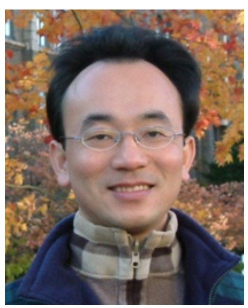
資料探勘
(Data Mining)
監督學習：分類和預測
(Supervised Learning: Classification and Prediction)

1092DM07
MBA, IM, NTPU (M5026) (Spring 2021)
Tue 2, 3, 4 (9:10-12:00) (B8F40)



Min-Yuh Day
戴敏育
Associate Professor
副教授
Institute of Information Management, National Taipei University
國立臺北大學 資訊管理研究所
https://web.ntpu.edu.tw/~myday

2021-04-27

課程大綱 (Syllabus)

週次 (Week)　日期 (Date)　內容 (Subject/Topics)

1 2021/02/23 資料探勘介紹 (Introduction to data mining)

2 2021/03/02 ABC：人工智慧，大數據，雲端運算
(ABC: AI, Big Data, Cloud Computing)

3 2021/03/09 Python資料探勘的基礎
(Foundations of Data Mining in Python)

4 2021/03/16 資料科學與資料探勘：發現，分析，可視化和呈現數據
(Data Science and Data Mining:
Discovering, Analyzing, Visualizing and Presenting Data)

5 2021/03/23 非監督學習：關聯分析，購物籃分析
(Unsupervised Learning: Association Analysis,
Market Basket Analysis)

6 2021/03/30 資料探勘個案研究 I
(Case Study on Data Mining I)

課程大綱 (Syllabus)

週次 (Week)　日期 (Date)　內容 (Subject/Topics)

7　2021/04/06　放假一天 (Day off)

8　2021/04/13　非監督學習：集群分析，行銷市場區隔
(Unsupervised Learning: Cluster Analysis, Market Segmentation)

9　2021/04/20　期中報告 (Midterm Project Report)

10　2021/04/27　監督學習：分類和預測
(Supervised Learning: Classification and Prediction)

11　2021/05/04　機器學習和深度學習
(Machine Learning and Deep Learning)

12　2021/05/11　卷積神經網絡
(Convolutional Neural Networks)

課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)

13 2021/05/18 資料探勘個案研究 II
(Case Study on Data Mining II)

14 2021/05/25 遞歸神經網絡
(Recurrent Neural Networks)

15 2021/06/01 強化學習
(Reinforcement Learning)

16 2021/06/08 社交網絡分析
(Social Network Analysis)

17 2021/06/15 期末報告 I (Final Project Report I)

18 2021/06/22 期末報告 II (Final Project Report II)

Supervised Learning: Classification and Prediction

Outline

- Supervised Learning
- Classification and Prediction
- Decision Tree (DT)
  - Information Gain (IG)
- Support Vector Machine (SVM)
- Data Mining Evaluation
  - Accuracy
  - Precision
  - Recall
  - F1 score (F-measure) (F-score)

Data Mining Tasks & Methods

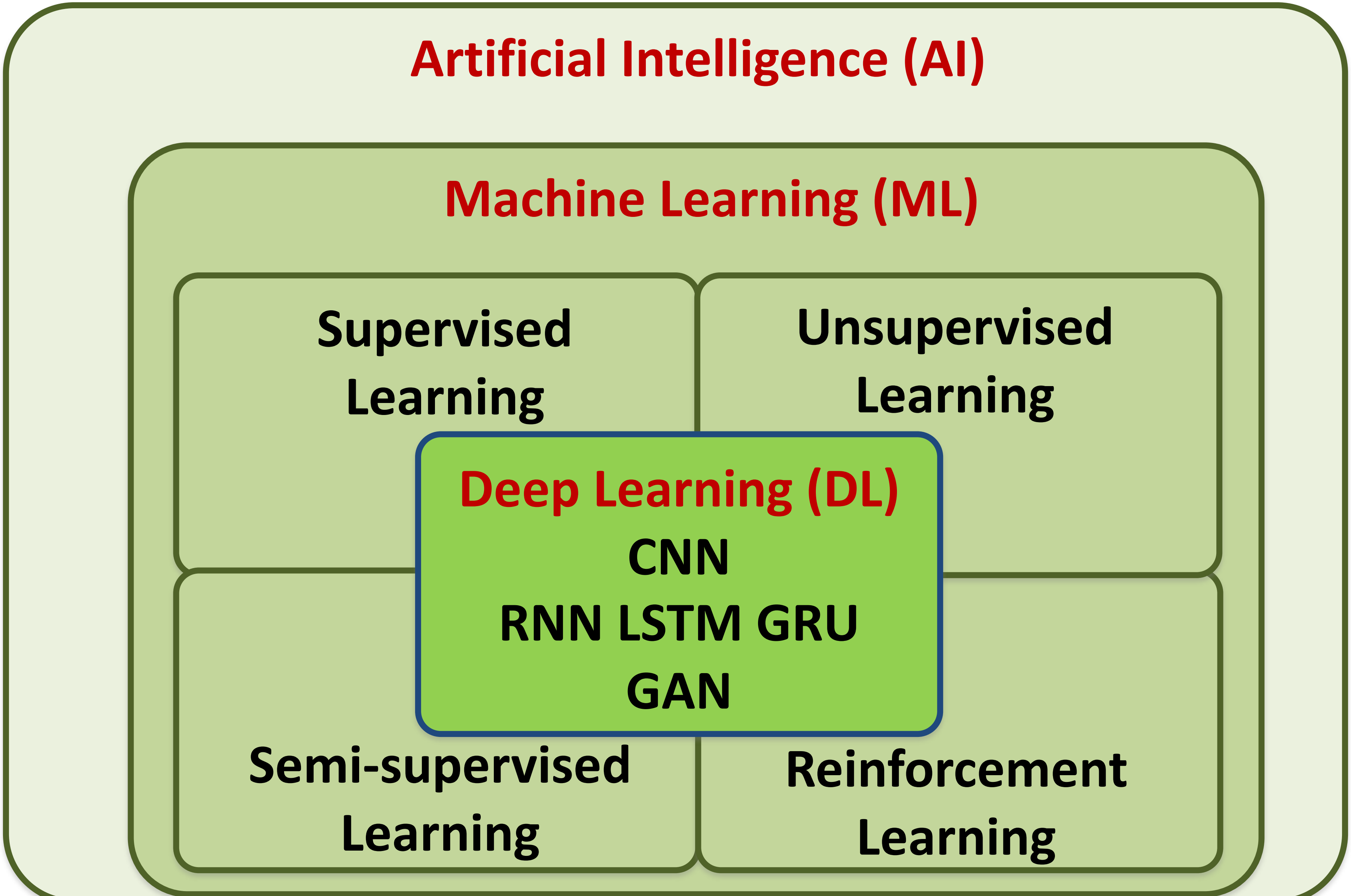
Prediction
Classification

Supervised Learning: Classification and Prediction

Data Mining Tasks & Methods	Data Mining Algorithms	Learning Type
Prediction		
Classification	Decision Trees, Neural Networks, Support Vector Machines, kNN, Naïve Bayes, GA	Supervised
Regression	Linear/Nonlinear Regression, ANN, Regression Trees, SVM, kNN, GA	Supervised
Time series	Autoregressive Methods, Averaging Methods, Exponential Smoothing, ARIMA	Supervised
Association		
Market-basket	Apriori, OneR, ZeroR, Eclat, GA	Unsupervised
Link analysis	Expectation Maximization, Apriori Algorithm, Graph-Based Matching	Unsupervised
Sequence analysis	Apriori Algorithm, FP-Growth, Graph-Based Matching	Unsupervised
Segmentation		
Clustering	*k*-means, Expectation Maximization (EM)	Unsupervised
Outlier analysis	*k*-means, Expectation Maximization (EM)	Unsupervised

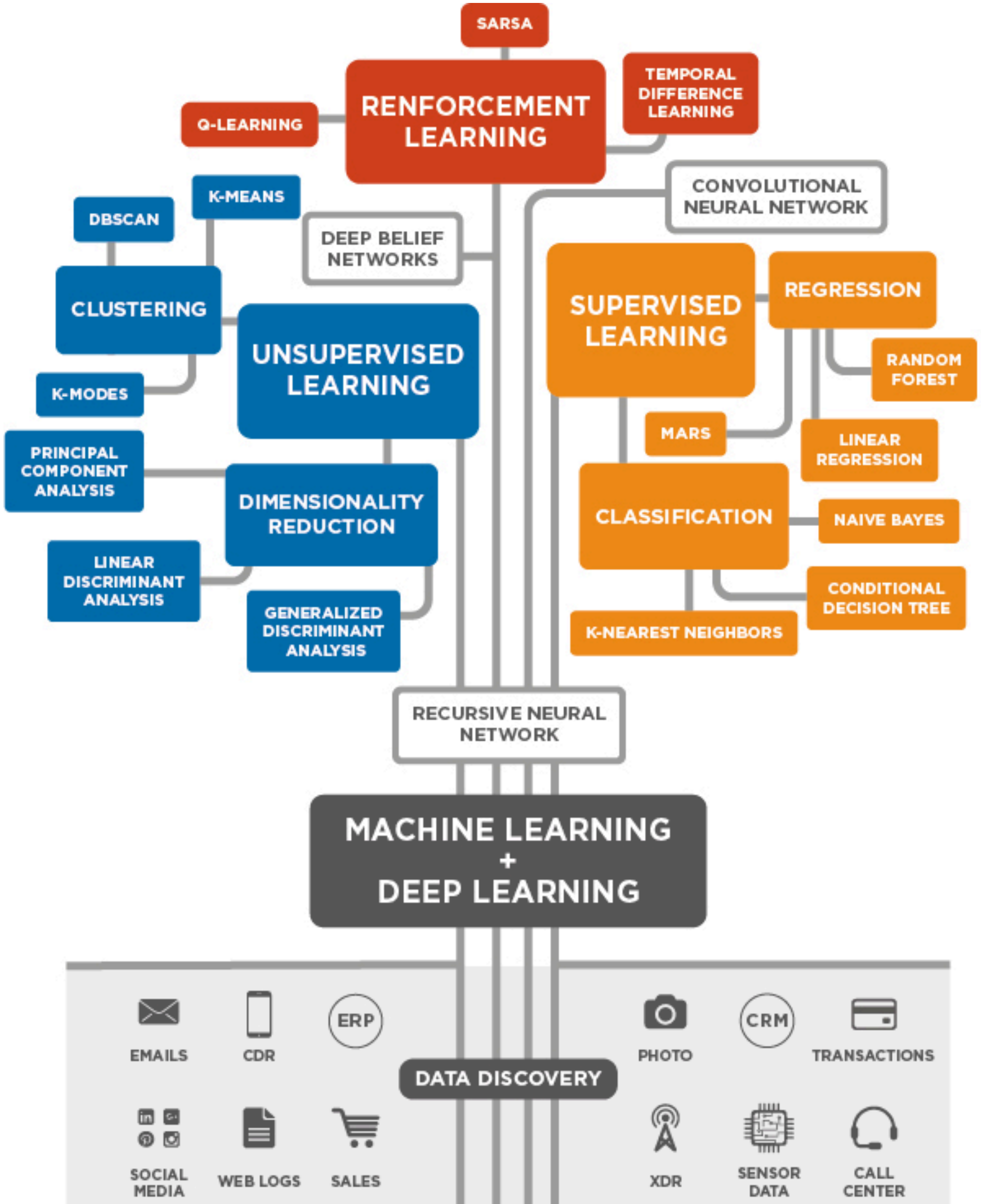
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

AI, ML, DL



Source: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/deep_learning.html

3 Machine Learning Algorithms



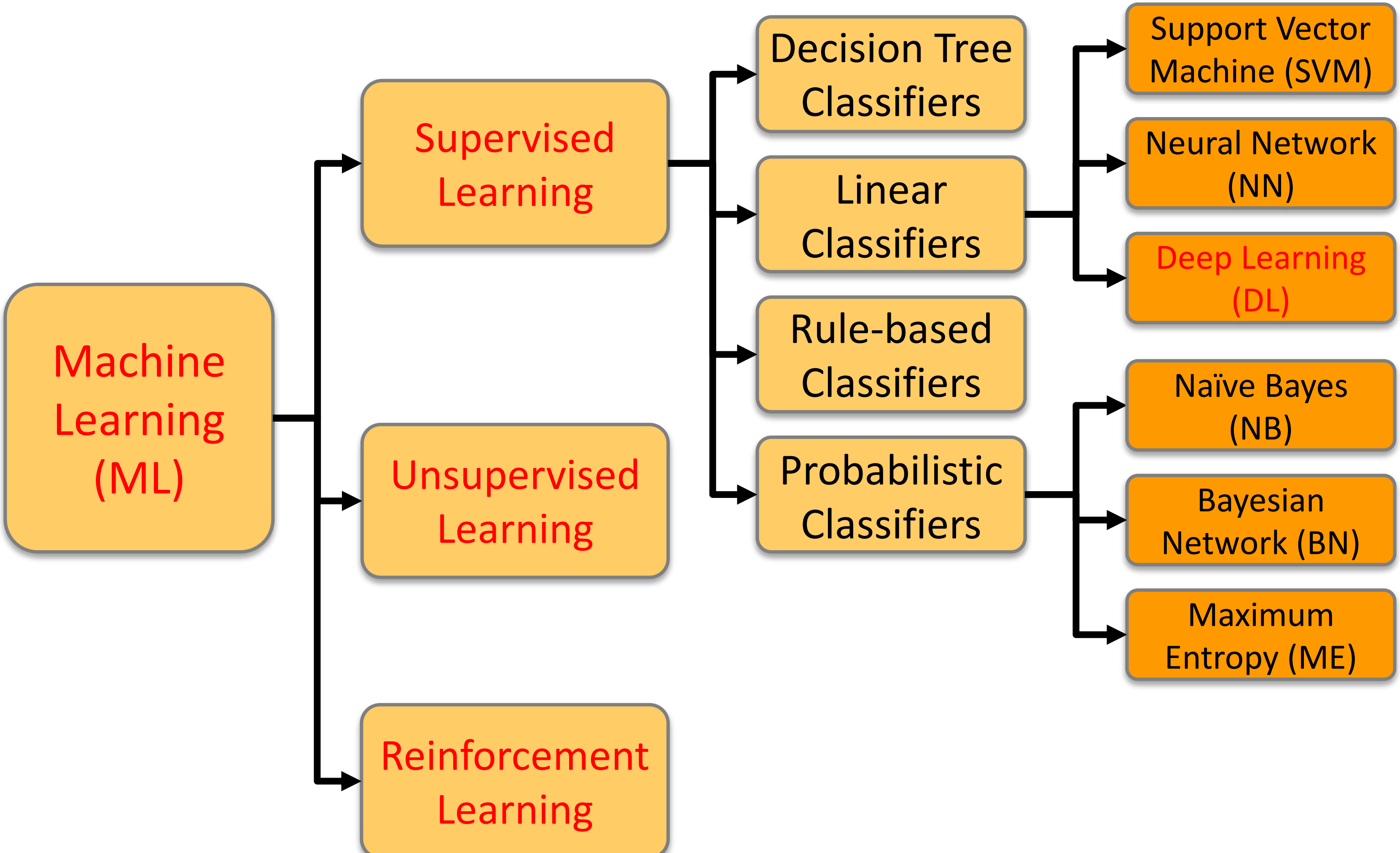
Source: Enrico Galimberti, http://blogs.teradata.com/data-points/tree-machine-learning-algorithms/

Machine Learning Models

- Deep Learning
- Kernel
- Association rules
- Ensemble
- Decision tree
- Dimensionality reduction
- Clustering
- Regression Analysis
- Bayesian
- Instance based

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

Machine Learning (ML) / Deep Learning (DL)



Source: Jesus Serrano-Guerrero, Jose A. Olivas, Francisco P. Romero, and Enrique Herrera-Viedma (2015), "Sentiment analysis: A review and comparative analysis of web services," Information Sciences, 311, pp. 18-38.

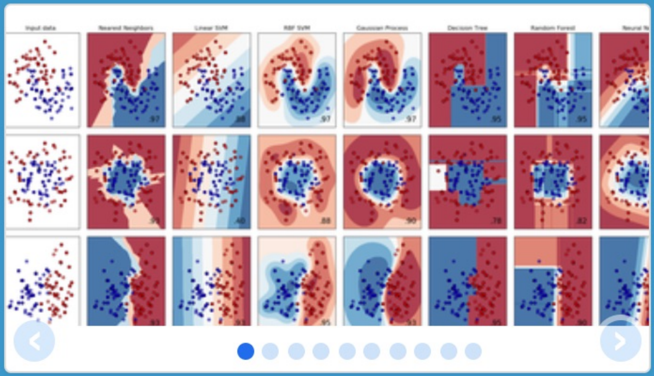
Scikit-Learn

Machine Learning in Python

Scikit-Learn

Home Installation Documentation Examples

Google Custom Search



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, … — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, … — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, … — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

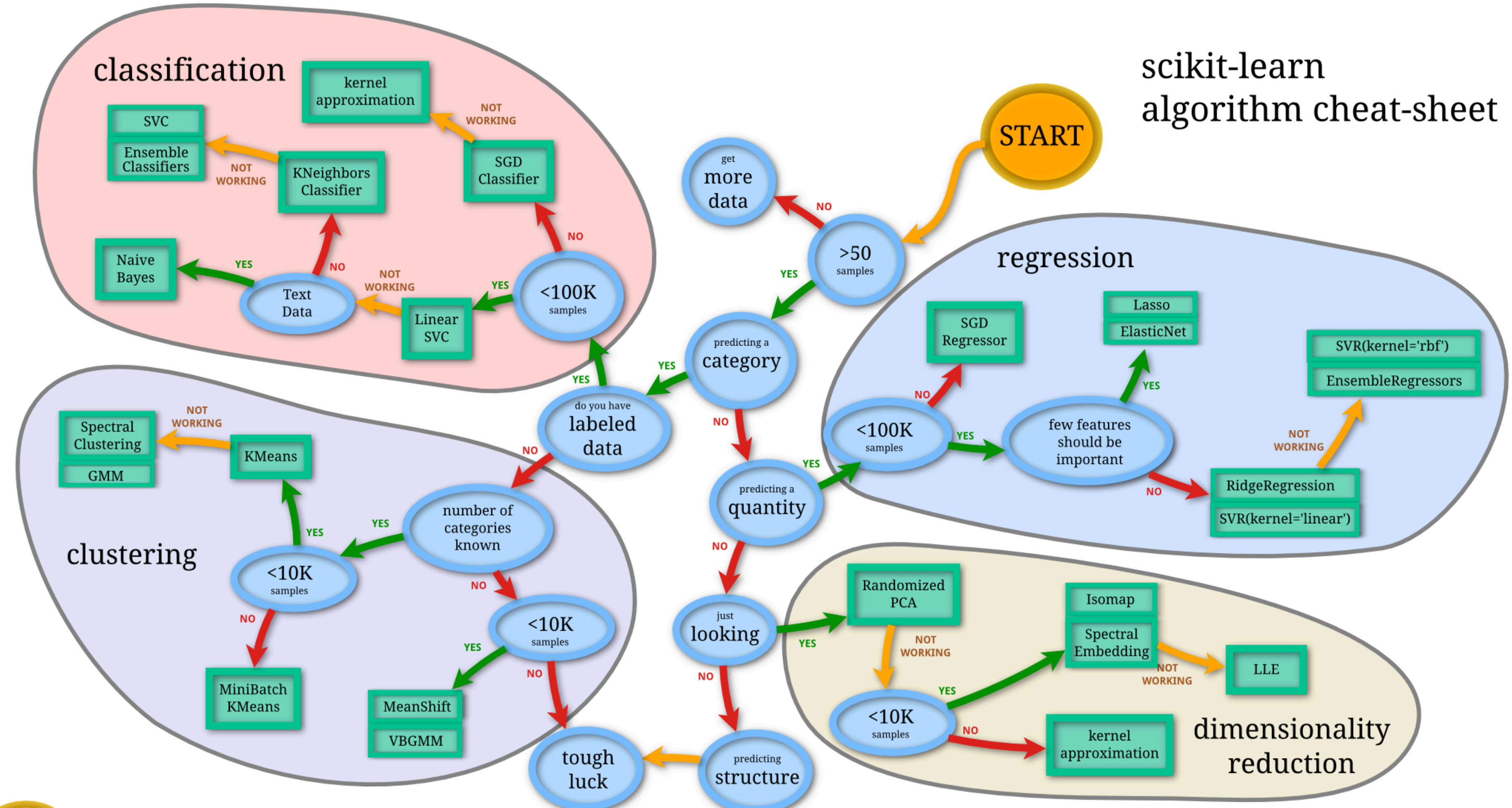
Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

