7CS108: Data Science and Data Mining

Anirban Chakraborty

a.chakraborty@wlv.ac.uk

UNIVERSITY OF WOLVERHAMPTON

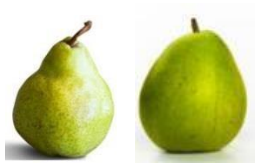
Classification

Classification is a technique or a function that classifies given data into one of the pre-defined categories/classes.

Apple

Pear

Distinguishing feature - colour



Classification

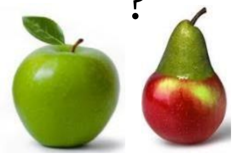
Classification is a technique or a function that classifies given data into one of the pre-defined categories/classes.

Apple

Pear

Distinguishing feature - colour

?

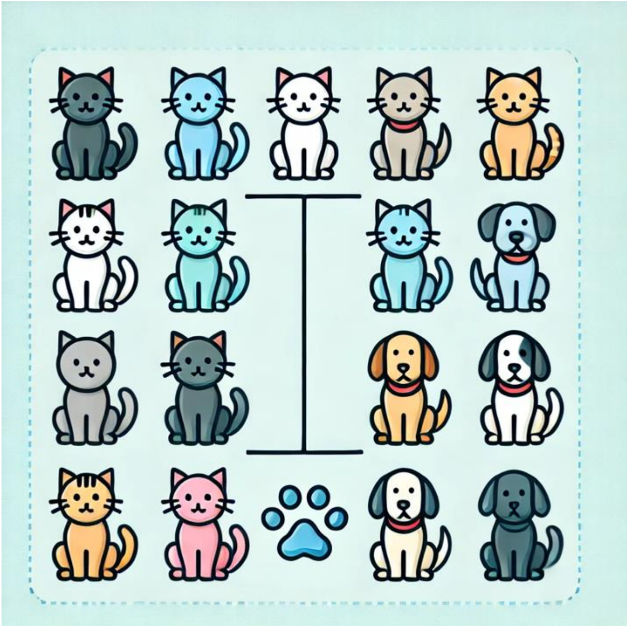


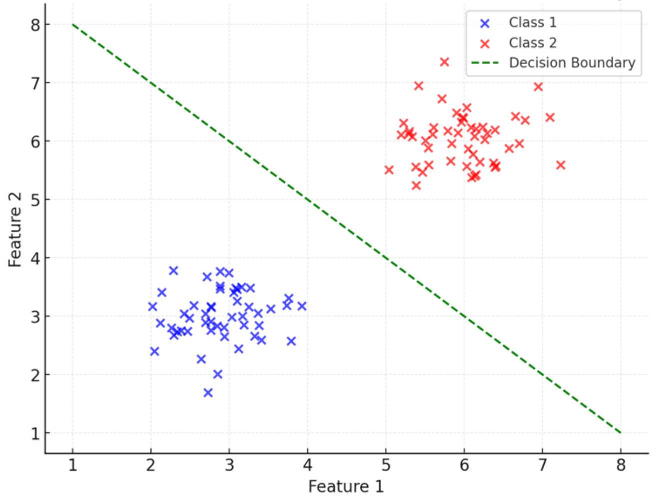
Is colour enough?

It may lead to misclassification!

Classification

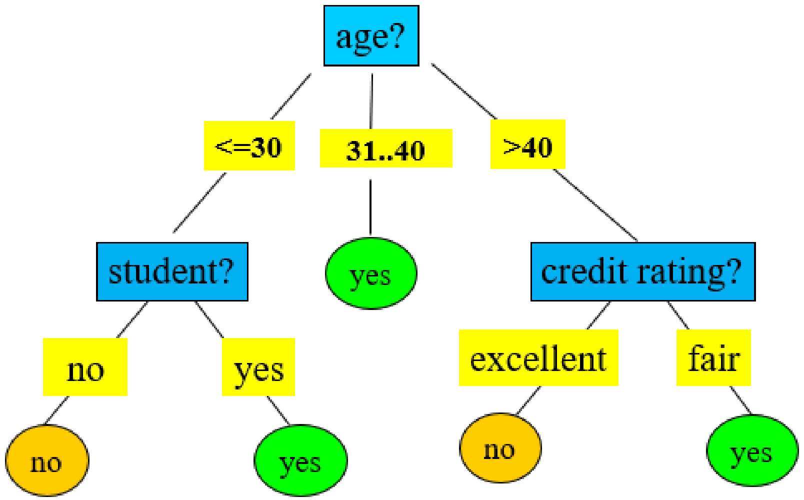
- Classification is a technique or a function that classifies given data into one of the pre-defined categories/classes.
- Classification is also known as "Supervised Learning"
  - There must be an "expert" (you) to "supervise" the computer.
    - In contrast, Clustering is known as "Unsupervised Learning". We will discuss it in the later lectures.
  - There are semi-supervised and reinforcement learning too.
- From the data mining point of view…
  - Classification ≈ Prediction ≈ Forecasting
  - This is because the techniques are the same





