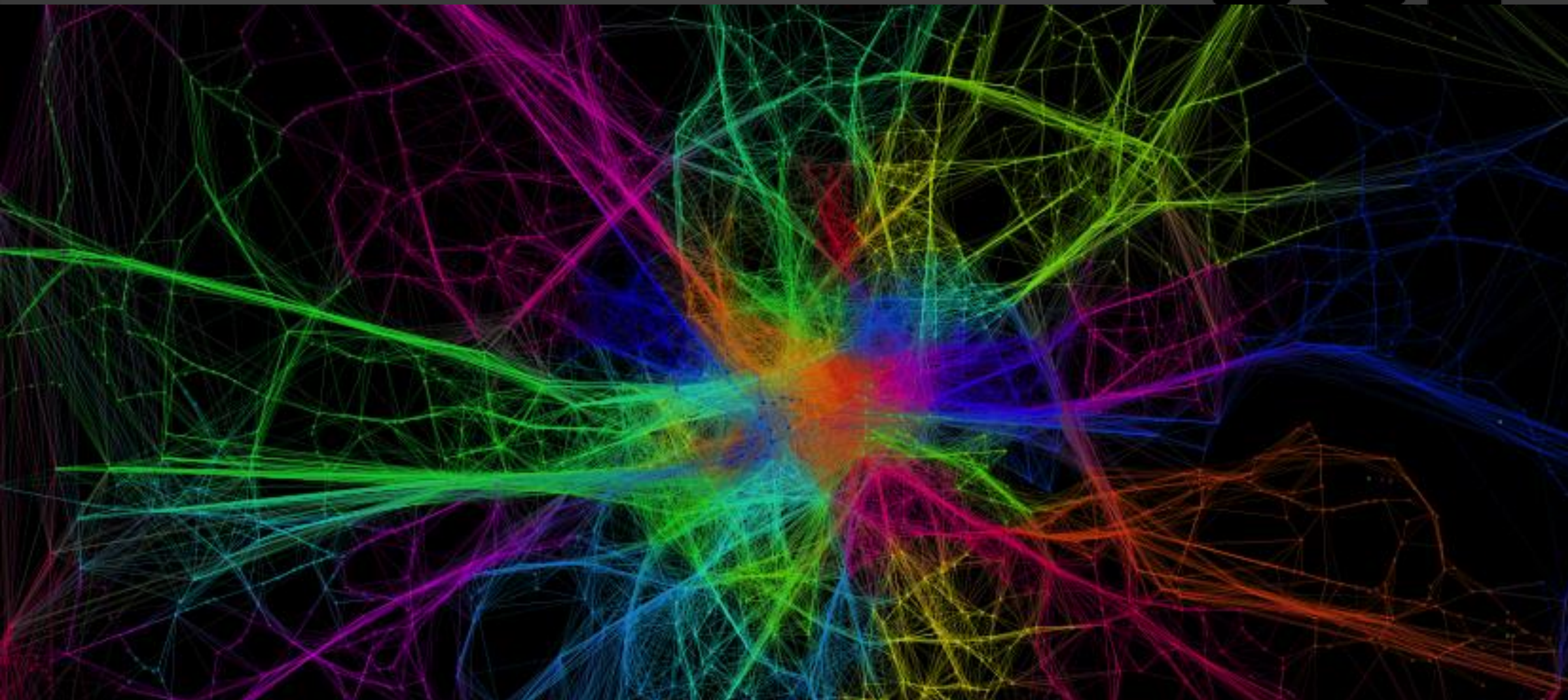


Classification

CASA0006: Data Science for Spatial Systems

Huanfa Chen

Some slides courtesy of Ed Manley

CASA0006

1. Introduction to Databases
2. Introduction to SQL
3. Advanced SQL
4. Data Munging
5. Advanced Clustering
6. Advanced Regression
7. Classification
8. Dimension Reduction
9. Unstructured Data
10. Analysis Workflow

Recap

What we already know

We can handle and clean data

Database, SQL, Python Pandas/Sklearn

We can do clustering analysis

Kmeans, DBSCAN, hierarchical clustering

We can do regression analysis

Linear regression, VIF, Lasso, decision tree, random forest

Today we extend our skills in data analysis, exploring the use of **classification methods** for analysing data