Source: http://scikit-learn.org/

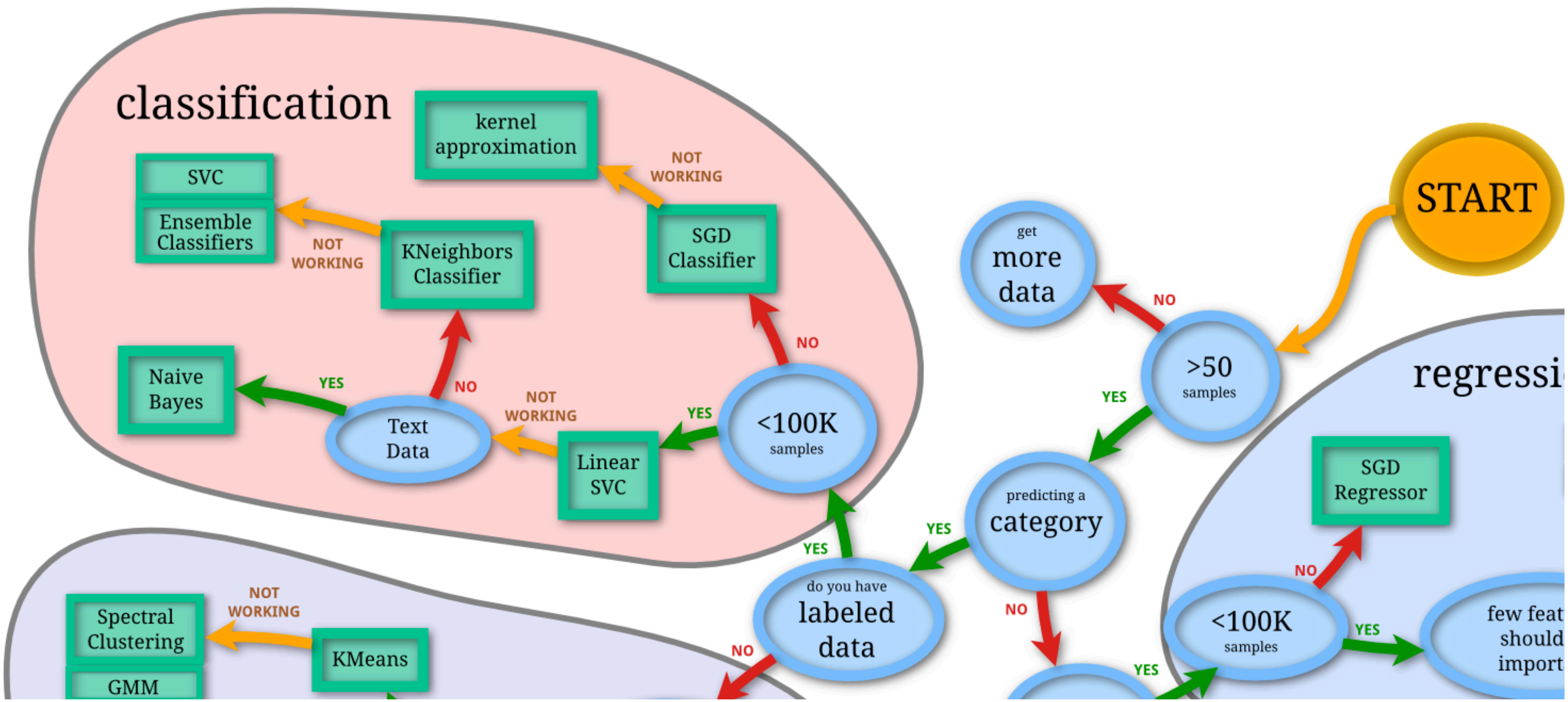
Scikit-Learn Machine Learning Map



Back

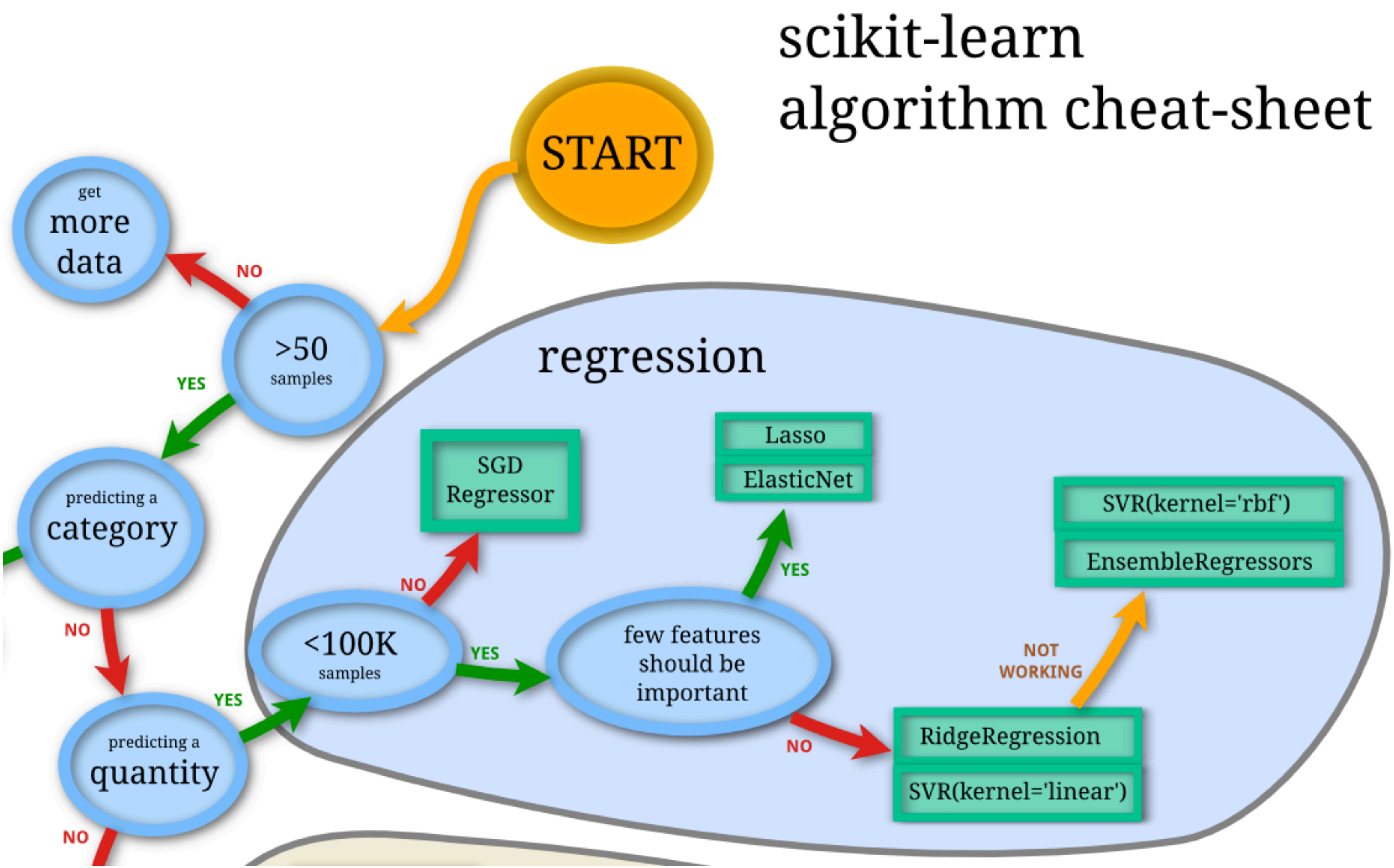
Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-Learn Machine Learning Map



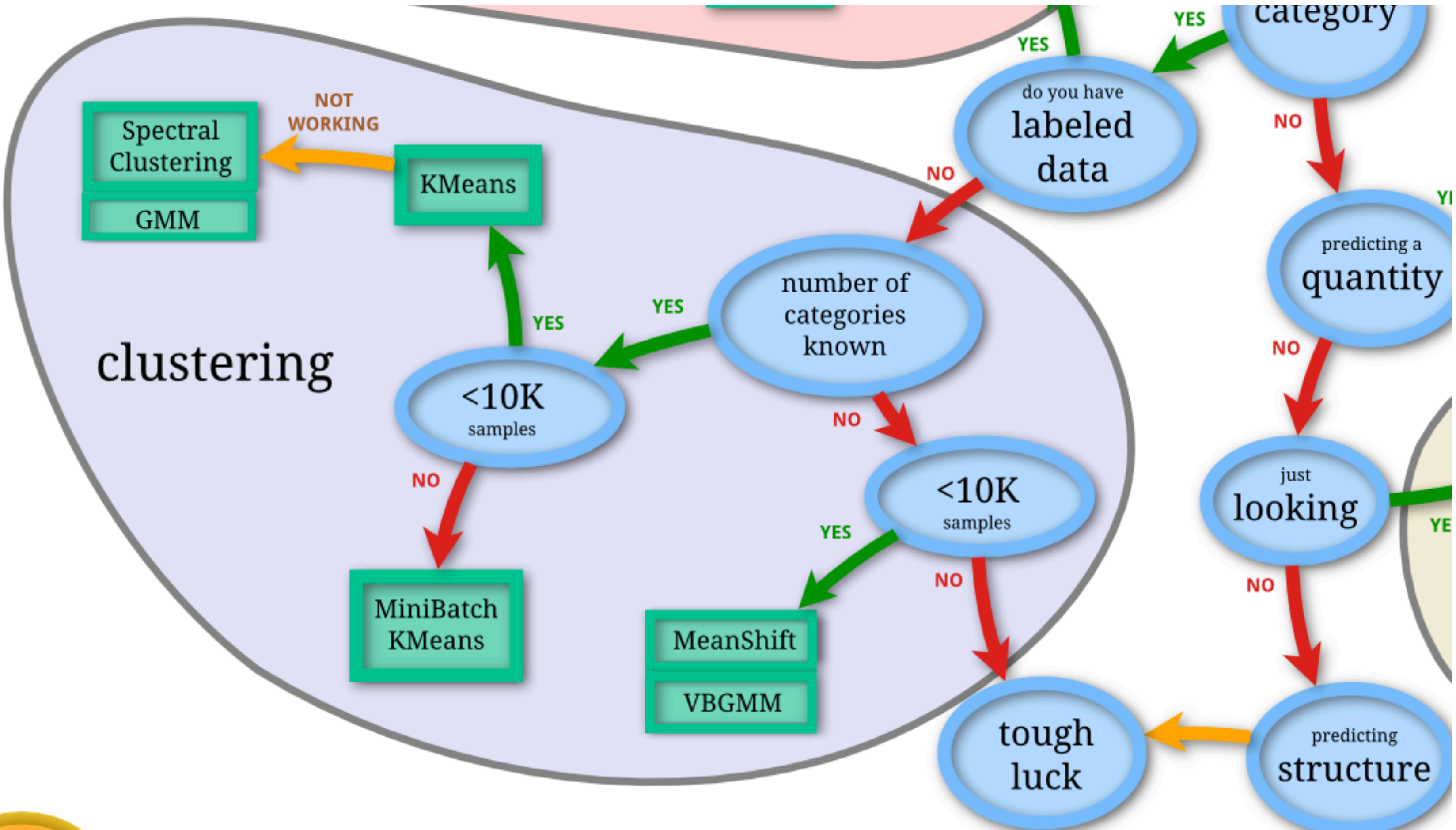
Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-Learn Machine Learning Map



Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-Learn Machine Learning Map



Back

Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Classification vs. 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
- Prediction
  - models continuous-valued functions
    - i.e., predicts unknown or missing values
- Typical applications
  - Credit approval
  - Target marketing
  - Medical diagnosis
  - Fraud detection

Source: Han & Kamber (2006)

Data Mining Methods: Classification

- Most frequently used DM method
- Part of the machine-learning family
- Employ supervised learning
- Learn from past data, classify new data
- The output variable is categorical (nominal or ordinal) in nature
- Classification versus regression?
- Classification versus clustering?

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

Classification Techniques

- **Decision Tree analysis (DT)**
- Statistical analysis
- **Neural networks (NN)**
- **Deep Learning (DL)**
- **Support Vector Machines (SVM)**
- Case-based reasoning
- Bayesian classifiers
- Genetic algorithms (GA)
- Rough sets

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

Example of Classification

- Loan Application Data
  - Which loan applicants are “safe” and which are “risky” for the bank?
  - “Safe” or “risky” for load application data
- Marketing Data
  - Whether a customer with a given profile will buy a new computer?
  - “yes” or “no” for marketing data
- **Classification**
  - Data analysis task
  - A model or **Classifier** is constructed to predict categorical labels
    - Labels: “safe” or “risky”; “yes” or “no”; “treatment A”, “treatment B”, “treatment C”

Source: Han & Kamber (2006)

What Is Prediction?

- (Numerical) prediction is similar to classification
  - construct a model
  - use model to predict continuous or ordered value for a given input
- Prediction is different from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions
- Major method for prediction: regression
  - model the relationship between one or more *independent* or **predictor** variables and a *dependent* or **response** variable
- Regression analysis
  - Linear and multiple regression
  - Non-linear regression
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees

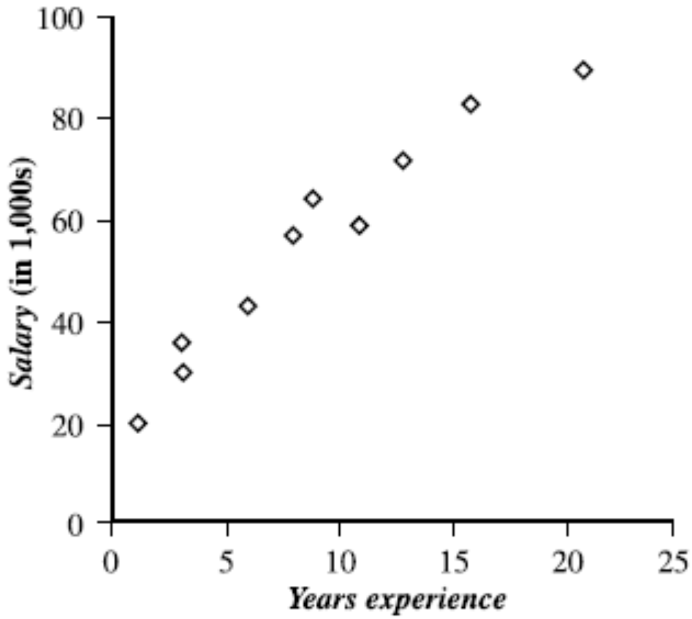
Source: Han & Kamber (2006)

Prediction Methods

- Linear Regression
- Nonlinear Regression
- Other Regression Methods

Salary data.

x years experience	y salary (in \$1000s)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83



Source: Han & Kamber (2006)

Classification and Prediction

- Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends.
- Classification
  - Effective and scalable methods have been developed for decision trees induction, Naive Bayesian classification, Bayesian belief network, rule-based classifier, Backpropagation, Support Vector Machine (SVM), associative classification, nearest neighbor classifiers, and case-based reasoning, and other classification methods such as genetic algorithms, rough set and fuzzy set approaches.
- Prediction
  - Linear, nonlinear, and generalized linear models of regression can be used for prediction. Many nonlinear problems can be converted to linear problems by performing transformations on the predictor variables. Regression trees and model trees are also used for prediction.

Source: Han & Kamber (2006)

Classification —A Two-Step Process

1. **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
2. **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 over-fitting will occur
   - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

Source: Han & Kamber (2006)

Supervised Learning vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - 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

Source: Han & Kamber (2006)

Issues Regarding Classification and Prediction: Data Preparation

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
  - Attribute subset selection
    - Feature Selection in machine learning
- Data transformation
  - Generalize and/or normalize data
  - Example
    - Income: low, medium, high

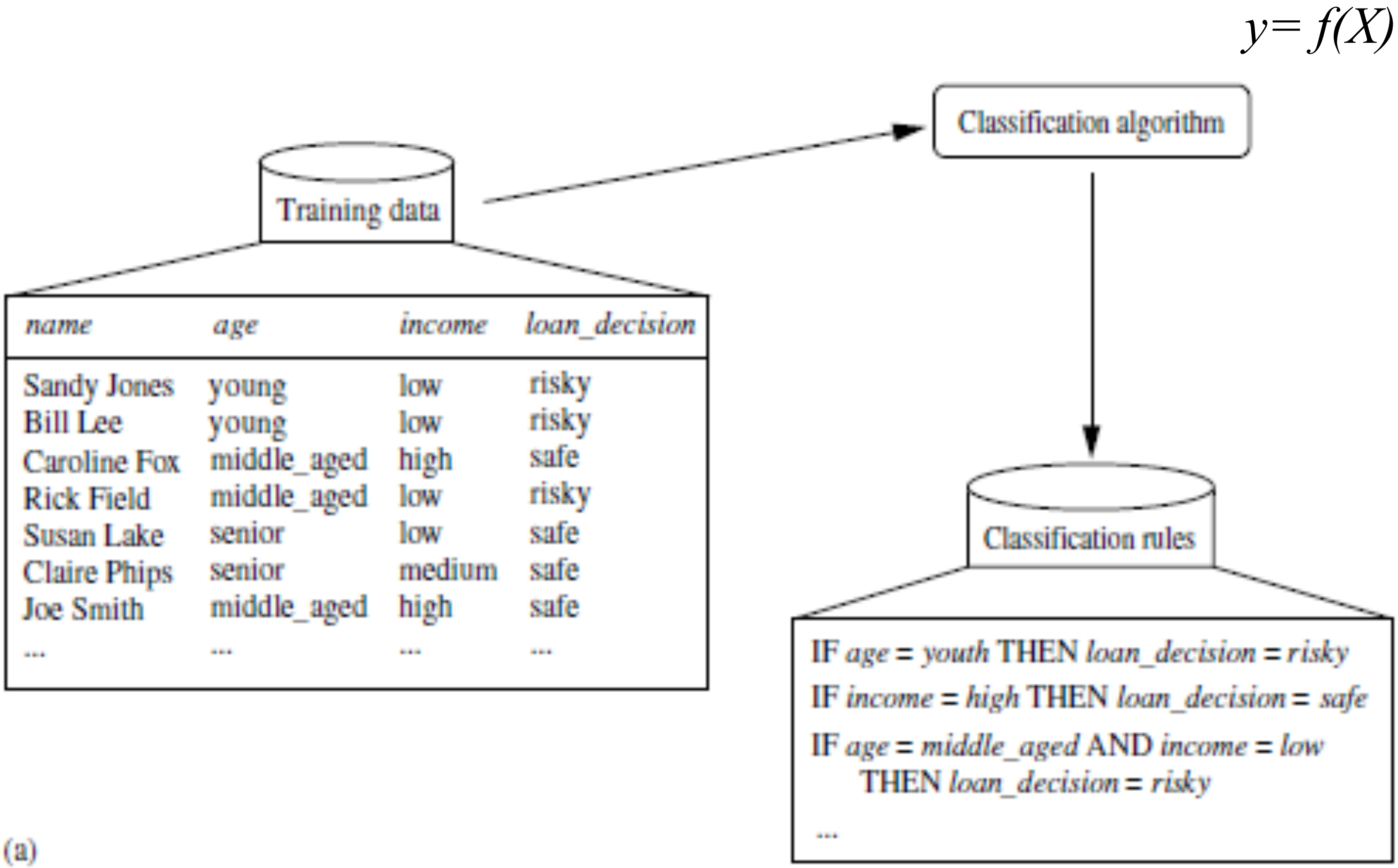
Source: Han & Kamber (2006)

Issues: Evaluating Classification and Prediction Methods

- Accuracy
  - classifier accuracy: predicting class label
  - predictor accuracy: guessing value of predicted attributes
  - estimation techniques: cross-validation and bootstrapping
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness
  - handling noise and missing values
- Scalability
  - ability to construct the classifier or predictor efficiently given large amounts of data
- Interpretability
  - understanding and insight provided by the model

Source: Han & Kamber (2006)

Data Classification Process 1: Learning (Training) Step
(a) Learning: Training data are analyzed by classification algorithm

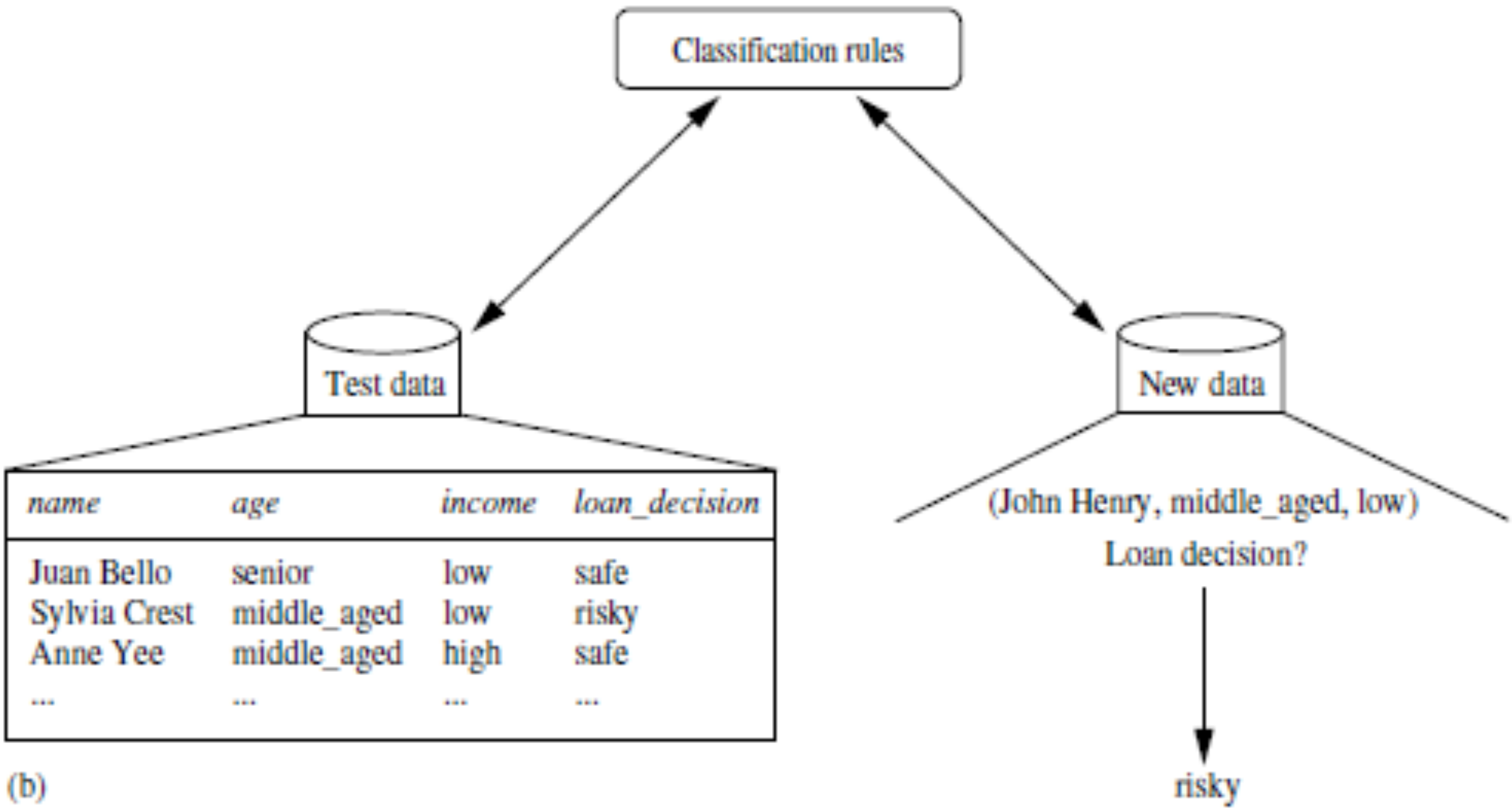


name	*age*	*income*	*loan_decision*
Sandy Jones	young	low	risky
Bill Lee	young	low	risky
Caroline Fox	middle_aged	high	safe
Rick Field	middle_aged	low	risky
Susan Lake	senior	low	safe
Claire Phips	senior	medium	safe
Joe Smith	middle_aged	high	safe
...	...	...	...