Question?

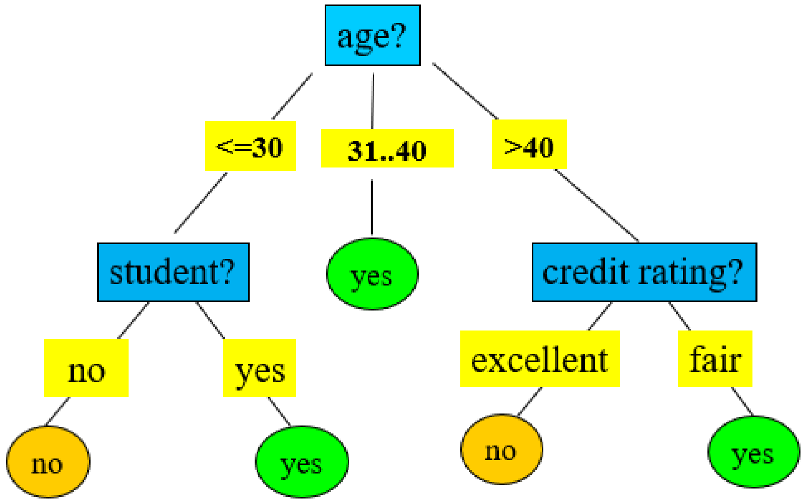
Buys computer = yes ? no



Age	**Student**	**Credit rating**	**Buys computer**
28	Yes	Fair	**?**

What is Decision Tree?

- Flow-chart-like tree structure
  - Root node: a test on root node
    - Age < 30
  - Internal node: a test on an attribute
    - Student = yes ?
  - Leaf node: classes
    - Buys computer: yes or no?



Training set

- Whether a customer is likely to buy a computer?