Data Analysis

Picking an Approach

The approach to take towards analyzing your data depends on what you want to understand from it

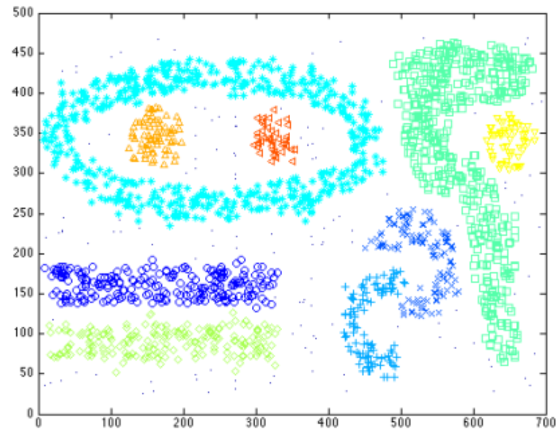
	Method		Output
Input Dataset ➡	Clustering	→	Creation of Groupings
	Regression	→	Identify Data Relationships
	Classification	→	Identify Discrete Class
	Dimensionality Reduction	→	Understand Influential Factors
	Association Rule Mining	→	Identify Dependencies
	Anomaly Detection	→	Identify Outliers

Unsupervised: no ground truth

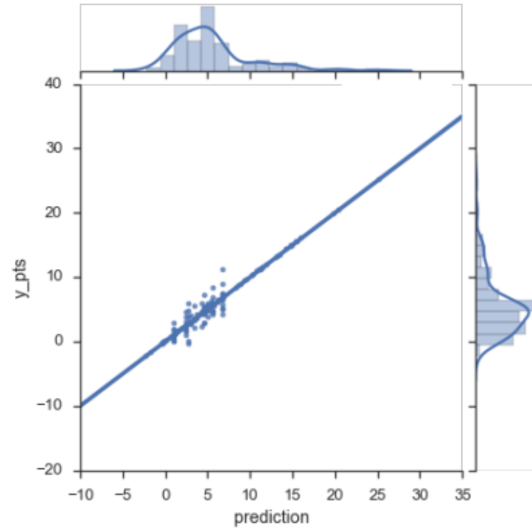
Supervised: with ground truth

Unsupervised Learning

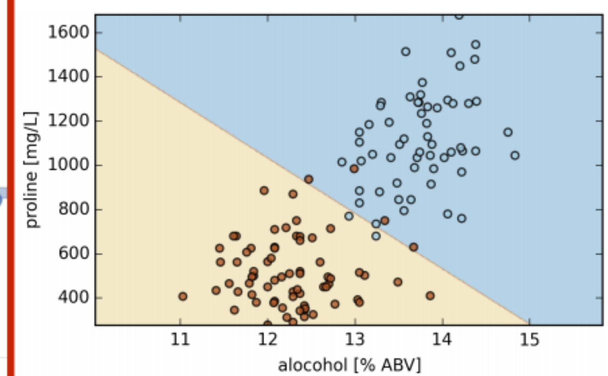
Supervised Learning



Clustering:
[DBSCAN on a toy dataset]



Regression:
[Soccer Fantasy Score prediction]



Classification:
[SVM on 2 classes of the Wine dataset]