Source: Han & Kamber (2006)

Data Classification Process 2

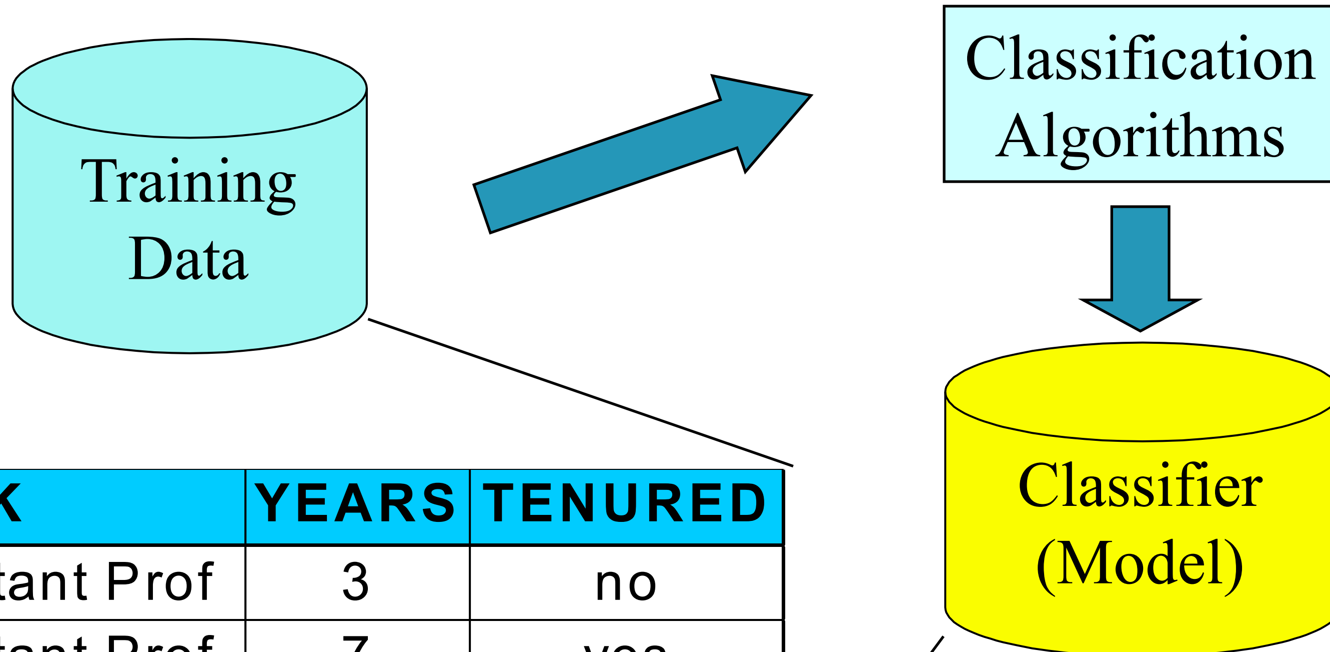
(b) Classification: Test data are used to estimate the accuracy of the classification rules.



name	*age*	*income*	*loan_decision*
Juan Bello	senior	low	safe
Sylvia Crest	middle_aged	low	risky
Anne Yee	middle_aged	high	safe
...	...	...	...

Source: Han & Kamber (2006)

Process (1): Model Construction

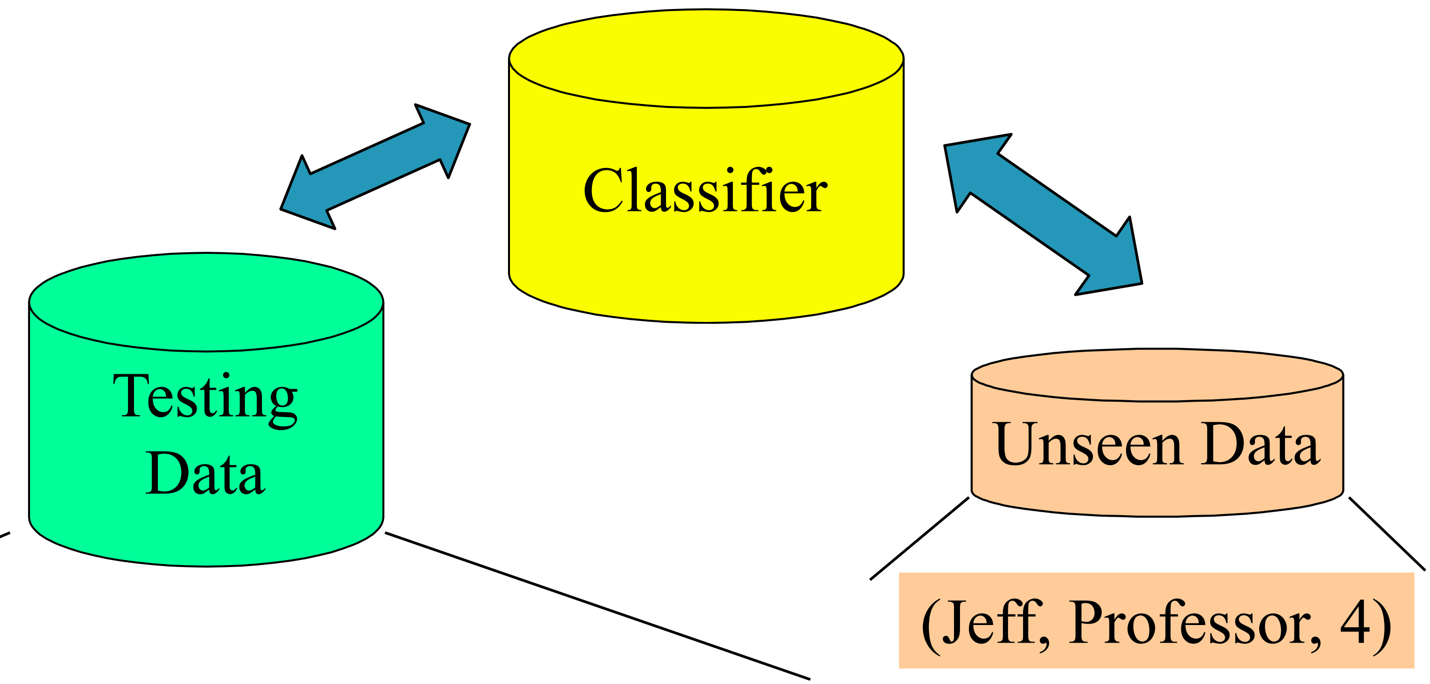


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

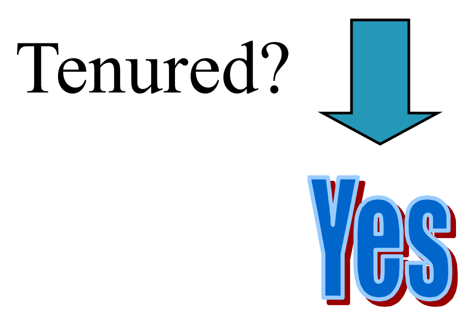
IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Source: Han & Kamber (2006)

Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes



Source: Han & Kamber (2006)

Decision Trees

Decision Trees

A general algorithm for decision tree building

- Employs the divide and conquer method
- Recursively divides a training set until each division consists of examples from one class
  1. Create a root node and assign all of the training data to it
  2. Select the best splitting attribute
  3. Add a branch to the root node for each value of the split. Split the data into mutually exclusive subsets along the lines of the specific split
  4. Repeat the steps 2 and 3 for each and every leaf node until the stopping criteria is reached

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

Decision Trees

- DT algorithms mainly differ on
  - Splitting criteria
    - Which variable to split first?
    - What values to use to split?
    - How many splits to form for each node?
  - Stopping criteria
    - When to stop building the tree
  - Pruning (generalization method)
    - Pre-pruning versus post-pruning
- Most popular DT algorithms include
  - ID3, C4.5, C5; CART; CHAID; M5

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

Decision Trees

- Alternative splitting criteria
  - Gini index determines the purity of a specific class as a result of a decision to branch along a particular attribute/value
    - Used in CART
  - Information gain uses entropy to measure the extent of uncertainty or randomness of a particular attribute/value split
    - Used in ID3, C4.5, C5
  - Chi-square statistics (used in CHAID)

Classification by Decision Tree Induction Training Dataset

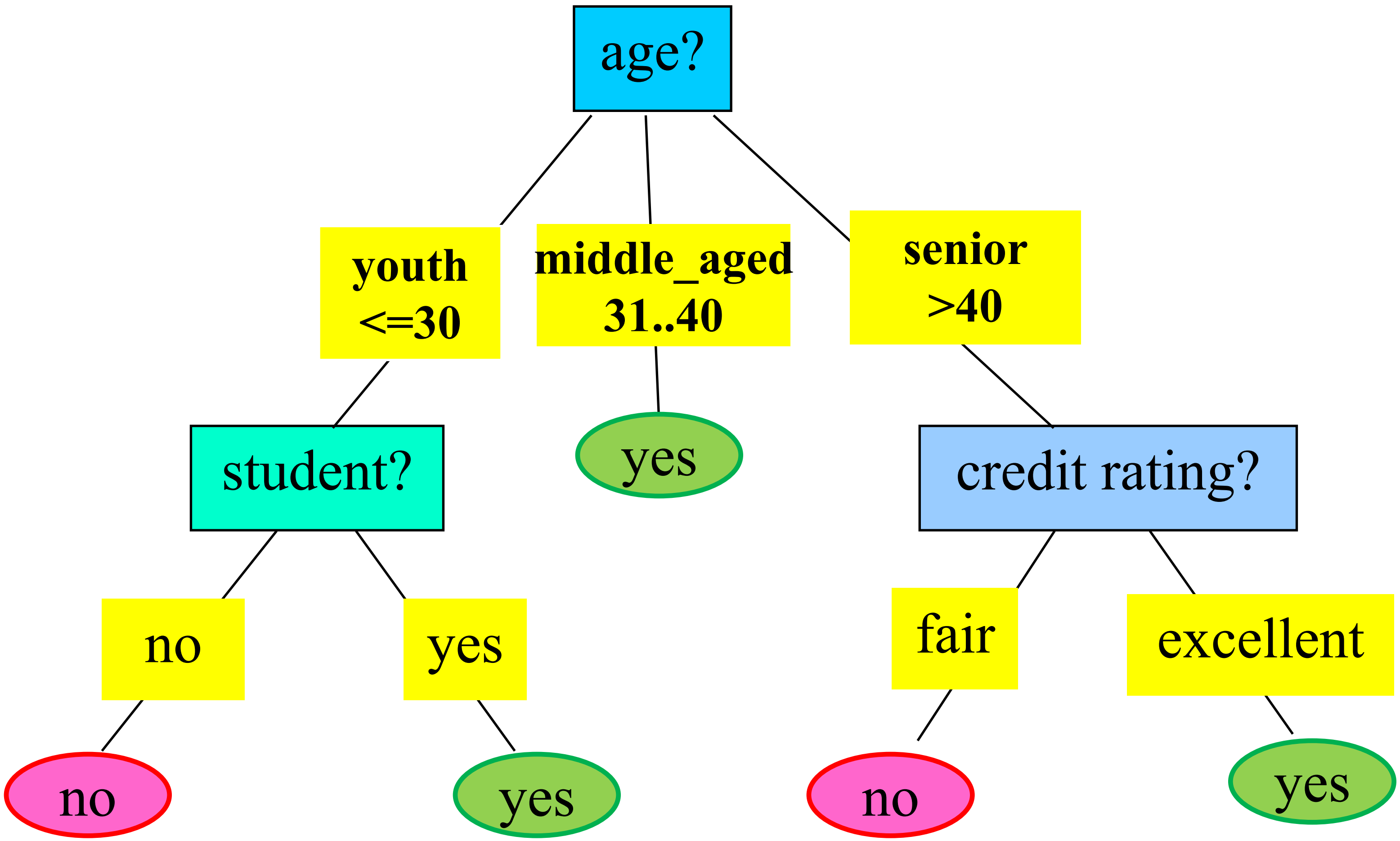
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent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

This follows an example of Quinlan's ID3 (Playing Tennis)

Source: Han & Kamber (2006)

Classification by Decision Tree Induction

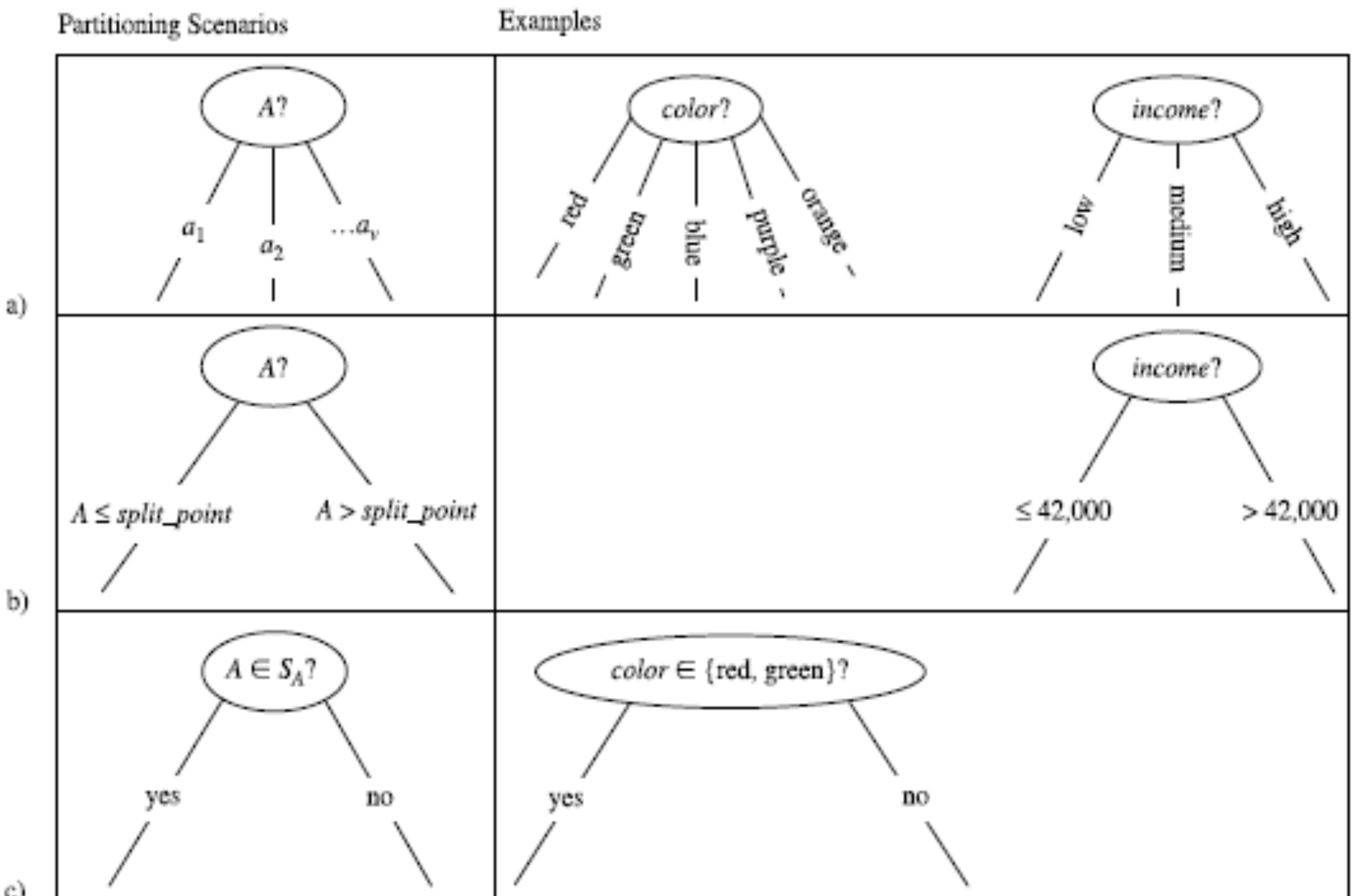
Output: A Decision Tree for "*buys_computer*"



buys_computer="yes" or buys_computer="no"

Source: Han & Kamber (2006)

Three possibilities for partitioning tuples based on the splitting Criterion



Source: Han & Kamber (2006)

Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Source: Han & Kamber (2006)

Attribute Selection Measure

- Notation: Let *D, the data partition, be a training set of* class-labeled tuples.
  Suppose the class label attribute has *m distinct values defining* *m* distinct classes, C_i *(for i = 1, … , m).*
  Let $C_{i,D}$ *be the set of tuples of class* C_i *in D.*
  Let |D| and *|* $C_{i,D}$ *| denote the number of tuples in D and* $C_{i,D}$ *, respectively.*
- *Example:*
  - *Class: buys_computer= "yes" or "no"*
  - *Two distinct classes (m=2)*
    - *Class* C_i *(i=1,2):*
      C_1 *= "yes",*
      C_2 *= "no"*

Source: Han & Kamber (2006)

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

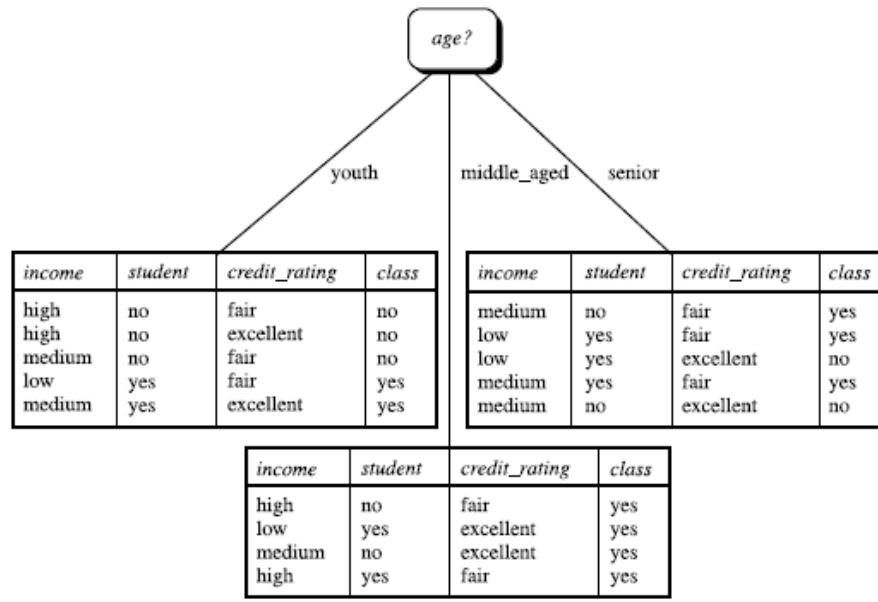
Source: Han & Kamber (2006)

$\log_2(1) = 0$
$\log_2(2) = 1$
$\log_2(3) = 1.5850$
$\log_2(4) = 2$
$\log_2(5) = 2.3219$
$\log_2(6) = 2.5850$
$\log_2(7) = 2.8074$
$\log_2(8) = 3$
$\log_2(9) = 3.1699$
$\log_2(10) = 3.3219$

$\log_2(0.1) = -3.3219$
$\log_2(0.2) = -2.3219$
$\log_2(0.3) = -1.7370$
$\log_2(0.4) = -1.3219$
$\log_2(0.5) = -1$
$\log_2(0.6) = -0.7370$
$\log_2(0.7) = -0.5146$
$\log_2(0.8) = -0.3219$
$\log_2(0.9) = -0.1520$
$\log_2(1) = 0$

Class-labeled training tuples from the *AllElectronics customer database*

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no



The attribute age has the highest information gain and therefore becomes the splitting attribute at the root node of the decision tree

Source: Han & Kamber (2006)

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

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

Decision Tree
Information Gain

Customer database

ID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	middle_aged	high	no	fair	yes
3	youth	high	no	excellent	no
4	senior	medium	no	fair	yes
5	senior	high	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	excellent	yes

What is the class (buys_computer = “yes” or buys_computer = “no”) for a customer (age=youth, income=medium, student =yes, credit= fair)?

Customer database

ID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	middle_aged	high	no	fair	yes
3	youth	high	no	excellent	no
4	senior	medium	no	fair	yes
5	senior	high	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	excellent	yes
11	youth	medium	yes	fair	?

What is the class (buys_computer = “yes” or buys_computer = “no”) for a customer (age=youth, income=medium, student =yes, credit= fair)?

Yes = 0.0889

No = 0.0167

Table 1 shows the class-labeled training tuples from customer database. Please calculate and illustrate the final **decision tree** returned by decision tree induction using **information gain**.

(1) What is the Information Gain of “age”?

(2) What is the Information Gain of “income”?

(3) What is the Information Gain of “student”?

(4) What is the Information Gain of “credit_rating”?

(5) What is the class (buys_computer = “yes” or buys_computer = “no”) for a customer (age=youth, income=medium, student =yes, credit= fair) based on the classification result by decision three induction?