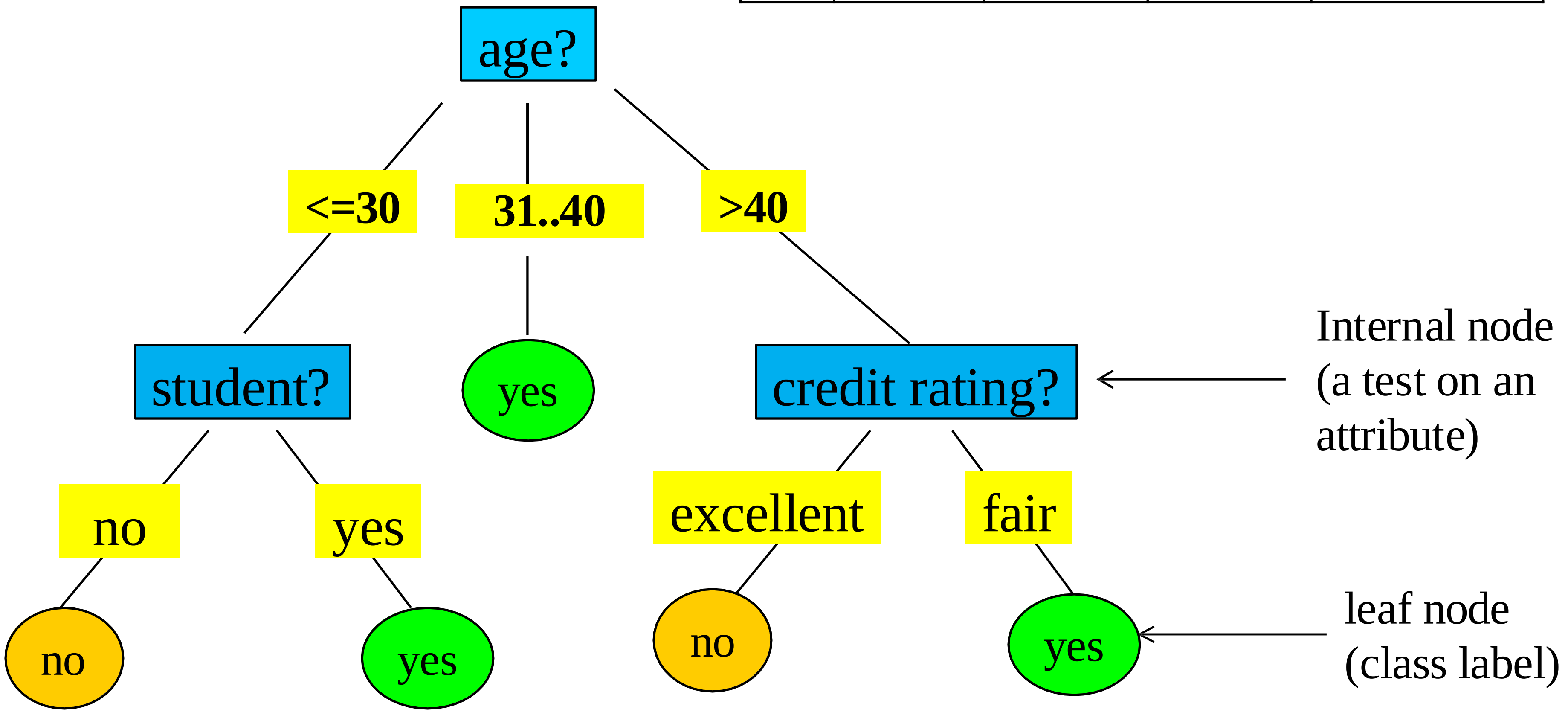
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent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

Data Preparation

- Data Cleaning
  - Pre-process data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
- Data transformation
  - Generalise and/or normalise data

Decision Tree: “Buys Computer”

Age	Income	Student	Credit rating	Buys computer
25	Low	Student	Fair	?



Decision Tree Induction

- Many algorithms:
  - Hunt's Algorithm
  - ID3 – late 1970's, J. Ross Quinlan
  - C4.5 – successor of ID3
  - CART – similar approach as C4.5
- General Structure
  - Attribute selection by calculating entropy and Information Gain
  - Decision tree induction
  - Rule generation

A Simple Thought on Decision Tree

- A simple minded algorithm:
  - If Age = a and Income = b and Student = c and Credit Rating = d
  - then buys_computer = ?
- This structure will yield a branch for each row, not eliminating superfluous attributes, and is unnecessarily complex.
- The complete classification space for m attributes is of size: $\prod_{j=1}^{m} N_j$

(N_j is the number of values of attribute j)

The Problem Statement

- Given a set of data described by a set of attributes and an outcome class, the problem is to find a **minimum decision tree** that will classify the values of the class based on the values of given attributes.

The Problem Statement

- Given a set of data described by a set of attributes and an outcome class, the problem is to find a **minimum decision tree** that will classify the values of the class based on the values of given attributes.

- How do we select an attribute?
  - Amongst four attributes: **AGE, INCOME, STUDENT** and **CREDIT RATING, which attribute carries "more" information than others?**

