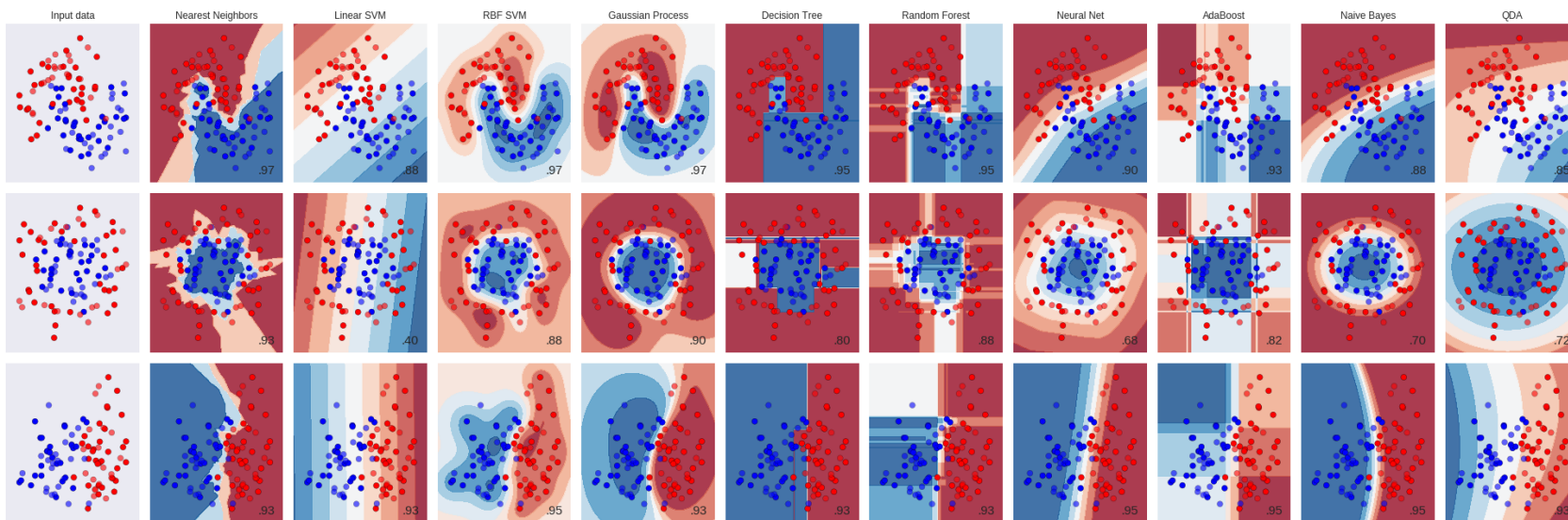
Classifier comparison 分類器的各種比較

https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html
[#sphx-glr-auto-examples-classification-plot-classifier-comparison-py](#)

A comparison of a several classifiers in scikit-learn on synthetic datasets. The point of this example is to illustrate the nature of decision boundaries of different classifiers. This should be taken with a grain of salt, as the intuition conveyed by these examples does not necessarily carry over to real datasets.

Particularly in high-dimensional spaces, data can more easily be separated linearly and the simplicity of classifiers such as naive Bayes and linear SVMs might lead to better generalization than is achieved by other classifiers.

The plots show training points in solid colors and testing points semi-transparent. The lower right shows the classification accuracy on the test set.



```
names = ["Nearest Neighbors", "Linear SVM", "RBF SVM", "Gaussian Process",  
        "Decision Tree", "Random Forest", "Neural Net", "AdaBoost",  
        "Naive Bayes", "QDA"]
```

```
classifiers = [  
    KNeighborsClassifier(3),  
    SVC(kernel="linear", C=0.025),  
    SVC(gamma=2, C=1),  
    GaussianProcessClassifier(1.0 * RBF(1.0)),  
    DecisionTreeClassifier(max_depth=5),  
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),  
    MLPClassifier(alpha=1),  
    AdaBoostClassifier(),  
    GaussianNB(),  
    QuadraticDiscriminantAnalysis()]
```

```
X, y = make_classification(n_features=2, n_redundant=0, n_informative=2,  
                          random_state=1, n_clusters_per_class=1)  
rng = np.random.RandomState(2)  
X += 2 * rng.uniform(size=X.shape)  
linearly_separable = (X, y)
```

```
datasets = [make_moons(noise=0.3, random_state=0),  
            make_circles(noise=0.2, factor=0.5, random_state=1),  
            linearly_separable  
            ]
```