ID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	middle_aged	high	no	fair	yes
3	youth	high	no	excellent	no
4	senior	medium	no	fair	yes
5	senior	high	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	excellent	yes

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

$\log_2 (1) = 0$

$\log_2 (2) = 1$

$\log_2 (3) = 1.5850$

$\log_2 (4) = 2$

$\log_2 (5) = 2.3219$

$\log_2 (6) = 2.5850$

$\log_2 (7) = 2.8074$

$\log_2 (8) = 3$

$\log_2 (9) = 3.1699$

$\log_2 (10) = 3.3219$

$\log_2 (0.1) = -3.3219$

$\log_2 (0.2) = -2.3219$

$\log_2 (0.3) = -1.7370$

$\log_2 (0.4) = -1.3219$

$\log_2 (0.5) = -1$

$\log_2 (0.6) = -0.7370$

$\log_2 (0.7) = -0.5146$

$\log_2 (0.8) = -0.3219$

$\log_2 (0.9) = -0.1520$

$\log_2 (1) = 0$

ID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	middle_aged	high	no	fair	yes
3	youth	high	no	excellent	no
4	senior	medium	no	fair	yes
5	senior	high	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	excellent	yes

Class P (Positive): buys_computer = “yes”

Class N (Negative): buys_computer = “no”

$P(buys = yes) = P_{i=1} = P_1$ = 6/10 = 0.6

$P(buys = no) = P_{i=2} = P_2$ = 4/10 = 0.4

$\log_2(0.1) = -3.3219$
$\log_2(0.2) = -2.3219$
$\log_2(0.3) = -1.7370$
$\log_2(0.4) = -1.3219$
$\log_2(0.5) = -1$
$\log_2(0.6) = -0.7370$
$\log_2(0.7) = -0.5146$
$\log_2(0.8) = -0.3219$
$\log_2(0.9) = -0.1520$
$\log_2(1) = 0$

$\log_2(1) = 0$
$\log_2(2) = 1$
$\log_2(3) = 1.5850$
$\log_2(4) = 2$
$\log_2(5) = 2.3219$
$\log_2(6) = 2.5850$
$\log_2(7) = 2.8074$
$\log_2(8) = 3$
$\log_2(9) = 3.1699$
$\log_2(10) = 3.3219$

Step 1: Expected information

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$Info(D) = I(6,4) = -\frac{6}{10}\log_2(\frac{6}{10}) + (-\frac{4}{10}\log_2(\frac{4}{10}))$$

$$= -0.6 \times \log_2(0.6) - 0.4 \times \log_2(0.4)$$

$$= -0.6 \times (-0.737) - 0.4 \times (-1.3219)$$

$$= 0.4422 + 0.5288$$

$$= 0.971$$

Info(D) = I(6,4) = 0.971

ID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	middle_aged	high	no	fair	yes
3	youth	high	no	excellent	no
4	senior	medium	no	fair	yes
5	senior	high	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	excellent	yes

age	p_i	n_i	*total*
youth	1	3	4
middle_aged	2	0	2
senior	3	1	4

income	p_i	n_i	*total*
high	2	2	4
medium	2	1	3
low	2	1	3

student	p_i	n_i	*total*
yes	4	1	5
no	2	3	5

credit_rating	p_i	n_i	*total*
excellent	2	2	4
fair	4	2	6

age	p_i	n_i	*total*	$I(p_i, n_i)$	$I(p_i, n_i)$
youth	1	3	4	$I(1,3)$	0.8112
middle_ aged	2	0	2	$I(2,0)$	0
senior	3	1	4	$I(3,1)$	0.8112

Step 2: Information
Step 3: Information Gain

$$I(1,3) = -\frac{1}{4}\log_2(\frac{1}{4}) + (-\frac{3}{4}\log_2(\frac{3}{4}))$$
$$= -0.25\times[\log_2 1 - \log_2 4] + (-0.75\times[\log_2 3 - \log_2 4])$$
$$= -0.25\times[0-2] - 0.75\times[1.585-2]$$
$$= -0.25\times[-2] - 0.75\times[-0.415]$$
$$= 0.5 + 0.3112 = 0.8112$$

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

Info(D) = I(6,4) = 0.971

$$Info_A(D) = \sum_{j=1}^{v}\frac{|D_j|}{|D|}\times I(D_j)$$

$$I(2,0) = -\frac{2}{2}\log_2(\frac{2}{2}) + (-\frac{0}{2}\log_2(\frac{0}{2}))$$
$$= -1\times\log_2 1 + (-0\times\log_2 0)$$
$$= -1\times 0 + (-0\times -\infty)$$
$$= 0 + 0 = 0$$

$$Info_{age}(D) = \frac{4}{10}I(1,3) + \frac{2}{10}I(2,0) + \frac{4}{10}I(3,1)$$
$$= \frac{4}{10}\times 0.8112 + \frac{2}{10}\times 0 + \frac{4}{10}\times 0.8112$$
$$= 0.3244 + 0 + 0.3244 = 0.6488$$

$$I(3,1) = -\frac{3}{4}\log_2(\frac{3}{4}) + (-\frac{1}{4}\log_2(\frac{1}{4}))$$
$$= -0.75\times[\log_2 3 - \log_2 4] + (-0.25\times[\log_2 1 - \log_2 4])$$
$$= -0.75\times[1.585-2] - 0.25\times[0-2]$$
$$= -0.75\times[-0.415] - 0.25\times[-2]$$
$$= 0.3112 + 0.5 = 0.8112$$

$$Gain(A) = Info(D) - Info_A(D)$$

$$Gain(age) = Info(D) - Info_{age}(D)$$
$$= 0.971 - 0.6488 = 0.3221$$

(1) Gain(age)= 0.3221

income	***p_i***	***n_i***	***total***	***$I(p_i, n_i)$***	***$I(p_i, n_i)$***
high	2	2	4	*I(2,2)*	*1*
medium	2	1	3	*I(2,1)*	*0.9182*
low	2	1	3	*I(2,1)*	*0.9182*

$$I(2,2) = -\frac{2}{4}\log_2(\frac{2}{4}) + (-\frac{2}{4}\log_2(\frac{2}{4}))$$
$$= -0.5\times[\log_2 2 - \log_2 4] + (-0.5\times[\log_2 2 - \log_2 4])$$
$$= -0.5\times[1-2] - 0.5\times[1-2]$$
$$= -0.5\times[-1] - 0.5\times[-1]$$
$$= 0.5 + 0.5 = 1$$

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$Info(D) = I(6,4) = 0.971$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

$$I(2,1) = -\frac{2}{3}\log_2(\frac{2}{3}) + (-\frac{1}{3}\log_2(\frac{1}{3}))$$
$$= -0.67\times[\log_2 2 - \log_2 3] + (-0.33\times[\log_2 1 - \log_2 3])$$
$$= -0.67\times[1-1.585] - 0.33\times[0-1.585]$$
$$= -0.67\times[-0.585] - 0.33\times[-1.585]$$
$$= 0.9182$$

$$Info_{income}(D) = \frac{4}{10}I(2,2) + \frac{3}{10}I(2,1) + \frac{3}{10}I(2,1)$$
$$= \frac{4}{10}\times 1 + \frac{3}{10}\times 0.9182 + \frac{3}{10}\times 0.9182$$
$$= 0.4 + 0.2755 + 0.2755 = 0.951$$

$$Gain(A) = Info(D) - Info_A(D)$$

$$Gain(income) = Info(D) - Info_{income}(D)$$
$$= 0.971 - 0.951 = 0.02$$

(2) Gain(income)= 0.02

student	p_i	n_i	*total*	$I(p_i, n_i)$	$I(p_i, n_i)$
yes	4	1	5	$I(4,1)$	0.7219
no	2	3	5	$I(2,3)$	0.971

$$I(4,1) = -\frac{4}{5}\log_2(\frac{4}{5}) + (-\frac{1}{5}\log_2(\frac{1}{5}))$$
$$= -0.8\times[\log_2 4 - \log_2 5] + (-0.2\times[\log_2 1 - \log_2 5)$$
$$= -0.8\times[2-2.3219] - 0.2\times[0-2.3219]$$
$$= -0.8\times[-0.3219] - 0.2\times[-2.3219]$$
$$= 0.25752 + 0.46438 = 0.7219$$

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

Info(D) = I(6,4) = 0.971

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|}\times I(D_j)$$

$$I(2,3) = -\frac{2}{5}\log_2(\frac{2}{5}) + (-\frac{3}{5}\log_2(\frac{3}{5}))$$
$$= -0.4\times[\log_2 0.4] + (-0.6\times[\log_2 0.6)$$
$$= -0.4\times[-1.3219] - 0.6\times[-0.737]$$
$$= 0.5288 + 0.4422 = 0.971$$

$$Info_{student}(D) = \frac{5}{10}I(4,1) + \frac{5}{10}I(2,3)$$
$$= 0.5\times 0.7219 + 0.5\times 0.971$$
$$= 0.36095 + 0.48545 = 0.8464$$

$$Gain(A) = Info(D) - Info_A(D)$$

$$Gain(student) = Info(D) - Info_{student}(D)$$
$$= 0.971 - 0.8464 = 0.1245$$

(3) Gain($_{student}$)= 0.1245

credit	p_i	n_i	total	$I(p_i, n_i)$	$I(p_i, n_i)$
excellent	2	2	4	$I(2,2)$	1
fair	4	2	6	$I(4,2)$	0.9183

$$I(2,2) = -\frac{2}{4}\log_2(\frac{2}{4}) + (-\frac{2}{4}\log_2(\frac{2}{4}))$$
$$= -0.5\times[\log_2 2 - \log_2 4] + (-0.5\times[\log_2 2 - \log_2 4])$$
$$= -0.5\times[1-2] - 0.5\times[1-2]$$
$$= -0.5\times[-1] - 0.5\times[-1]$$
$$= 0.5 + 0.5 = 1$$

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

Info(D) = I(6,4) = 0.971

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

$$I(4,2) = -\frac{4}{6}\log_2(\frac{4}{6}) + (-\frac{2}{6}\log_2(\frac{2}{6}))$$
$$= -0.67\times[\log_2 2 - \log_2 3] + (-0.33\times[\log_2 1 - \log_2 3])$$
$$= -0.67\times[1-1.585] - 0.33\times[0-1.585]$$
$$= -0.67\times[-0.585] - 0.33\times[-1.585]$$
$$= 0.9182$$

$$Info_{credit}(D) = \frac{4}{10}I(2,2) + \frac{6}{10}I(4,2)$$
$$= \frac{4}{10}\times 1 + \frac{6}{10}\times 0.9182$$
$$= 0.4 + 0.5509 = 0.9509$$

$$Gain(A) = Info(D) - Info_A(D)$$

$$Gain(credit) = Info(D) - Info_{credit}(D)$$
$$= 0.971 - 0.9509 = 0.019$$

(4) Gain($_{credit}$)= 0.019

What is the class (buys_computer = "yes" or buys_computer = "no") for a customer (age=youth, income=medium, student =yes, credit= fair)?

age	p_i	n_i	*total*
youth	**1**	**3**	**4**
middle_aged	2	0	2
senior	3	1	4

student	p_i	n_i	*total*
yes	**4**	**1**	**5**
no	2	3	5

income	p_i	n_i	*total*
high	2	2	4
midium	2	1	3
low	2	1	3

credit_rating	p_i	n_i	*total*
excellent	2	2	4
fair	4	2	6

(5) What is the class (buys_computer = “yes” or buys_computer = “no”) for a customer (age=youth, income=medium, student =yes, credit= fair) based on the classification result by decision three induction?

(5) Yes =0.0889 (No=0.0167)

age (0.3221) > student (0.1245) > income (0.02) > credit (0.019)

buys_computer = “yes”

age:youth (1/4) x student:yes (4/5) x income:medium (2/3) x credit:fair (4/6)

Yes: 1/4 x 4/5 x 2/3 x 4/6 = 4/45 = 0.0889

buys_computer = “no”

age:youth (3/4) x student:yes (1/5) x income:medium (1/3) x credit:fair (2/6)

No: 3/4 x 1/5 x 1/3 x 2/6 = 0.01667

What is the class (buys_computer = “yes” or buys_computer = “no”) for a customer (age=youth, income=medium, student =yes, credit= fair)?

Yes = 0.0889

No = 0.0167

Customer database

ID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	middle_aged	high	no	fair	yes
3	youth	high	no	excellent	no
4	senior	medium	no	fair	yes
5	senior	high	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	excellent	yes

Customer database

ID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	middle_aged	high	no	fair	yes
3	youth	high	no	excellent	no
4	senior	medium	no	fair	yes
5	senior	high	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	excellent	yes
11	youth	medium	yes	fair	?

Customer database

ID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	middle_aged	high	no	fair	yes
3	youth	high	no	excellent	no
4	senior	medium	no	fair	yes
5	senior	high	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	excellent	yes
11	youth	medium	yes	fair	Yes (0.0889)

Support Vector Machines
(SVM)

SVM—Support Vector Machines

- A new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors)

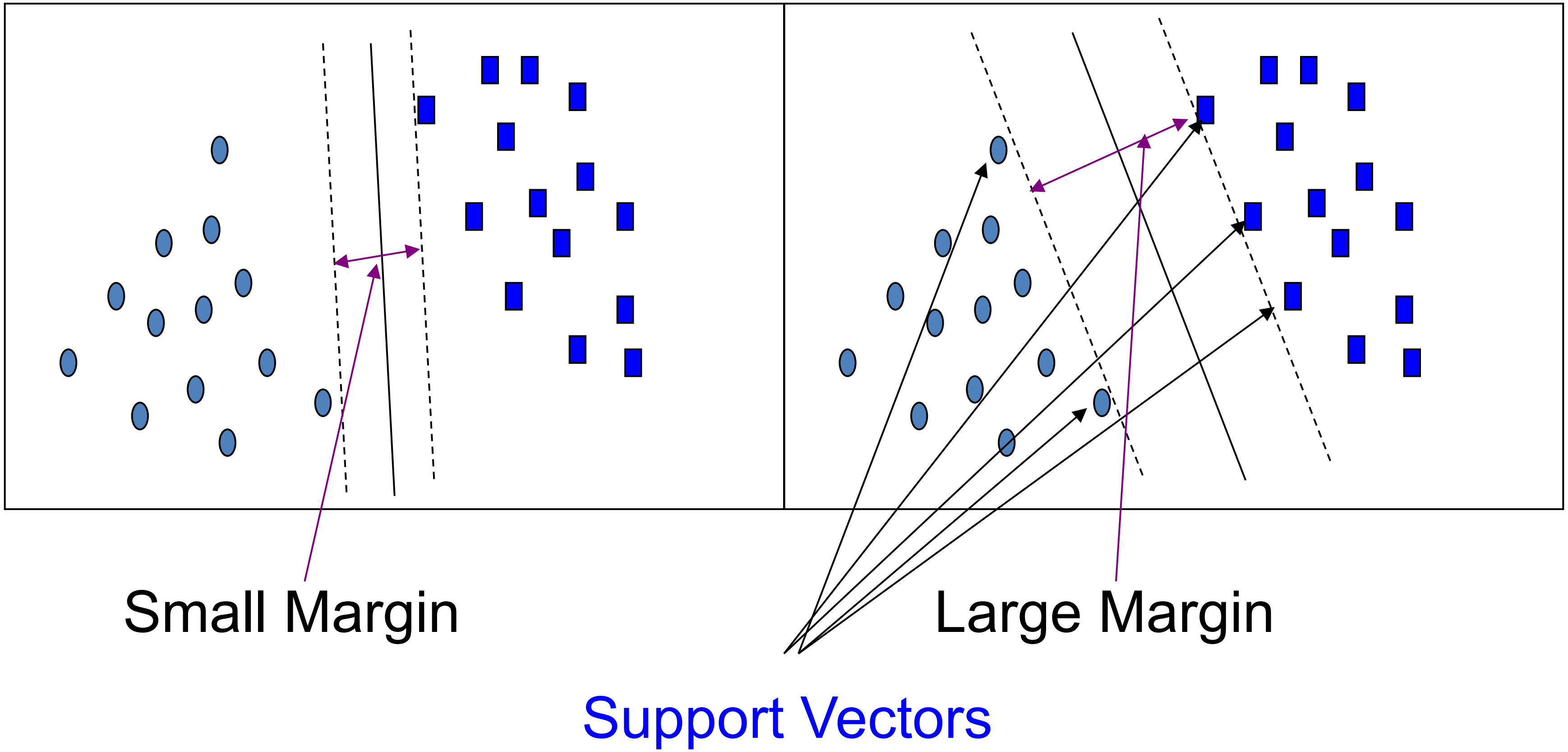
Source: Han & Kamber (2006)

SVM—History and Applications

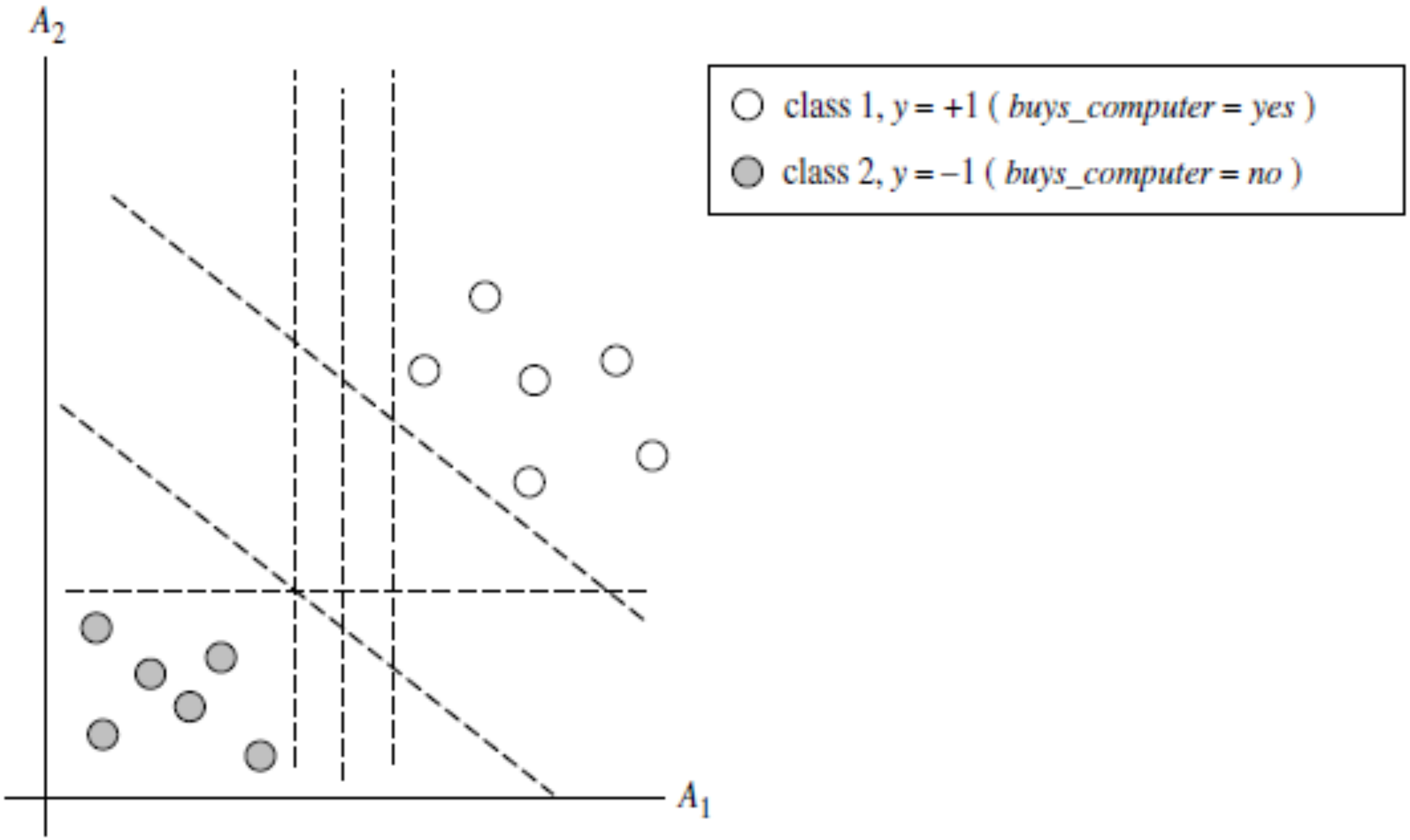
- Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis' statistical learning theory in 1960s
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used both for classification and prediction
- Applications:
  - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests, document classification

Source: Han & Kamber (2006)

SVM—General Philosophy



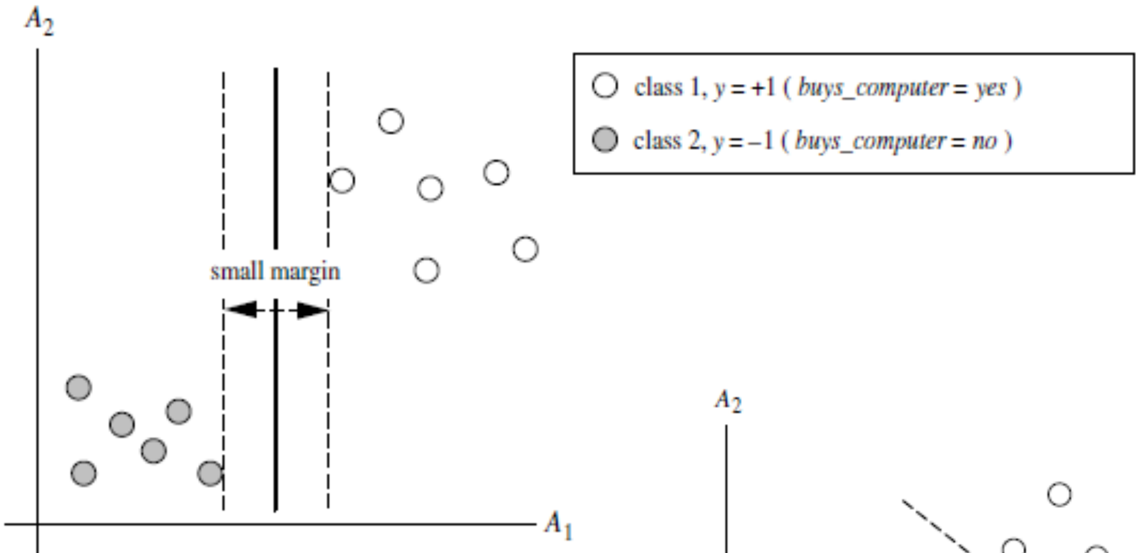
Classification (SVM)

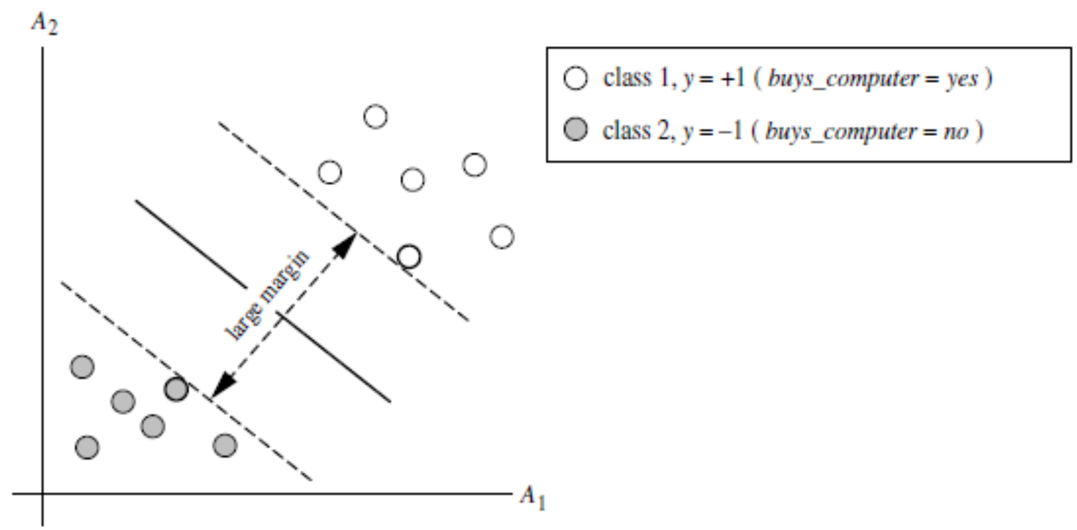


The 2-D training data are linearly separable. There are an infinite number of (possible) separating hyperplanes or “decision boundaries.”Which one is best?