Information Theory

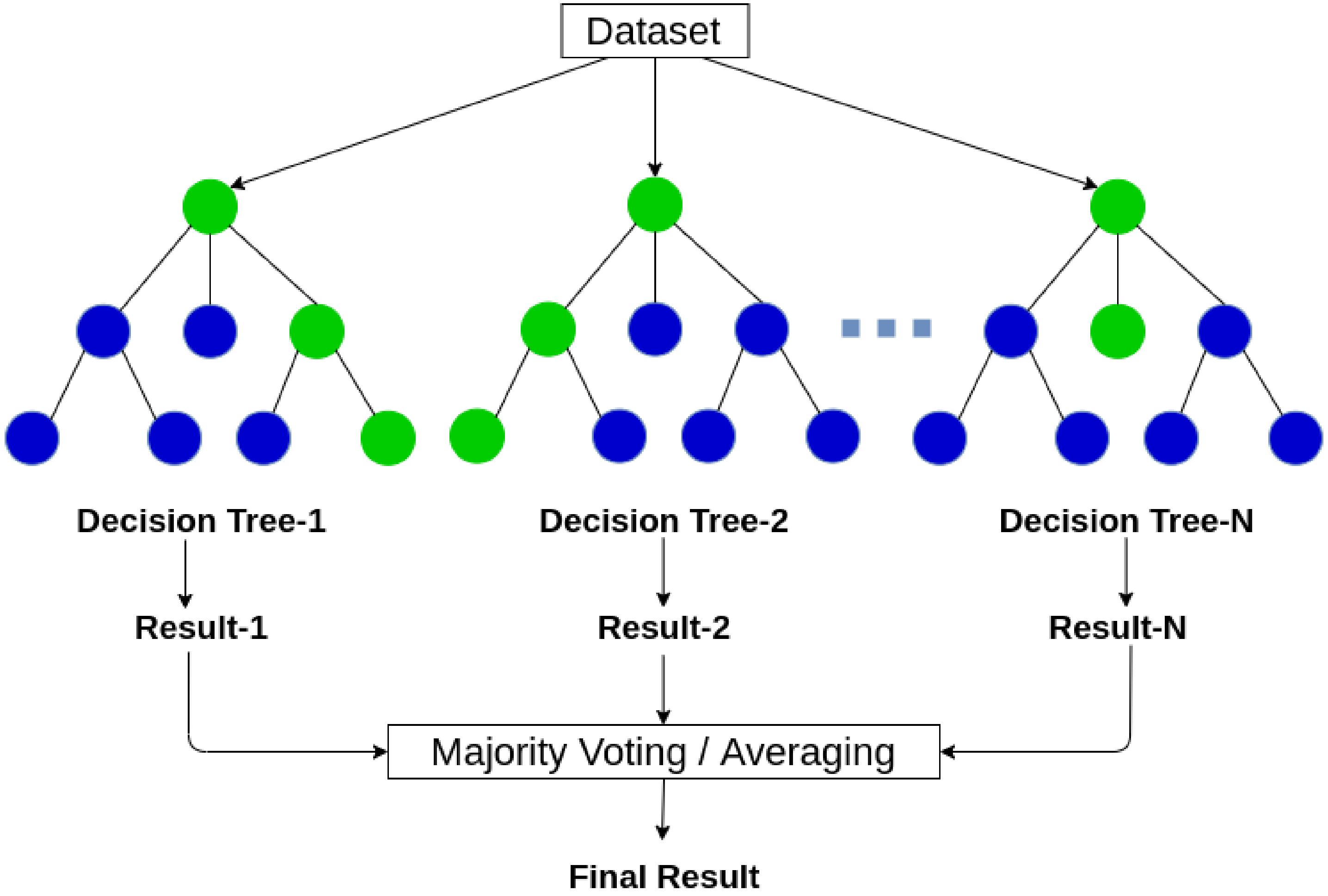
- Information Entropy → uncertainty
- Information Gain → reduce entropy

E = $-\sum p_i \log_2(p_i)$

IG = $E(Parent) - \sum w_i \, E(Child_i)$

Random Forest

- Combines the output of multiple (randomly created) Decision Trees to generate the final output.