Today's topic

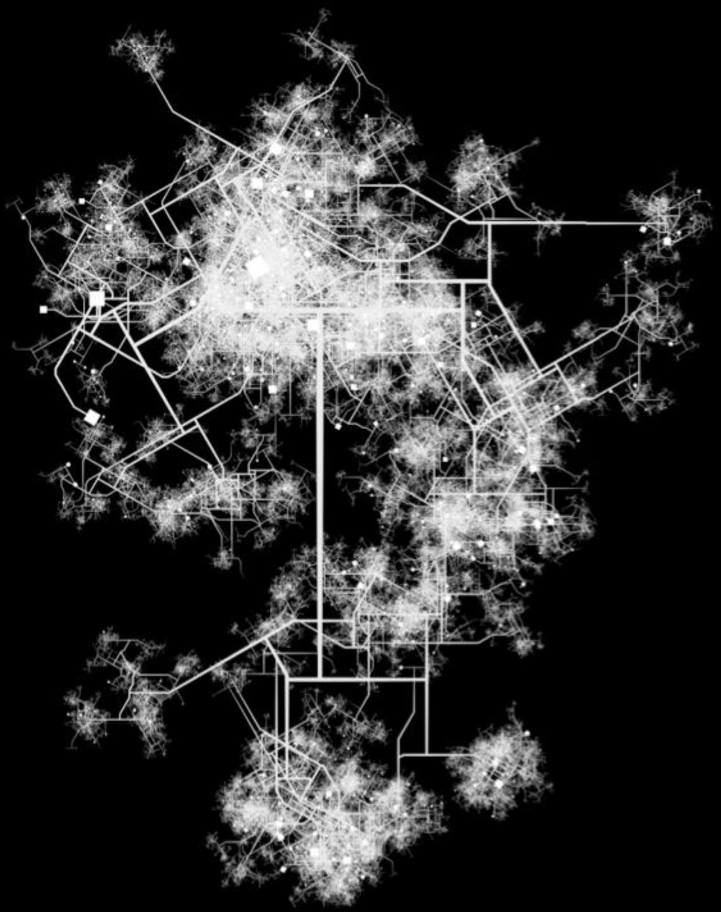
No labels

Continuous Y as labels

Discrete Y as labels

http://archive.ics.uci.edu/ml/datasets/Wine

Outline



1. Classification and Prediction
2. Supervised Classification
3. Classification Methods
    a. K Nearest Neighbours
    b. Logistic Regression
    c. Artificial Neural Networks
    d. Decision Trees
    e. Random Forests
4. Validation

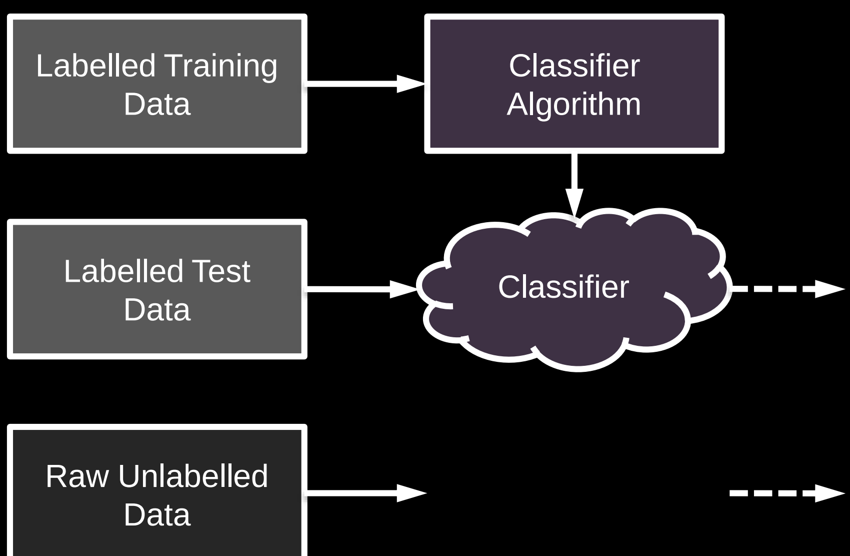
Supervised Classification

What is it?

We train a classifier with a **labelled** sample of data, and it figures out how to create a relationship between a feature's attributes and its label