Source: Han & Kamber (2006)

Classification (SVM)

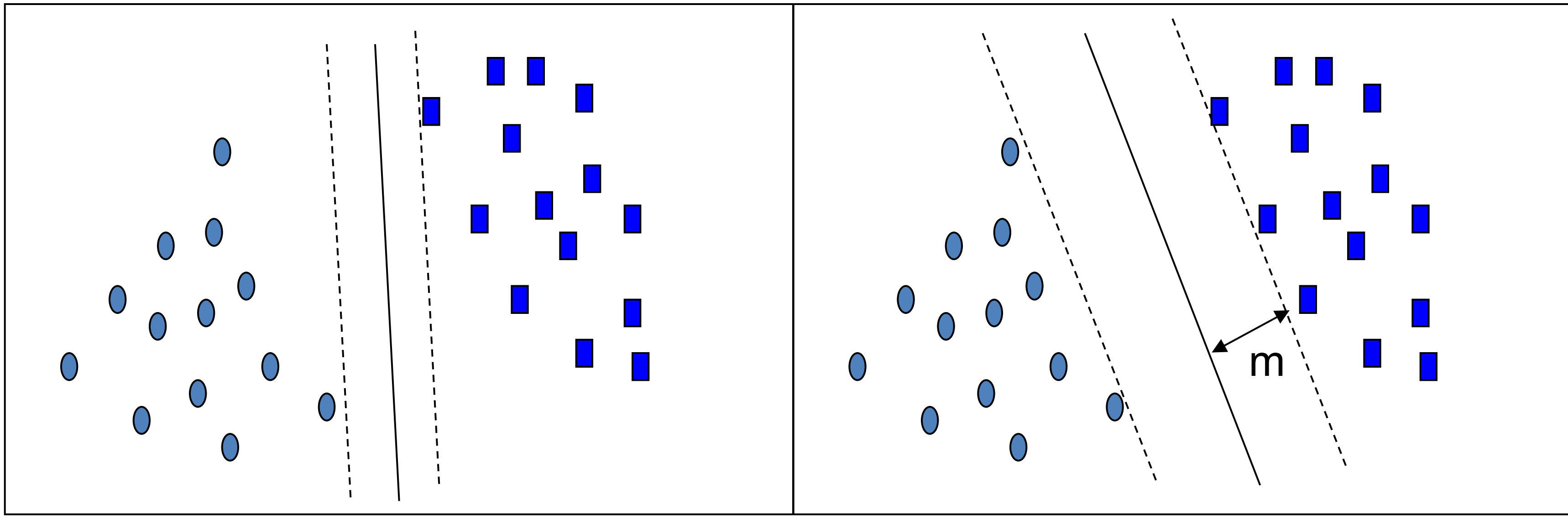




Which one is better? The one with the larger margin should have greater generalization accuracy.

Source: Han & Kamber (2006)

SVM—When Data Is Linearly Separable



Let data D be ($\mathbf{X}_1$, y_1), …, ($\mathbf{X}_{|D|}$, $y_{|D|}$), where $\mathbf{X}_i$ is the set of training tuples associated with the class labels y_i

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)

SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)

Source: Han & Kamber (2006)

SVM—Linearly Separable

- A separating hyperplane can be written as

  $$\mathbf{W} \bullet \mathbf{X} + b = 0$$

  where **W**=$\{w_1, w_2, \ldots, w_n\}$ is a weight vector and b a scalar (bias)
- For 2-D it can be written as

  $$w_0 + w_1 x_1 + w_2 x_2 = 0$$

- The hyperplane defining the sides of the margin:

  H_1: $w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

  H_2: $w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**
- This becomes a **constrained (convex) quadratic optimization** problem: Quadratic objective function and linear constraints → *Quadratic Programming (QP)* → Lagrangian multipliers

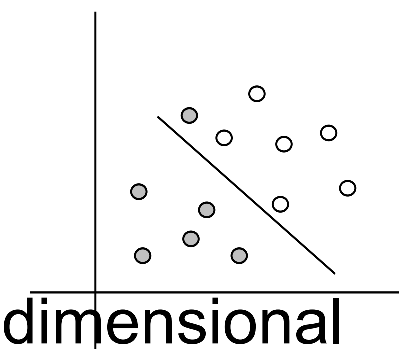
Source: Han & Kamber (2006)

Why Is SVM Effective on High Dimensional Data?

- The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The support vectors are the essential or critical training examples — they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

Source: Han & Kamber (2006)

SVM—Linearly Inseparable



- Transform the original input data into a higher dimensional space

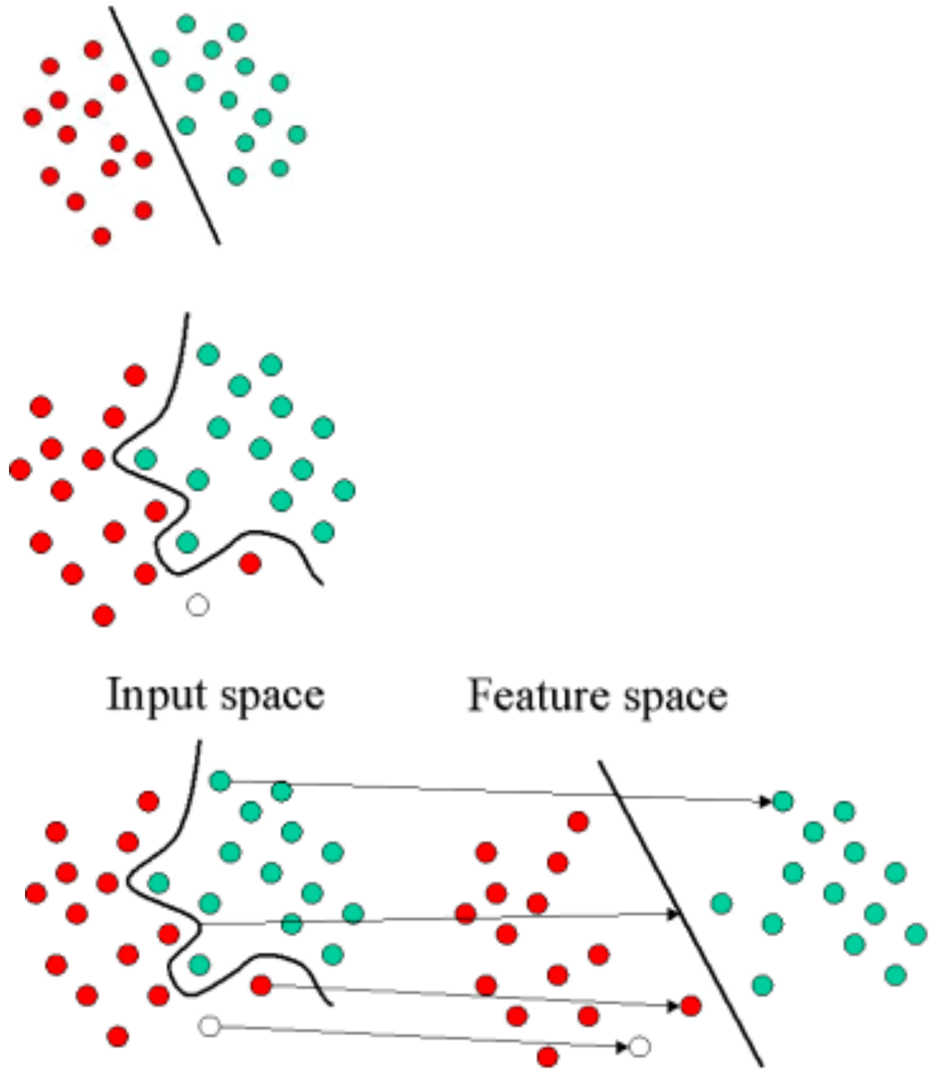
Example 6.8 Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector $\mathbf{X} = (x_1, x_2, x_3)$ is mapped into a 6D space Z using the mappings $\phi_1(X) = x_1, \phi_2(X) = x_2, \phi_3(X) = x_3$, $\phi_4(X) = (x_1)^2, \phi_5(X) = x_1x_2$, and $\phi_6(X) = x_1x_3$. A decision hyperplane in the new space is $d(\mathbf{Z})= \mathbf{WZ} + b$, where $\mathbf{W}$ and $\mathbf{Z}$ are vectors. This is linear. We solve for $\mathbf{W}$ and b and then substitute back so that we see that the linear decision hyperplane in the new $(\mathbf{Z})$ space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$\begin{aligned} d(Z) &= w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b \\ &= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b \end{aligned} \quad \blacksquare$$

- Search for a linear separating hyperplane in the new space

Source: Han & Kamber (2006)

Mapping Input Space to Feature Space



Source: http://www.statsoft.com/textbook/support-vector-machines/

SVM—Kernel functions

- Instead of computing the dot product on the transformed data tuples, it is mathematically equivalent to instead applying a kernel function $K(\mathbf{X}_i, \mathbf{X}_j)$ to the original data, i.e., $K(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i)\ \Phi(\mathbf{X}_j)$
- Typical Kernel Functions

$$\text{Polynomial kernel of degree } h: \quad K(X_i, X_j) = (X_i \cdot X_j + 1)^h$$

$$\text{Gaussian radial basis function kernel}: \quad K(X_i, X_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$$

$$\text{Sigmoid kernel}: \quad K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$$

- SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional user parameters)

Source: Han & Kamber (2006)

Evaluation

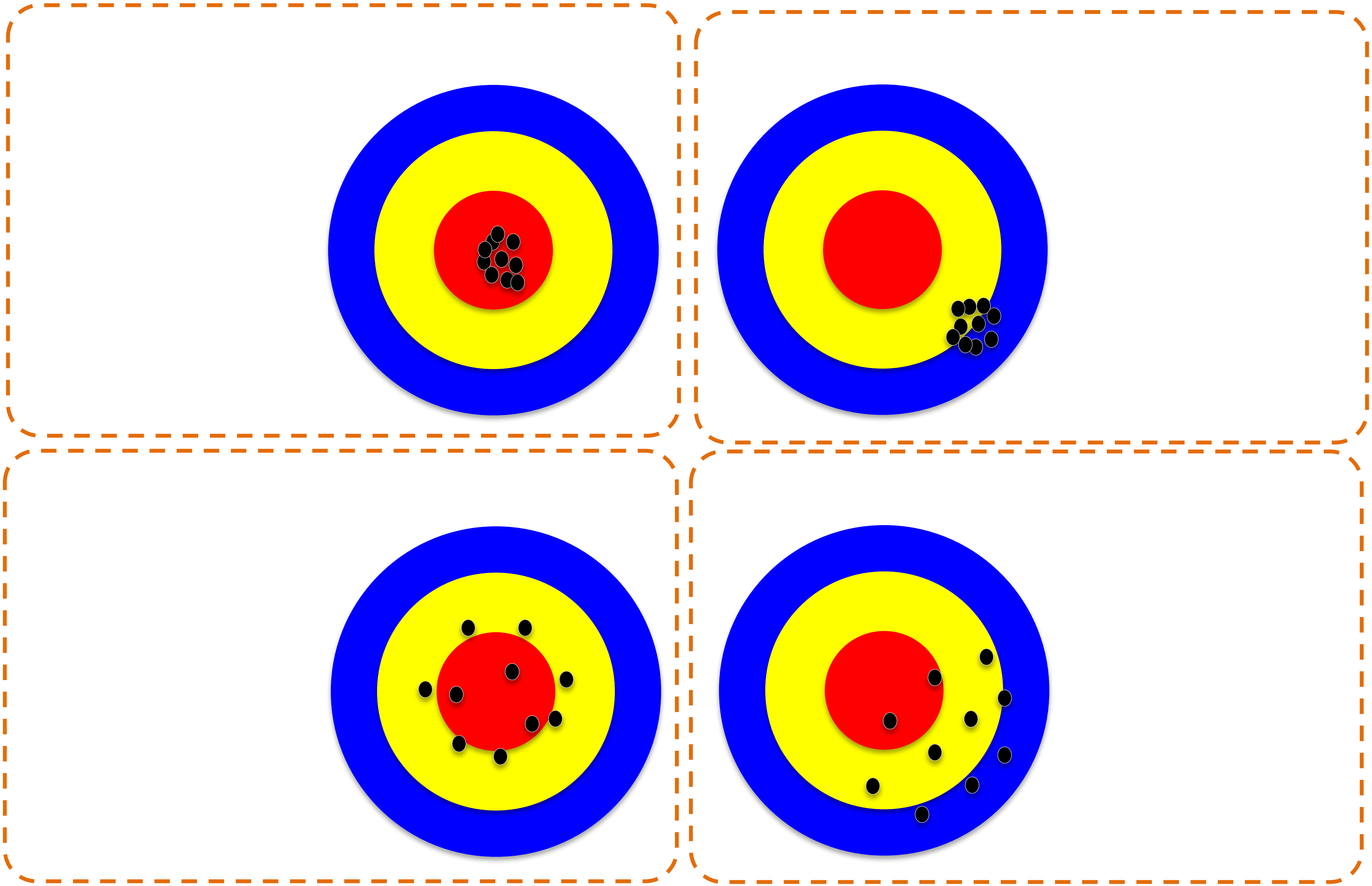
(Accuracy of Classification Model)

Assessing the Classification Model

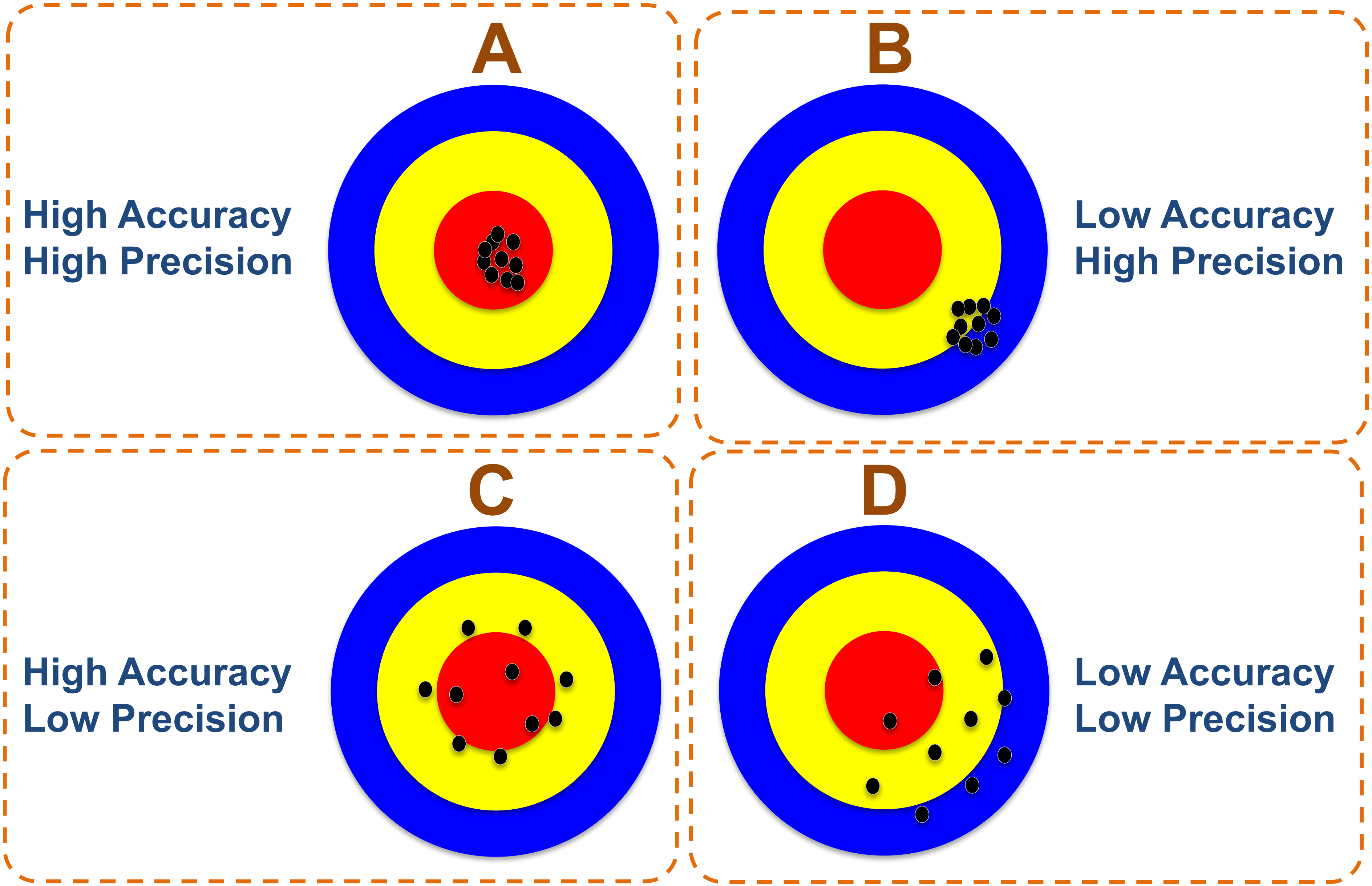
- Predictive accuracy
  - Hit rate
- Speed
  - Model building; predicting
- Robustness
- Scalability
- Interpretability
  - Transparency, explainability

Accuracy Validity

Precision Reliability



Accuracy vs. Precision



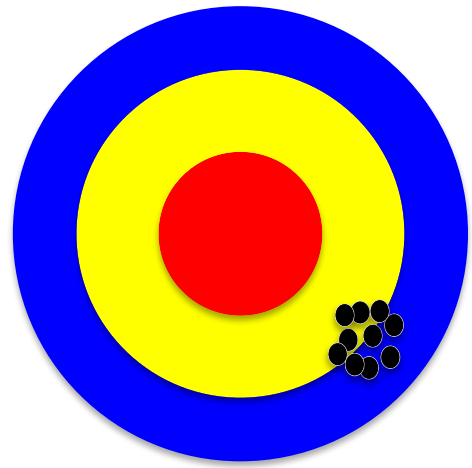
Accuracy vs. Precision

A

High Accuracy
High Precision

High Validity
High Reliability

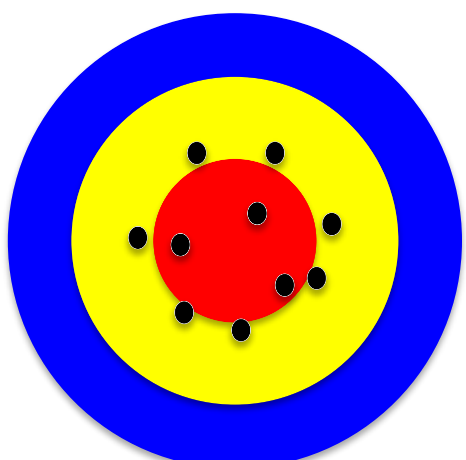
B



Low Accuracy
High Precision

Low Validity
High Reliability

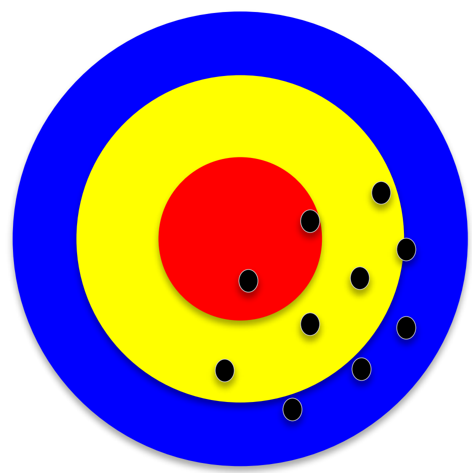
C



High Accuracy
Low Precision

High Validity
Low Reliability

D



Low Accuracy
Low Precision

Low Validity
Low Reliability

Accuracy vs. Precision

A

High Accuracy
High Precision

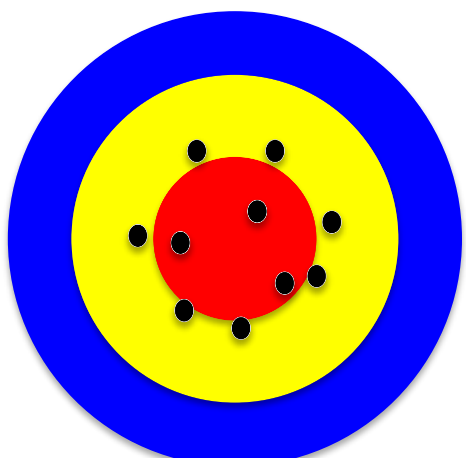
High Validity
High Reliability

B

Low Accuracy
High Precision

Low Validity
High Reliability

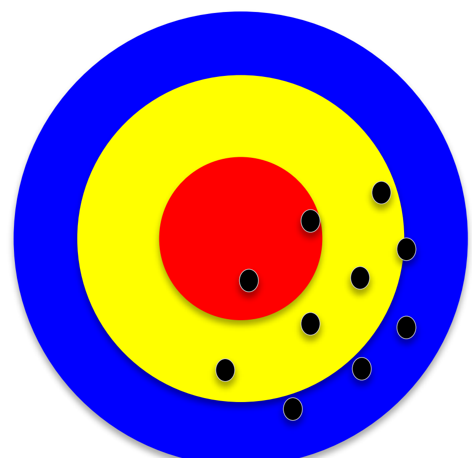
C



High Accuracy
Low Precision

High Validity
Low Reliability

D



Low Accuracy
Low Precision

Low Validity
Low Reliability

Confusion Matrix for Tabulation of Two-Class Classification Results

		True/Observed Class	
		Positive	**Negative**
Predicted Class	**Positive**	True Positive Count (TP)	False Positive Count (FP)
	Negative	False Negative Count (FN)	True Negative Count (TN)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