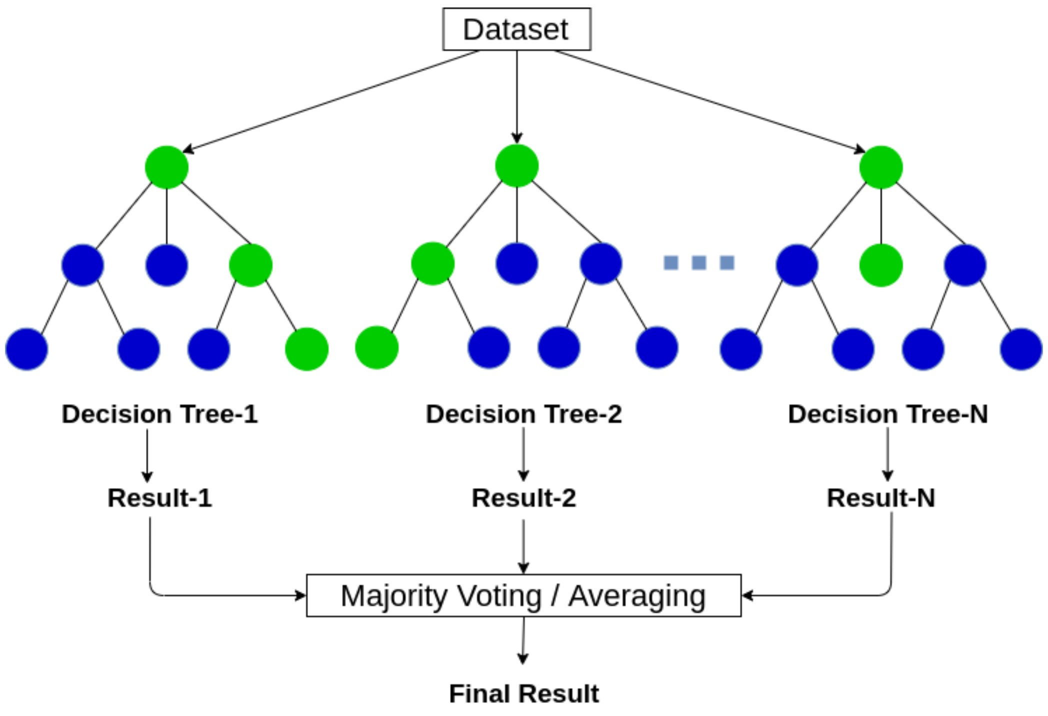


Random Forest

- **Individual Trees:** Each Decision Tree is trained on a random subset of the data (with replacement, i.e., bootstrapping) and a random subset of features.

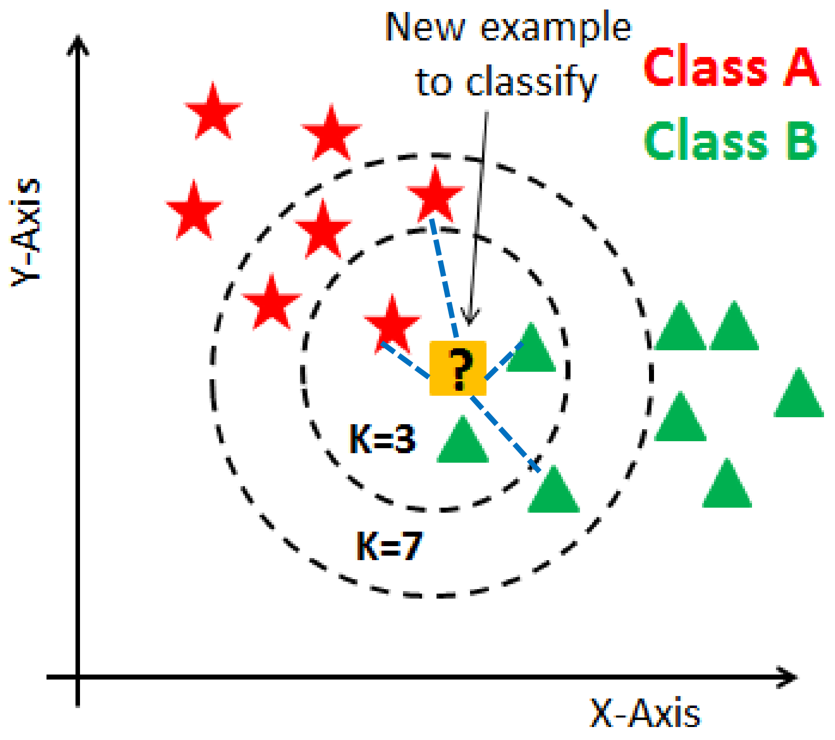
- **Tree Predictions:** Each tree makes an independent prediction based on its learned splits and conditions.