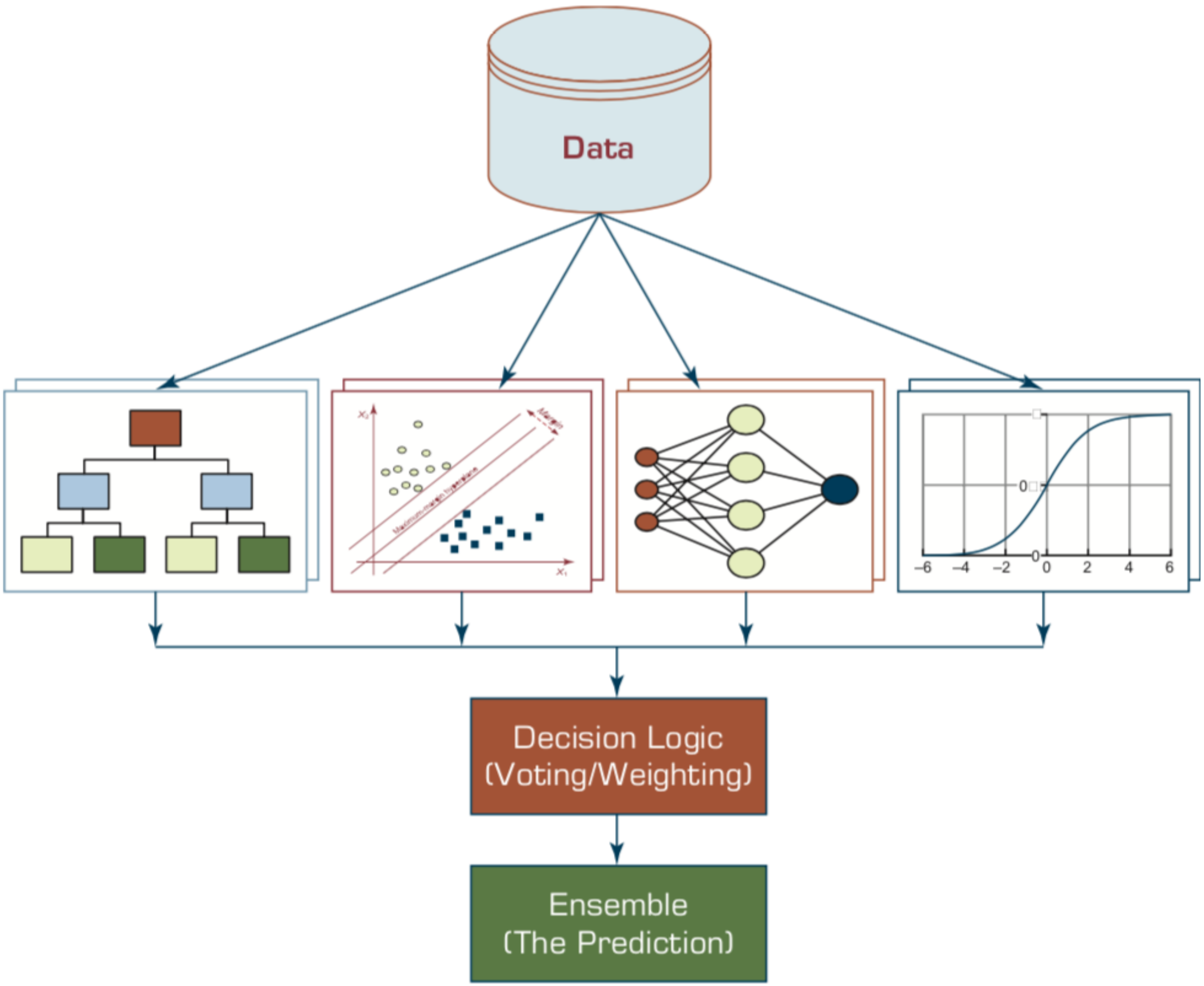
$$True\ Positive\ Rate = \frac{TP}{TP + FN}$$

$$True\ Negative\ Rate = \frac{TN}{TN + FP}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Ensemble Models
Heterogeneous Ensemble



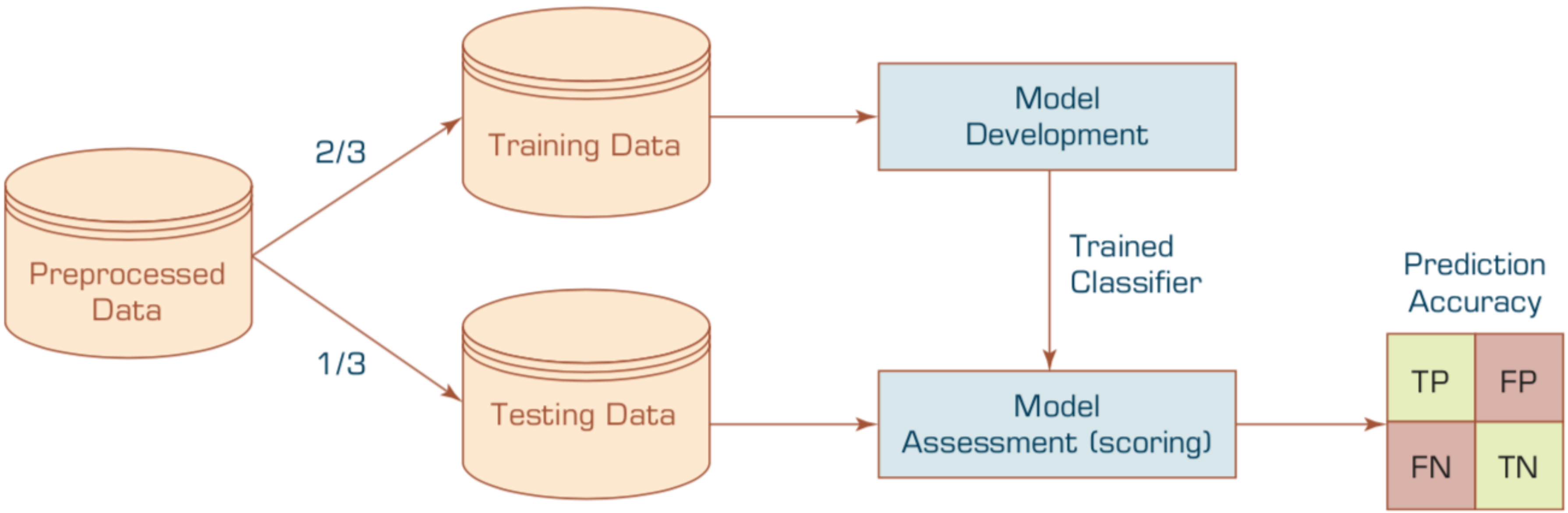
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Sensitivity =True Positive Rate

Specificity =True Negative Rate

Estimation Methodologies for Classification

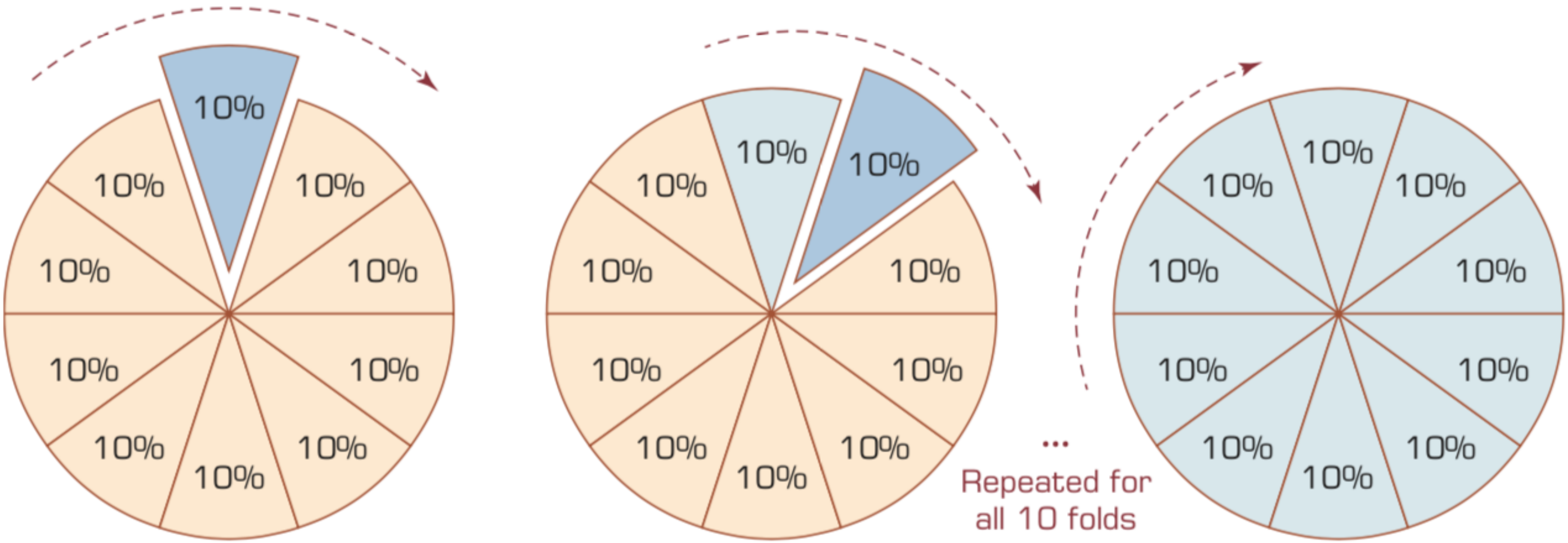
- Simple split (or holdout or test sample estimation)
  - Split the data into 2 mutually exclusive sets training (~70%) and testing (30%)



  - For ANN, the data is split into three sub-sets (training [~60%], validation [~20%], testing [~20%])

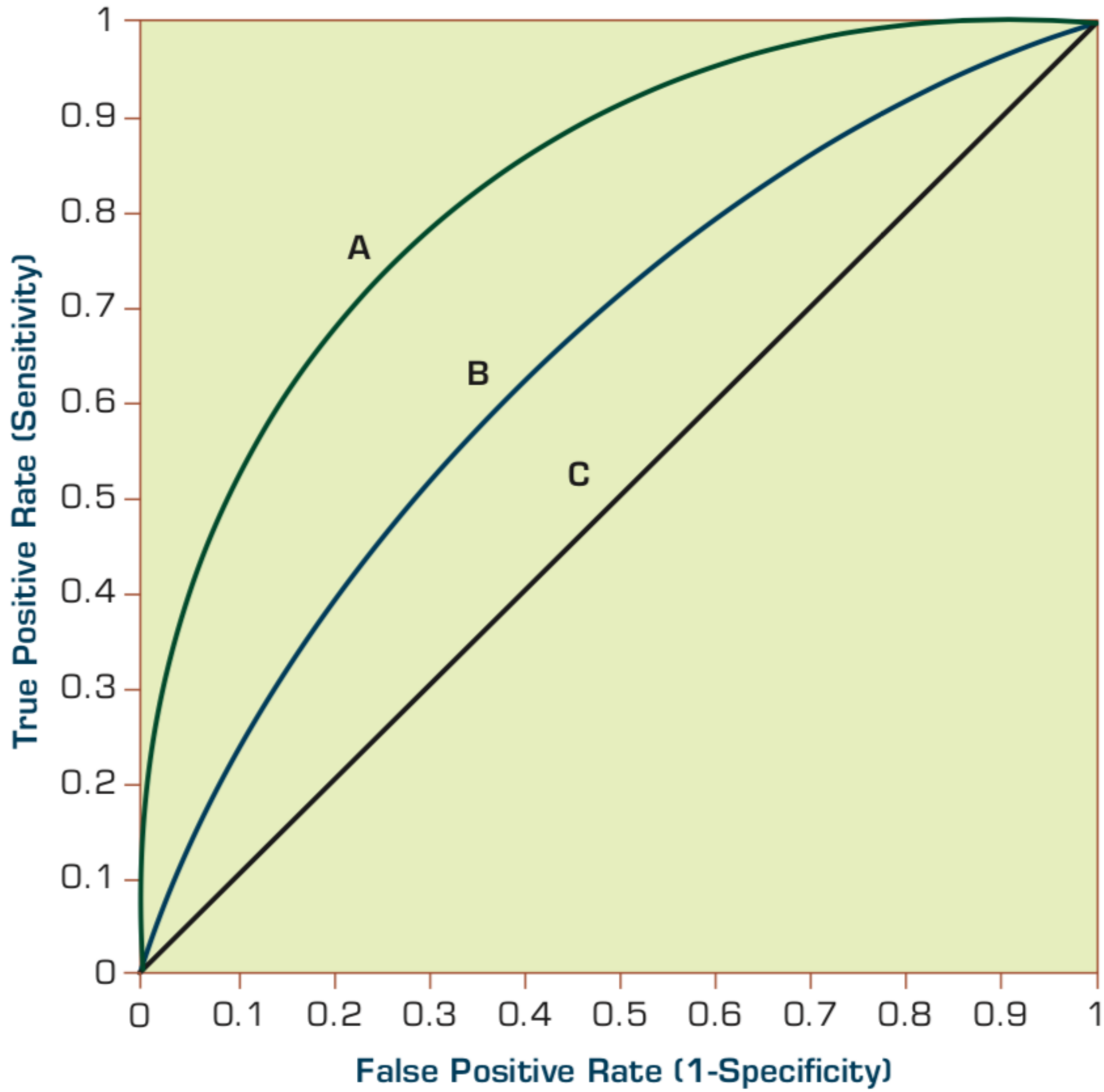
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

k-Fold Cross-Validation



Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

Estimation Methodologies for Classification
Area under the ROC curve



Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

		True Class (actual value)		
		Positive	Negative	total
Predictive Class (prediction outcome)	Positive	True Positive (TP)	False Positive (FP)	P’
	Negative	False Negative (FN)	True Negative (TN)	N’
total		P	N	

$$True\ Positive\ Rate\ (\text{Sensitivity}) = \frac{TP}{TP + FN}$$

$$True\ Negative\ Rate\ (\text{Specificity}) = \frac{TN}{TN + FP}$$

$$False\ Positive\ Rate = \frac{FP}{FP + TN}$$

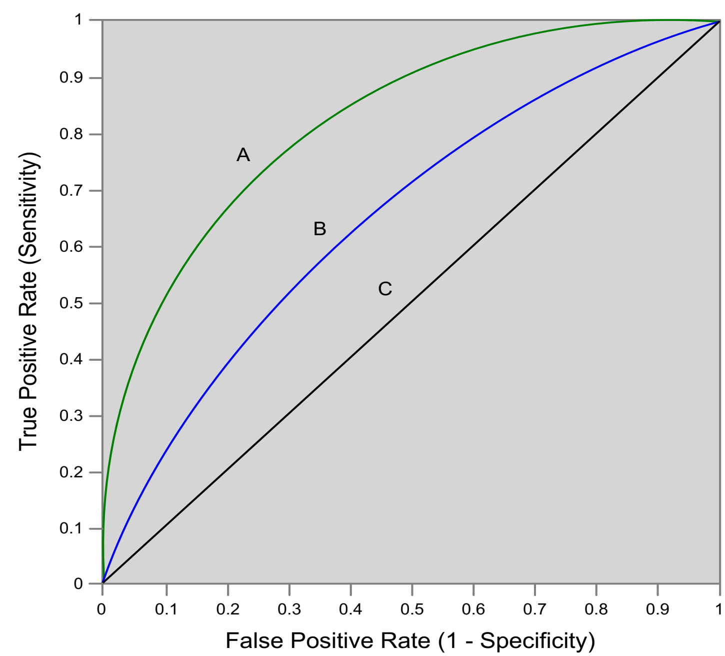
$$False\ Positive\ Rate\ (1\text{-Specificity}) = \frac{FP}{FP + TN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

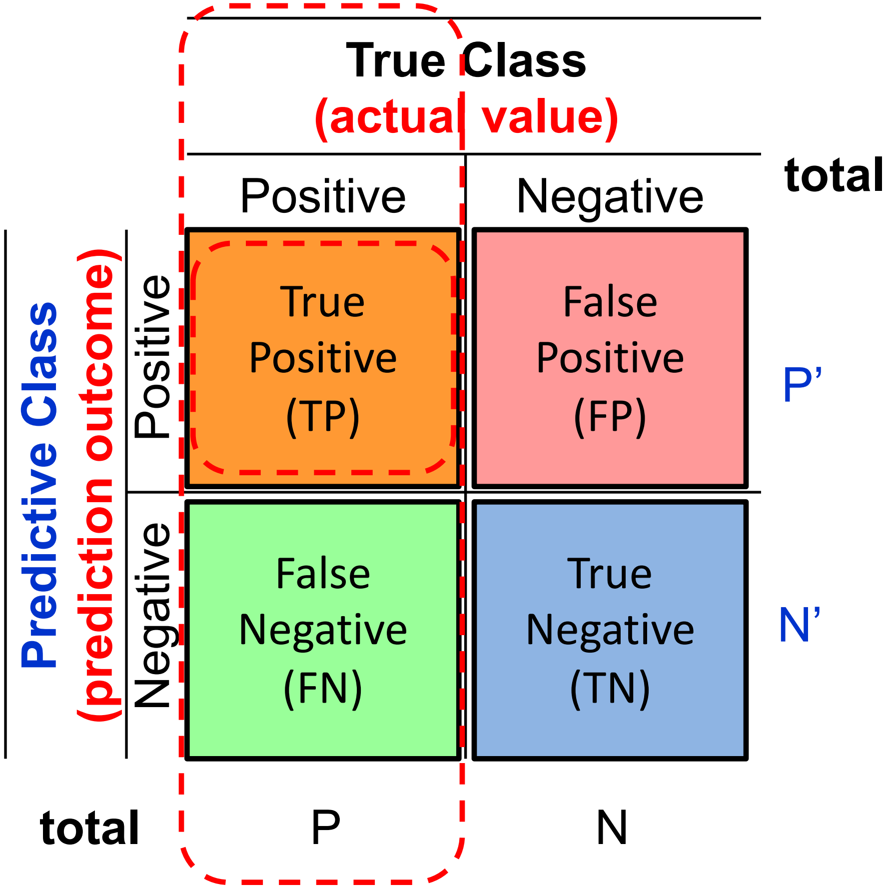
$$True\ Positive\ Rate = \frac{TP}{TP + FN}$$

$$True\ Negative\ Rate = \frac{TN}{TN + FP}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$



Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic



$$True\ Positive\ Rate = \frac{TP}{TP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$True\ Positive\ Rate\ (\text{Sensitivity}) = \frac{TP}{TP + FN}$$

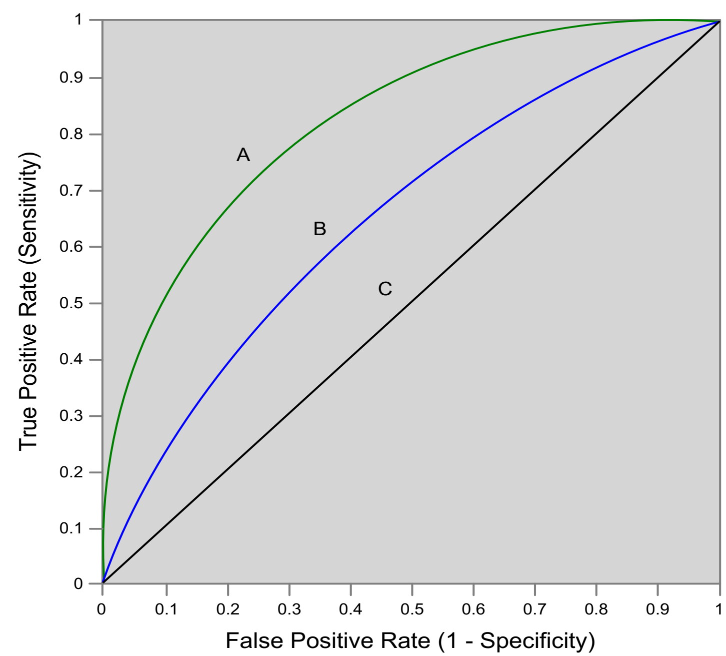
Sensitivity

= True Positive Rate

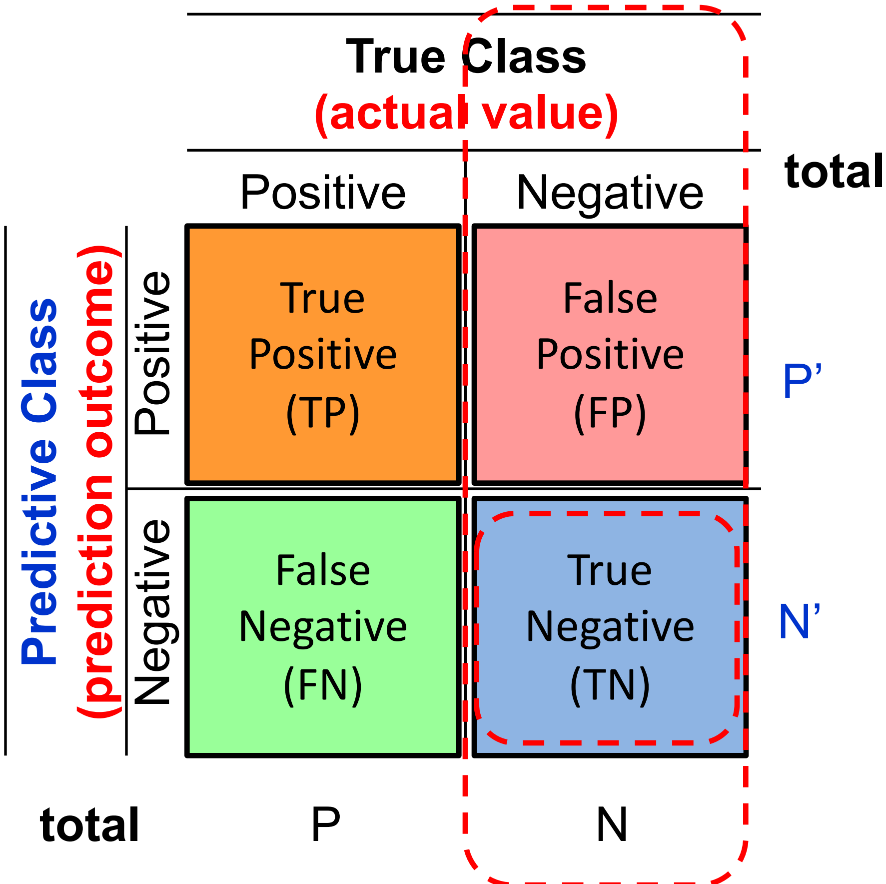
= Recall

= Hit rate

= TP / (TP + FN)



Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic



$$True\ Negative\ Rate = \frac{TN}{TN + FP}$$

Specificity

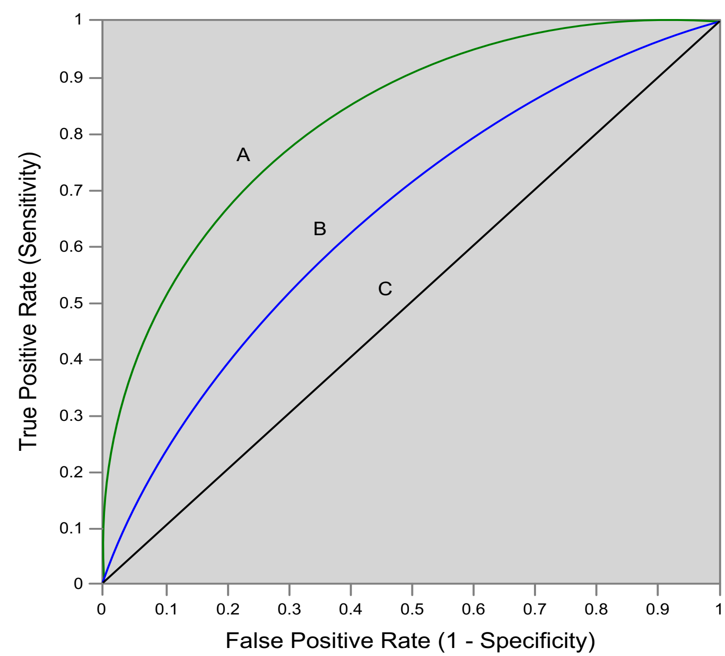
= True Negative Rate

= TN / N

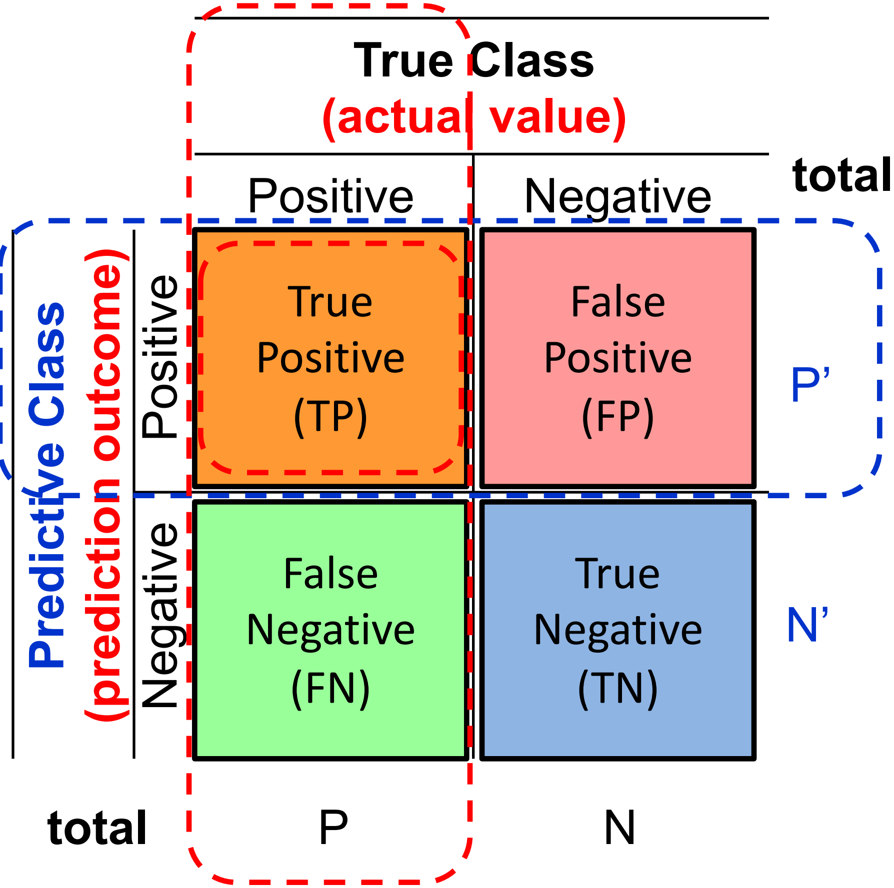
= TN / (TN+ FP)

$$True\ Negative\ Rate\ (\text{Specificity}) = \frac{TN}{TN + FP}$$

$$\text{False}\ Positive\ Rate\ (1\text{-Specificity}) = \frac{FP}{FP + TN}$$



Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic



Precision

= Positive Predictive Value (PPV)

$$Precision = \frac{TP}{TP + FP}$$

Recall

= True Positive Rate (TPR)

= Sensitivity

= Hit Rate

$$Recall = \frac{TP}{TP + FN}$$

F1 score (F-score)(F-measure)

is the harmonic mean of precision and recall

= 2TP / (P + P')

= 2TP / (2TP + FP + FN)

$$F = 2 * \frac{precision * recall}{precision + recall}$$

Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

A

63 (TP)	28 (FP)	91
37 (FN)	72 (TN)	109
100	100	200

Recall
= True Positive Rate (TPR)
= Sensitivity
= Hit Rate
= TP / (TP + FN)

Specificity
= True Negative Rate
= TN / N
= TN / (TN + FP)

TPR = 0.63

$$Recall = \frac{TP}{TP + FN}$$

$$True\ Negative\ Rate\ (\text{Specificity}) = \frac{TN}{TN + FP}$$

FPR = 0.28

$$\text{False}\ Positive\ Rate\ (1\text{-Specificity}) = \frac{FP}{FP + TN}$$

PPV = 0.69
=63/(63+28)
=63/91

$$Precision = \frac{TP}{TP + FP}$$

Precision
= Positive Predictive Value (PPV)

F1 = 0.66
= 2*(0.63*0.69)/(0.63+0.69)
= (2 * 63) /(100 + 91)
= (0.63 + 0.69) / 2 =1.32 / 2 =0.66

$$F = 2 * \frac{precision * recall}{precision + recall}$$

F1 score (F-score) (F-measure)
is the harmonic mean of precision and recall
= 2TP / (P + P')
= 2TP / (2TP + FP + FN)

ACC = 0.68
= (63 + 72) / 200
= 135/200 = 67.5

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

A

63 (TP)	28 (FP)	91
37 (FN)	72 (TN)	109
100	100	200

TPR = 0.63

FPR = 0.28

PPV = 0.69
=63/(63+28)
=63/91

F1 = 0.66
= 2*(0.63*0.69)/(0.63+0.69)
= (2 * 63) /(100 + 91)
= (0.63 + 0.69) / 2 =1.32 / 2 =0.66

ACC = 0.68
= (63 + 72) / 200
= 135/200 = 67.5

B

77 (TP)	77 (FP)	154
23 (FN)	23 (TN)	46
100	100	200

TPR = 0.77
FPR = 0.77
PPV = 0.50
F1 = 0.61
ACC = 0.50

Recall
= True Positive Rate (TPR)
= Sensitivity
= Hit Rate

$$Recall = \frac{TP}{TP + FN}$$

Precision
= Positive Predictive Value (PPV)

$$Precision = \frac{TP}{TP + FP}$$

Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

C

24 (TP)	88 (FP)	112
76 (FN)	12 (TN)	88
100	100	200

TPR = 0.24
FPR = 0.88
PPV = 0.21
F1 = 0.22
ACC = 0.18

C'

76 (TP)	12 (FP)	88
24 (FN)	88 (TN)	112
100	100	200

TPR = 0.76
FPR = 0.12
PPV = 0.86
F1 = 0.81
ACC = 0.82

Recall
= True Positive Rate (TPR)
= Sensitivity
= Hit Rate

$$Recall = \frac{TP}{TP + FN}$$

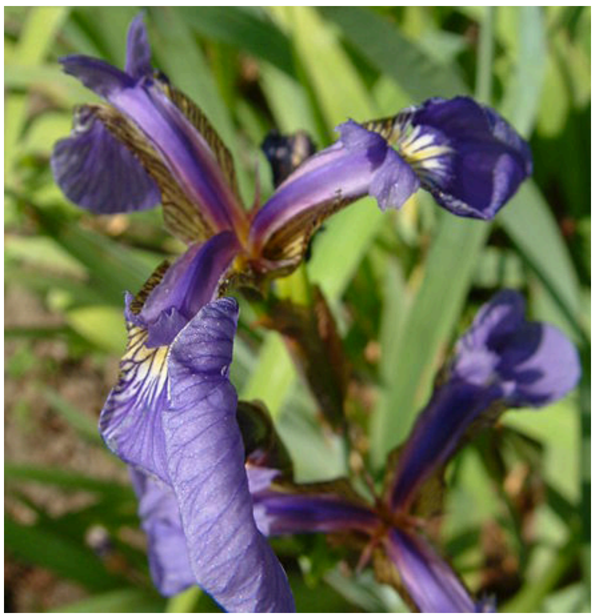
Precision
= Positive Predictive Value (PPV)

$$Precision = \frac{TP}{TP + FP}$$

Source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

Iris flower data set

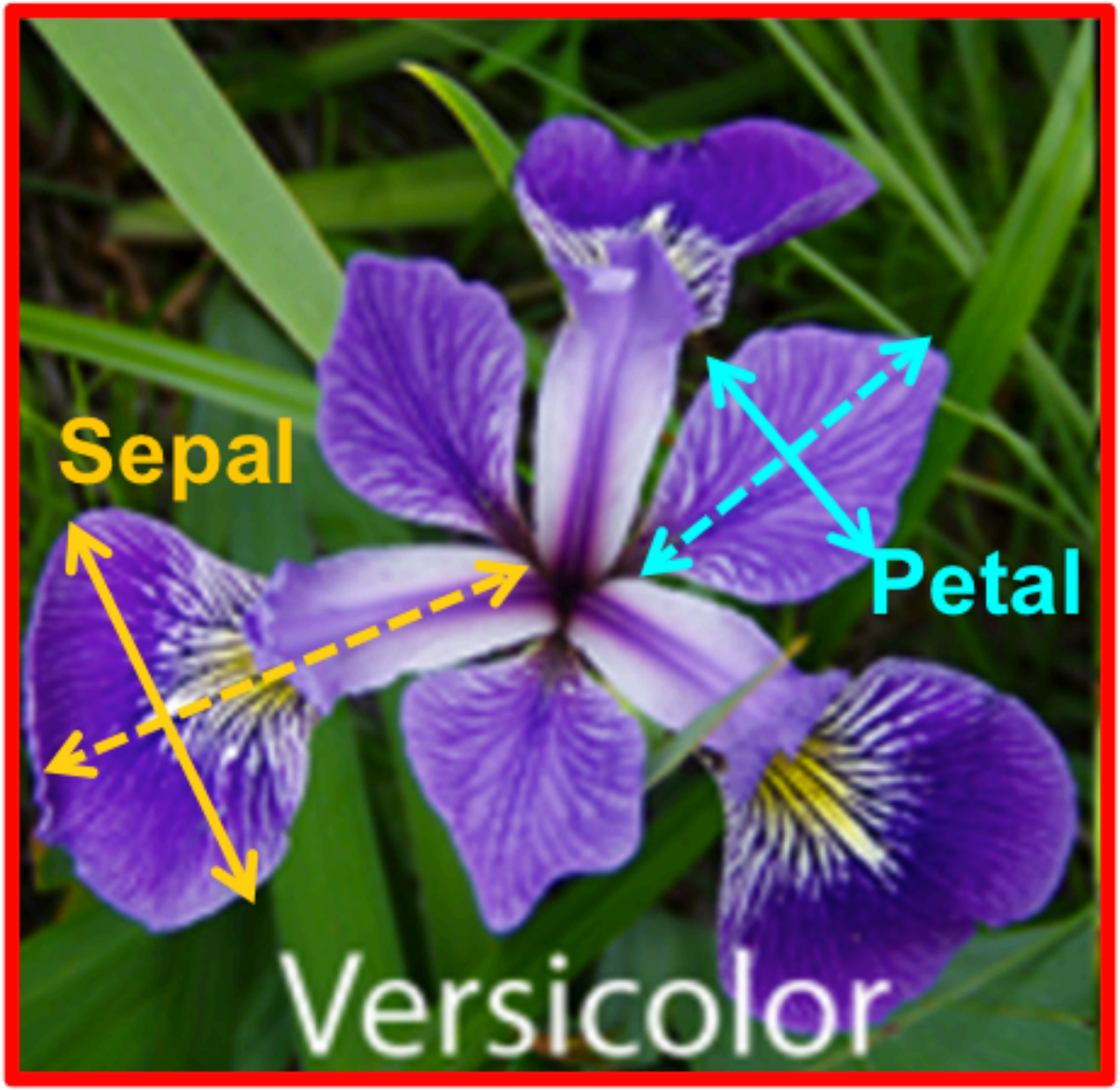
setosa **versicolor** **virginica**



Source: https://en.wikipedia.org/wiki/Iris_flower_data_set

Source: http://suruchifialoke.com/2016-10-13-machine-learning-tutorial-iris-classification/

Iris Classfication



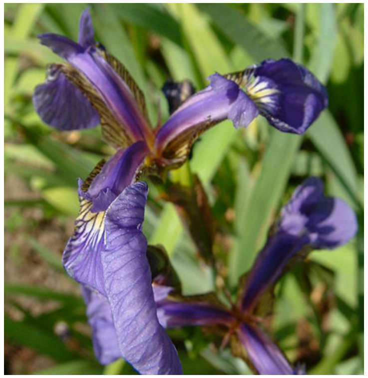
Source: http://suruchifialoke.com/2016-10-13-machine-learning-tutorial-iris-classification/

iris.data

https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5.0,3.0,1.6,0.2,Iris-setosa
```

setosa



virginica

versicolor

Machine Learning
Supervised Learning (Classification)
Learning from Examples

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
```

Machine Learning
Supervised Learning (Classification)
Learning from Examples

$$y = f(x)$$

x

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
```

y

Machine Learning
Supervised Learning (Classification)
Learning from Examples

$$y = f(x)$$

x	y

```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
```

```
# Import Libraries
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
print('imported')
```

```
imported
```

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)
print(df.head(10))
```

```
# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)
print(df.head(10))
```

	sepal-length	sepal-width	petal-length	petal-width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

df.tail(10)

```
print(df.tail(10))
```

	sepal-length	sepal-width	petal-length	petal-width	class
140	6.7	3.1	5.6	2.4	Iris-virginica
141	6.9	3.1	5.1	2.3	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica
143	6.8	3.2	5.9	2.3	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

df.describe()

```
print(df.describe())
```

```
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

print(df.info())
print(df.shape)

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal-length    150 non-null float64
sepal-width     150 non-null float64
petal-length    150 non-null float64
petal-width     150 non-null float64
class           150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
None
```

```
print(df.shape)
```

```
(150, 5)
```

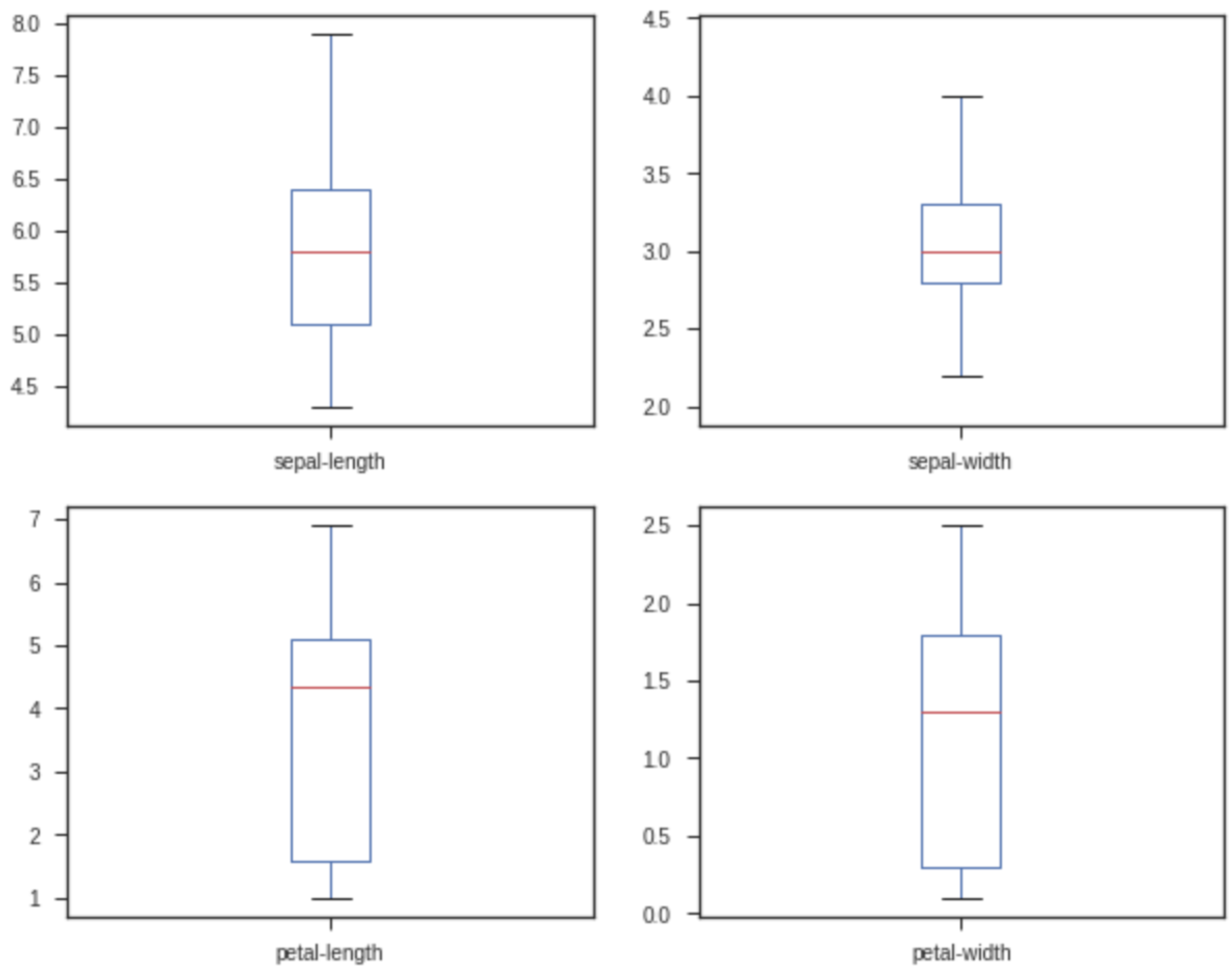
df.groupby('class').size()

```
print(df.groupby('class').size())
```

```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

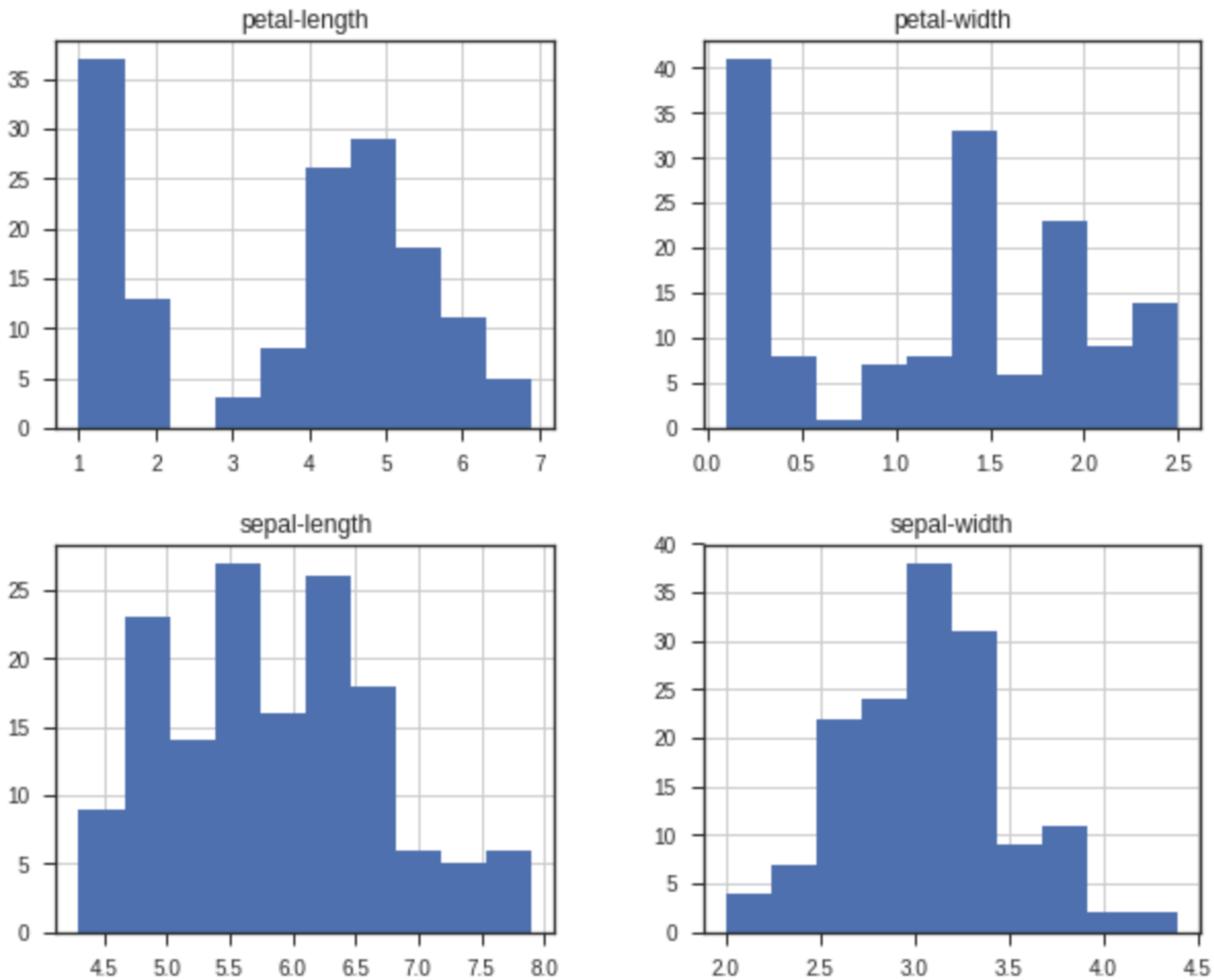
```
plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```


```
plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```

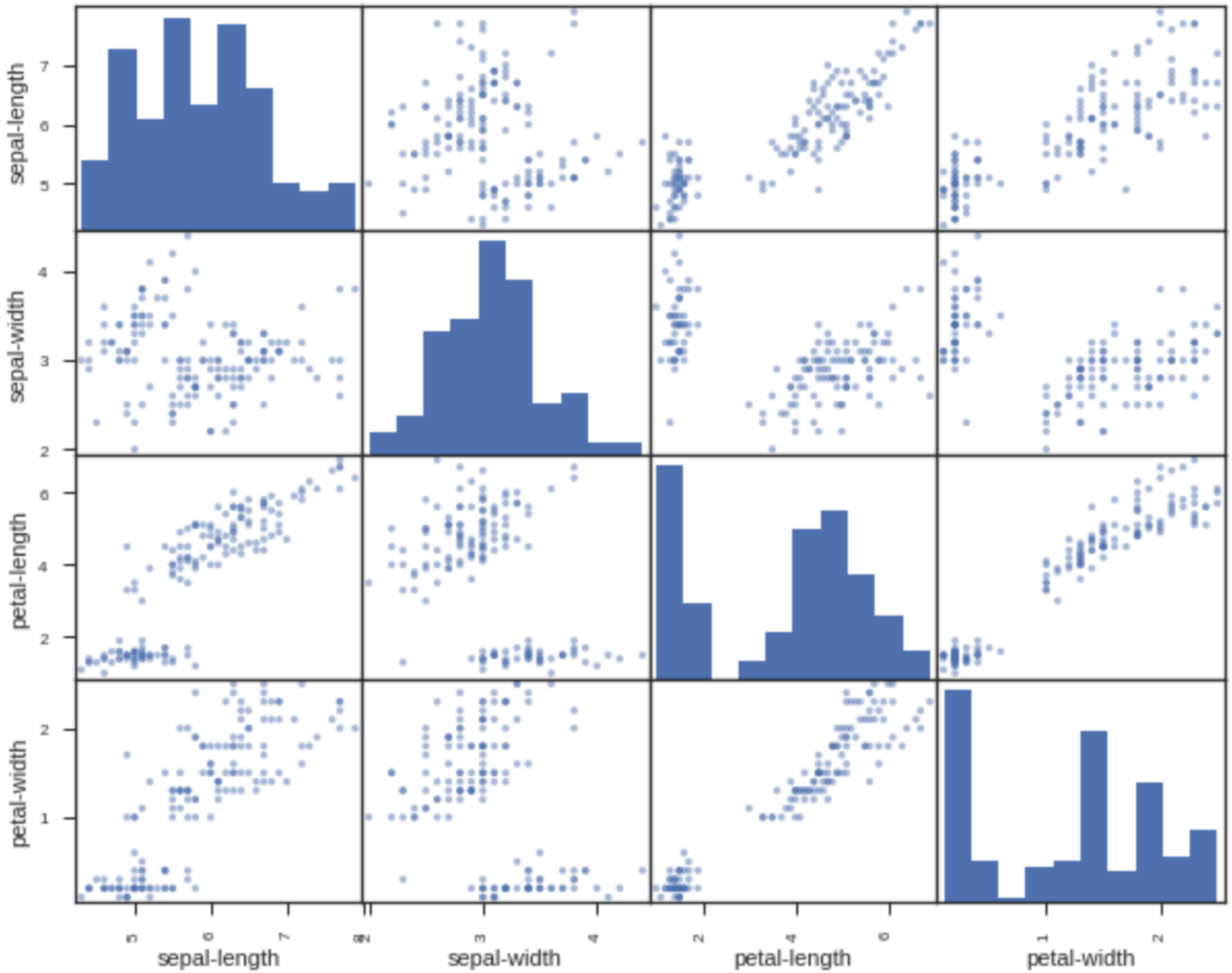
```
df.hist()
plt.show()
```