- **Aggregation by Majority Voting:** The Random Forest combines the predictions of all trees, and the class with the highest votes becomes the final classification.



K-Nearest Neighbours (kNN)

- Choose the value of k (number of neighbours).
- Compute the distance between the query point and all other data points.
- Identify the k nearest neighbours.
- Assign the class label based on the majority vote among k neighbours.
- In case of regression, take the mean of the k neighbours' values.

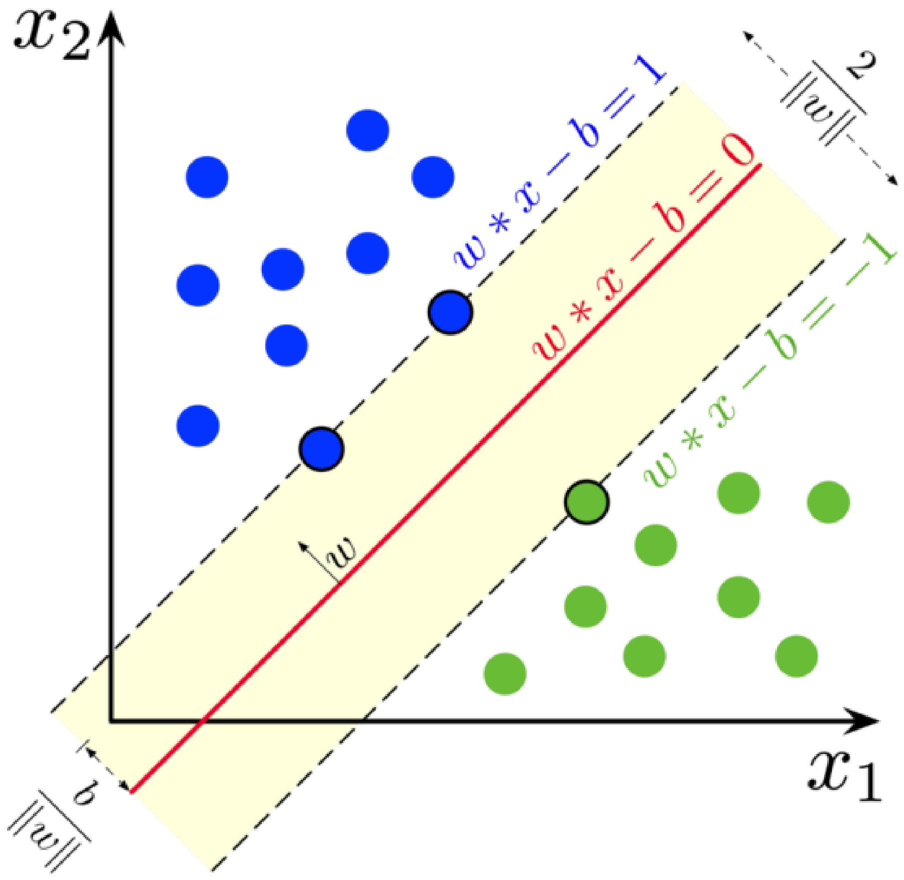


Note: Different distance measures e.g. Euclidean distance, Hamming distance, Manhattan distance, Minkowski distance etc.

Support Vector Machine (SVM)

- The objective is to find a hyperplane in an n-dimensional space that separates the data points to their potential classes. The hyperplane should be positioned with the maximum distance to the data points.

- The data points with the minimum distance to the hyperplane are called Support Vectors

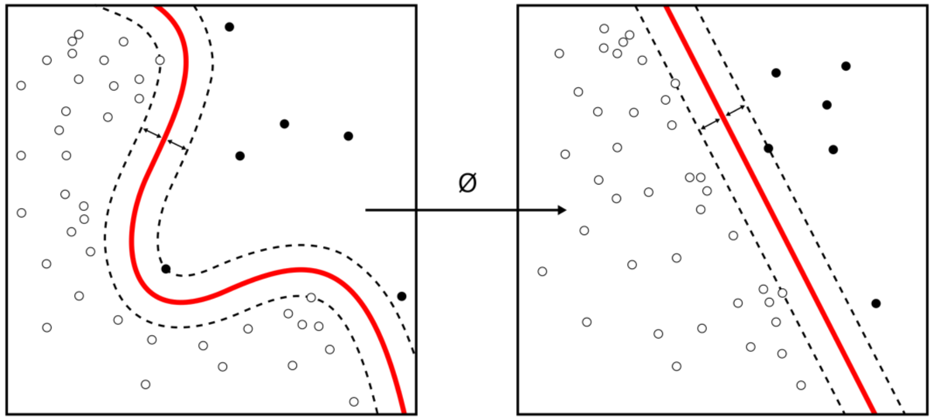


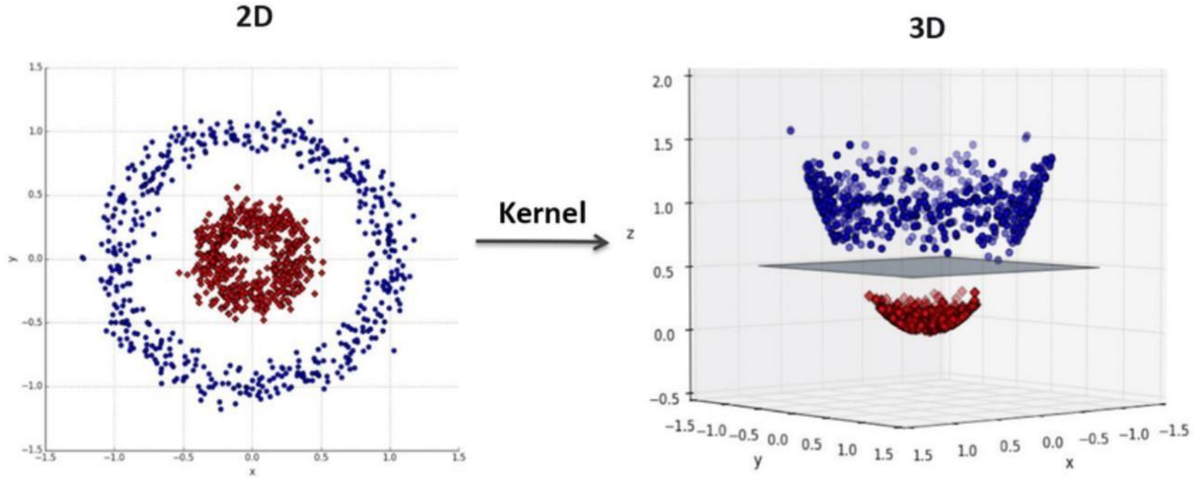
SVM

- The original maximum-margin hyperplane algorithm proposed by Vapnik in 1963 constructed a linear classifier that works when data are linearly separable (using a line or hyperplane)

- For data that are not linearly separable, we need to use a kernel function and transform data to another space where they are linearly separable. This is called the kernel trick.

- There are different kernel functions including linear function, Polynomial function, Radial basis function (RBF), and Sigmoid function.





Evaluation

- Confusion matrix
- Accuracy
- Precision
- Recall

Evaluation

Confusion Matrix

	Actual Positive (Apple)	Actual Negative (Non-apple)
Predicted Positive (Apple)	TP	FP
Predicted Negative (Non-apple)	FN	TN

TP: true positive

FN: false negative

FP: false positive

TN: true negative

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

NOTE: Accuracy is NOT always a good measure!

Confusion Matrix

Total fruits: 20

	Actual Positive (Apple)	**Actual Negative (Non-apple)**
Predicted Positive (Apple)	10 TP	5 FP
Predicted Negative (Non-apple)	2 FN	3 TN

TP: true positive

FN: false negative

FP: false positive

TN: true negative

Confusion Matrix

Total fruits: 20

	Actual Positive (Apple)	Actual Negative (Non-apple)
Predicted Positive (Apple)	10 TP	5 FP
Predicted Negative (Non-apple)	2 FN	3 TN

True Positive (TP): the classifier correctly makes positive decisions

True Negative (TN): the classifier correctly makes negative decision

False Positive (FP): the classifier mistakenly makes positive decisions

False Negative (FN): the classifier mistakenly makes negative decisions

Limitation

Suppose

- Total number of fruits in the testing examples = 10,000
- Number of Non-apple = 9990
- Number of Apple = 10

Can you classify Apple?