```
df.hist()
plt.show()
```



```
scatter_matrix(df)
plt.show()
```

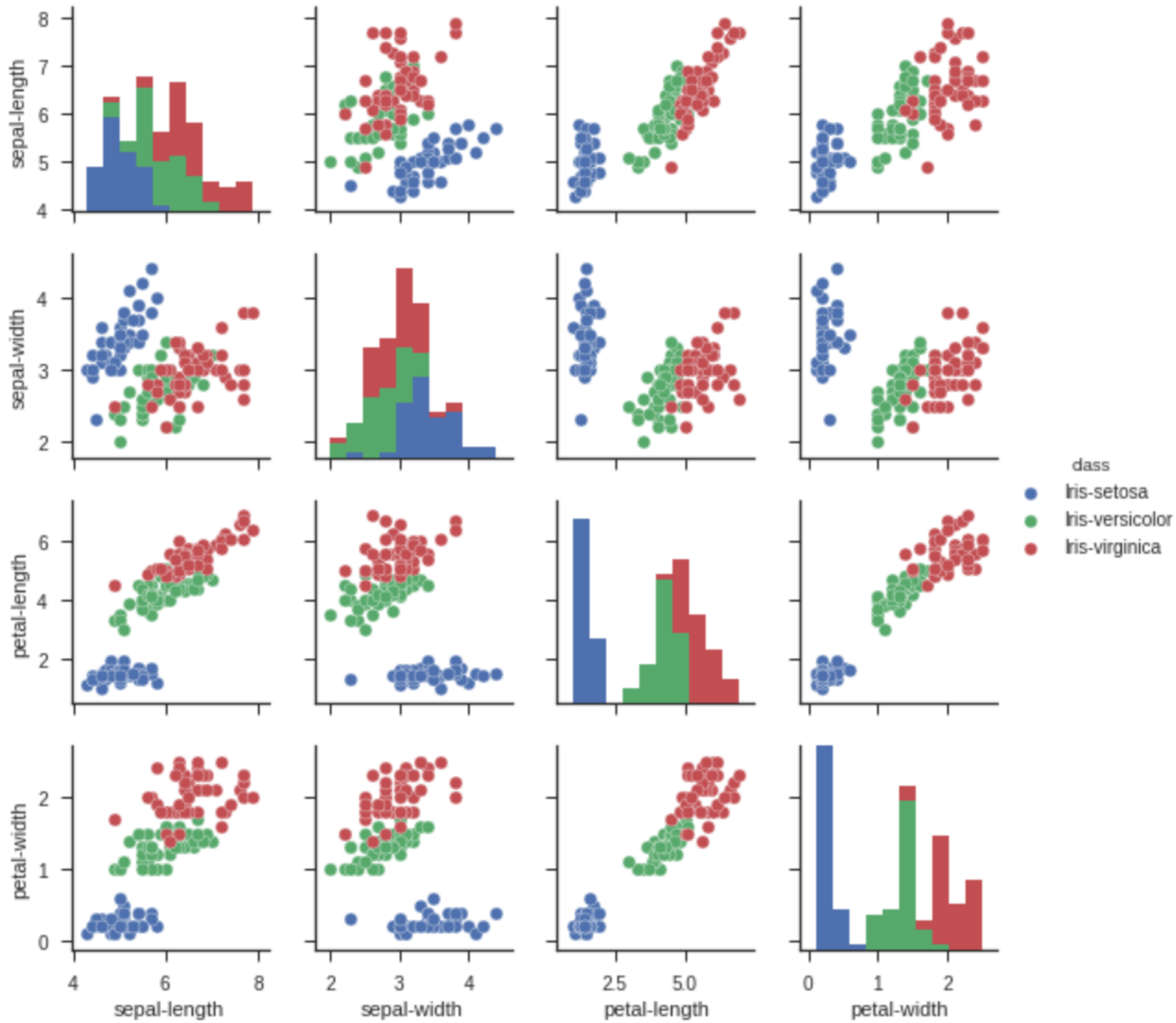

```
scatter_matrix(df)
plt.show()
```

sns.pairplot(df, hue="class", size=2)

```
sns.pairplot(df, hue="class", size=2)
```

```
<seaborn.axisgrid.PairGrid at 0x7f1d21267390>
```



Machine Learning

Supervised Learning

Classification

and

Prediction

Machine Learning: Supervised Learning Classification and Prediction

python101.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 10:43 AM

Comment Share A

Table of contents

+ Code + Text Connect Editing

Machine Learning with scikit-learn

Classification and Prediction

```
# Import libraries
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

# Import sklearn
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
print("Imported")

# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
```

https://tinyurl.com/aintpupython101

```
# Import sklearn
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
print("Imported")
```

```python
# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)

print(df.head(10))
print(df.tail(10))
print(df.describe())
print(df.info())
print(df.shape)
print(df.groupby('class').size())

plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()

df.hist()
plt.show()

scatter_matrix(df)
plt.show()

sns.pairplot(df, hue="class", size=2)
```

```
   sepal-length  sepal-width  petal-length  petal-width        class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
5           5.4          3.9           1.7          0.4  Iris-setosa
6           4.6          3.4           1.4          0.3  Iris-setosa
7           5.0          3.4           1.5          0.2  Iris-setosa
8           4.4          2.9           1.4          0.2  Iris-setosa
9           4.9          3.1           1.5          0.1  Iris-setosa
     sepal-length  sepal-width  petal-length  petal-width           class
140           6.7          3.1           5.6          2.4  Iris-virginica
141           6.9          3.1           5.1          2.3  Iris-virginica
142           5.8          2.7           5.1          1.9  Iris-virginica
```

```python
# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)

print(df.head(10))
print(df.tail(10))
print(df.describe())
print(df.info())
print(df.shape)
print(df.groupby('class').size())

plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()

df.hist()
plt.show()

scatter_matrix(df)
plt.show()

sns.pairplot(df, hue="class", size=2)
```

```
   sepal-length  sepal-width  petal-length  petal-width        class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
5           5.4          3.9           1.7          0.4  Iris-setosa
6           4.6          3.4           1.4          0.3  Iris-setosa
7           5.0          3.4           1.5          0.2  Iris-setosa
8           4.4          2.9           1.4          0.2  Iris-setosa
9           4.9          3.1           1.5          0.1  Iris-setosa
     sepal-length  sepal-width  petal-length  petal-width           class
140           6.7          3.1           5.6          2.4  Iris-virginica
141           6.9          3.1           5.1          2.3  Iris-virginica
142           5.8          2.7           5.1          1.9  Iris-virginica
```

https://tinyurl.com/aintpupython101

df.corr()

```
df.corr()
```

	sepal-length	sepal-width	petal-length	petal-width
sepal-length	1.000000	-0.109369	0.871754	0.817954
sepal-width	-0.109369	1.000000	-0.420516	-0.356544
petal-length	0.871754	-0.420516	1.000000	0.962757
petal-width	0.817954	-0.356544	0.962757	1.000000

https://tinyurl.com/aintpupython101

```
# Split-out validation dataset
array = df.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation =
model_selection.train_test_split(X, Y,
test_size=validation_size, random_state=seed)
scoring = 'accuracy'
```

```
# Split-out validation dataset
array = df.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=validation_size, random_state=seed)
scoring = 'accuracy'
```

```
len(Y_validation)
```

```
30
```

```
# Models
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA',
LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('DT',
DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
```

https://tinyurl.com/aintpupython101

```
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10,
random_state=seed)
    cv_results =
model_selection.cross_val_score(model,
X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %.4f (%.4f)" % (name,
cv_results.mean(), cv_results.std())
    print(msg)
```

https://tinyurl.com/aintpupython101

```python
# Models
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %.4f (%.4f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.9667 (0.0408)
LDA: 0.9750 (0.0382)
KNN: 0.9833 (0.0333)
DT: 0.9750 (0.0382)
NB: 0.9750 (0.0534)
SVM: 0.9917 (0.0250)
```

```
# Make predictions on validation dataset
model = KNeighborsClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation,
predictions))
print(confusion_matrix(Y_validation,
predictions))
print(classification_report(Y_validation,
predictions))
print(model)
```

```python
# Make predictions on validation dataset
model = KNeighborsClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```
0.9000
[[ 7  0  0]
 [ 0 11  1]
 [ 0  2  9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.85      0.92      0.88        12
 Iris-virginica       0.90      0.82      0.86        11

    avg / total       0.90      0.90      0.90        30

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

```
# Make predictions on validation dataset
model = SVC()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation,
predictions))
print(confusion_matrix(Y_validation,
predictions))
print(classification_report(Y_validation,
predictions))
print(model)
```

```python
model = SVC()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
```

```python
# Make predictions on validation dataset
model = SVC()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```
0.9333
[[ 7  0  0]
 [ 0 10  2]
 [ 0  0 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       1.00      0.83      0.91        12
 Iris-virginica       0.85      1.00      0.92        11

    avg / total       0.94      0.93      0.93        30

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

https://tinyurl.com/aintpupython101

```python
# Make predictions on validation dataset
model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```
0.9000
[[ 7  0  0]
 [ 0 11  1]
 [ 0  2  9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.85      0.92      0.88        12
 Iris-virginica       0.90      0.82      0.86        11

    avg / total       0.90      0.90      0.90        30

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

https://tinyurl.com/aintpupython101

```python
# Make predictions on validation dataset
model = GaussianNB()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```
0.8333
[[7 0 0]
 [0 9 3]
 [0 2 9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.82      0.75      0.78        12
 Iris-virginica       0.75      0.82      0.78        11

    avg / total       0.84      0.83      0.83        30

GaussianNB(priors=None)
```

https://tinyurl.com/aintpupython101

```python
# Make predictions on validation dataset
model = LogisticRegression()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```
0.8000
[[ 7  0  0]
 [ 0  7  5]
 [ 0  1 10]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.88      0.58      0.70        12
 Iris-virginica       0.67      0.91      0.77        11

    avg / total       0.83      0.80      0.80        30

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```python
# Make predictions on validation dataset
model = LinearDiscriminantAnalysis()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```
0.9667
[[ 7  0  0]
 [ 0 11  1]
 [ 0  0 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       1.00      0.92      0.96        12
 Iris-virginica       0.92      1.00      0.96        11

    avg / total       0.97      0.97      0.97        30

LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None,
              solver='svd', store_covariance=False, tol=0.0001)
```

https://tinyurl.com/aintpupython101

```python
# Make predictions on validation dataset
model = MLPClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
print("%.4f" % accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
print(model)
```

```
0.9000
[[ 7  0  0]
 [ 0  9  3]
 [ 0  0 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       1.00      0.75      0.86        12
 Iris-virginica       0.79      1.00      0.88        11

    avg / total       0.92      0.90      0.90        30

MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False)
```

Papers with Code
State-of-the-Art (SOTA)

Search for papers, code and tasks | Browse State-of-the-Art | Follow | Discuss | Trends | About | Log In/Register

Browse State-of-the-Art

1509 leaderboards • 1327 tasks • 1347 datasets • 17810 papers with code

Follow on Twitter for updates

Computer Vision

- **Semantic Segmentation** — 33 leaderboards, 667 papers with code
- **Image Classification** — 52 leaderboards, 564 papers with code
- **Object Detection** — 54 leaderboards, 467 papers with code
- **Image Generation** — 51 leaderboards, 231 papers with code
- **Pose Estimation** — 40 leaderboards, 231 papers with code

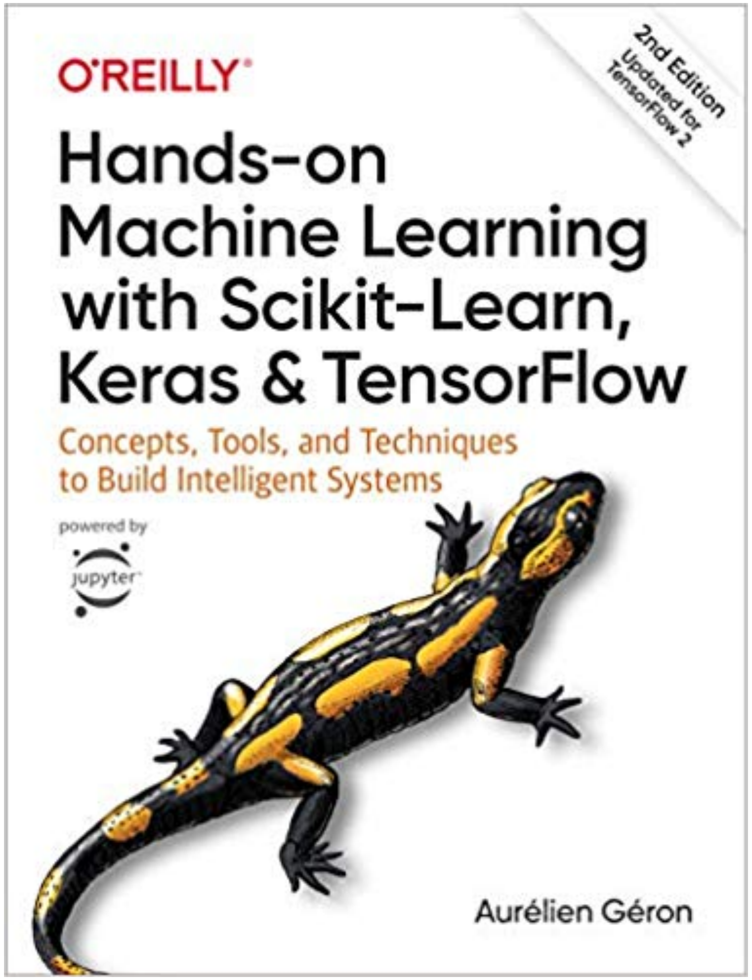
See all 707 tasks

Natural Language Processing

- Machine Translation
- Language Modelling
- Question Answering
- Sentiment Analysis
- Text Generation

https://paperswithcode.com/sota

Aurélien Géron (2019),
Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition
O'Reilly Media, 2019



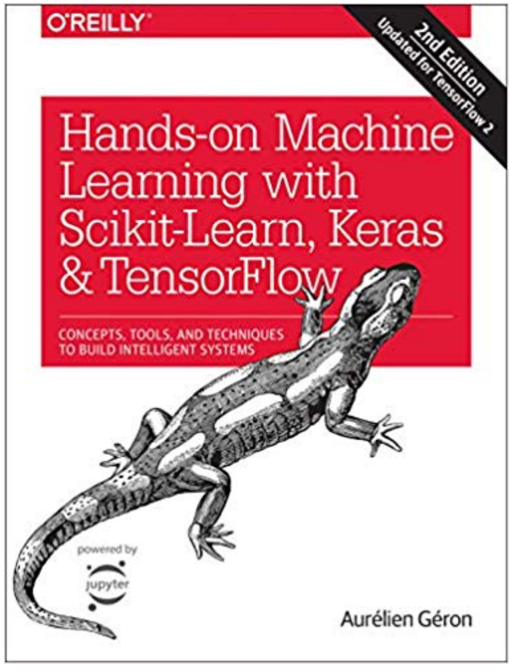
https://github.com/ageron/handson-ml2

Source: https://www.amazon.com/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1492032646/

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

Notebooks

1. The Machine Learning landscape
2. End-to-end Machine Learning project
3. Classification
4. Training Models
5. Support Vector Machines
6. Decision Trees
7. Ensemble Learning and Random Forests
8. Dimensionality Reduction
9. Unsupervised Learning Techniques
10. Artificial Neural Nets with Keras
11. Training Deep Neural Networks
12. Custom Models and Training with TensorFlow
13. Loading and Preprocessing Data
14. Deep Computer Vision Using Convolutional Neural Networks
15. Processing Sequences Using RNNs and CNNs
16. Natural Language Processing with RNNs and Attention
17. Representation Learning Using Autoencoders
18. Reinforcement Learning
19. Training and Deploying TensorFlow Models at Scale



https://github.com/ageron/handson-ml2

Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

python101.ipynb

File Edit View Insert Runtime Tools Help Last saved at 10:43 AM

Comment Share

+ Code + Text Connect Editing

Table of contents

Machine Learning with scikit-learn

Classification and Prediction

```python
# Import libraries
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

# Import sklearn
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
print("Imported")

# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
```

https://tinyurl.com/aintpupython101

Summary

- Supervised Learning
- Classification and Prediction
- Decision Tree (DT)
  - Information Gain (IG)
- Support Vector Machine (SVM)
- Data Mining Evaluation
  - Accuracy
  - Precision
  - Recall
  - F1 score (F-measure) (F-score)

References

- Jiawei Han and Micheline Kamber (2006), Data Mining: Concepts and Techniques, Second Edition, Elsevier, 2006.
- Jiawei Han, Micheline Kamber and Jian Pei (2011), Data Mining: Concepts and Techniques, Third Edition, Morgan Kaufmann 2011.
- Efraim Turban, Ramesh Sharda, Dursun Delen (2011), Decision Support and Business Intelligence Systems, Ninth Edition, Pearson.
- Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson.
- Jake VanderPlas (2016), Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media.
- Robert Layton (2017), Learning Data Mining with Python - Second Edition, Packt Publishing.
- Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media.
- Aurélien Géron (2019), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition, O'Reilly Media.
  https://github.com/wesm/pydata-book
- Min-Yuh Day (2021), Python 101, https://tinyurl.com/aintpupython101