If model predicts everything to be of class non-apple, the accuracy is 9990/10000 = 99.9 %!!!

Here, accuracy is misleading because model cannot detect any Apple at all, still achieving high accuracy.

Evaluation

	Actual Positive (Apple)	Actual Negative (Non-apple)
Predicted Positive (Apple)	TP	FP
Predicted Negative (Non-apple)	FN	TN

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

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

$$F_1 = 2 * \frac{Pression * Recall}{Pression + Recall}$$

Evaluation

	Actual Positive (Apple)	Actual Negative (Non-apple)
Predicted Positive (Apple)	TP	FP
Predicted Negative (Non-apple)	FN	TN

$$Pression = \frac{TP}{TP + FP} = \frac{TP}{predicted\ positive}$$

Precision is the total number of **correctly identified actual apple cases out of retrieved apple**

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

Recall is the number of **correctly identified apple cases from the total number of actual apple cases**

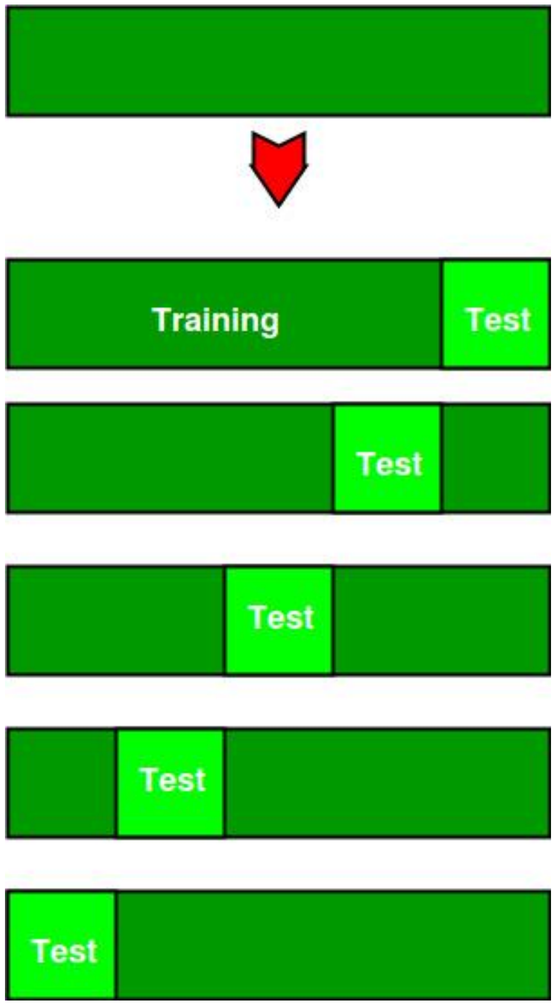
Evaluation

- Methods for Performance Evaluation
  - How to obtain reliable estimates?
  - i.e. how to partitioning the data?

- N-Fold cross validation
- Training and Testing sets

Cross Validation

- Ensures generalization to unseen data.
- Prevents overfitting.
- Provides reliable performance evaluation.
- Improves model robustness and reduces evaluation bias.
- Popular approaches:
  - K-Fold Cross Validation
  - Leave-One-Out



train_test_split()

x_train, x_test, y_train, y_test = train_test_split(features, targets, test_size= .30, random_state = 50)

features targets

age	competition	type	profit	targets
old	no	software	down	0
midlife	yes	software	up	0
midlife	no	hardware	down	0
old	no	hardware	down	1
young	no	hardware	up	1
young	no	software	up	1
midlife	no	software	up	0
young	yes	software	up	1
midlife	yes	hardware	down	1
old	yes	software	down	1

Company.csv

x_train y_train

old	no	software	down	0
midlife	yes	software	up	0
midlife	no	hardware	down	0
old	no	hardware	down	1
young	no	hardware	up	1
young	no	software	up	1
midlife	no	software	up	0
young	yes	software	up	1
midlife	yes	hardware	down	1
old	yes	software	down	1

train

x_test

Test

y_test