Establishing this relationship allows us to predict future datasets

Example: travel mode detection

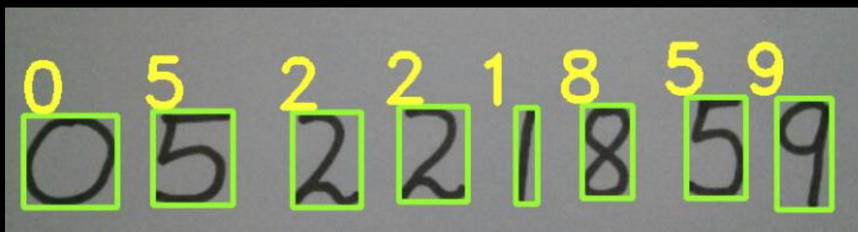


Label	Predicted Class
Car	Car
Car	Truck
Truck	SUV
SUV	SUV

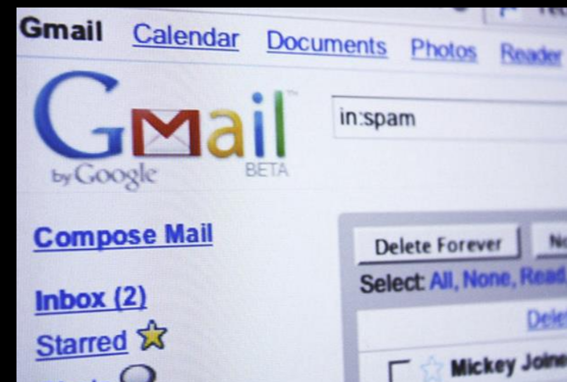
Predicted Class
Bicycle
Car
Truck
SUV

Classification

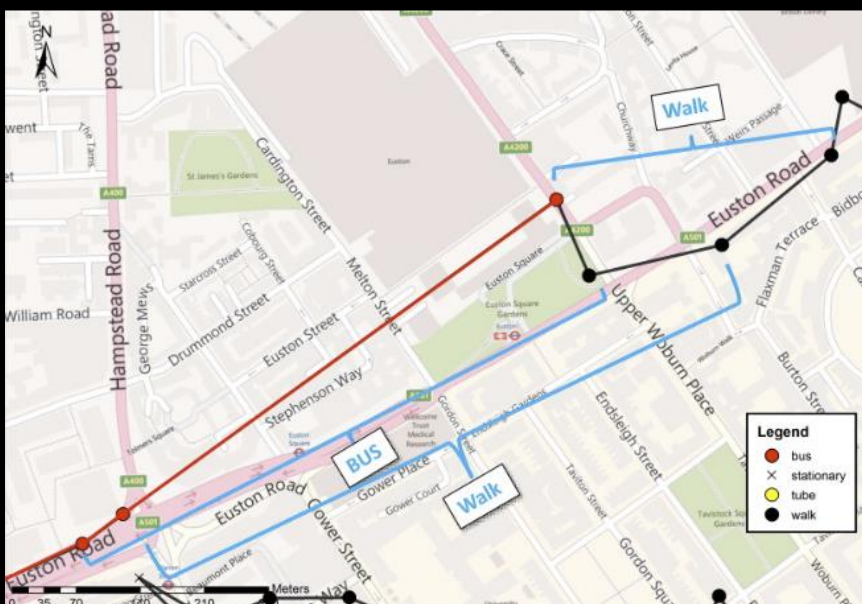
Examples



Digit Recognition



Spam Filtering



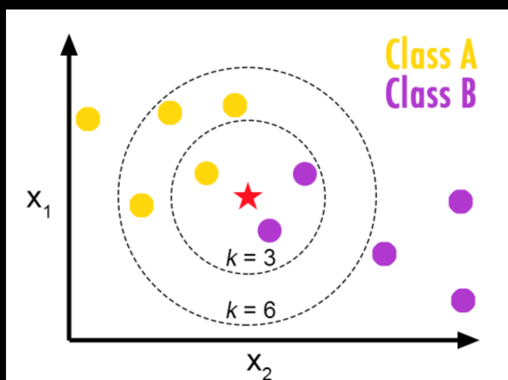
Transport Mode Detection
(from GPS data)

Classification Methods

Nearest Neighbours
Logistic Regression
Artificial Neural Networks
Decision Trees
Random Forests

Nearest Neighbours
k-NN

Considers **neighbouring points** and their classifications, and then classifies a point according to majority vote



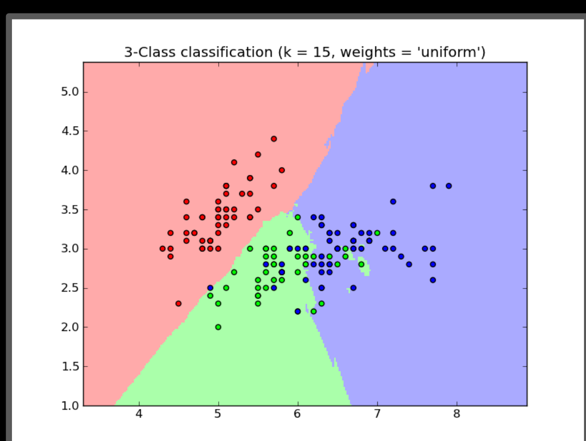
Need to tune k

Method
Choose *k* neighbours to consider; choose whether to weight neighbours by distance; classify using known points.

Hyperparameter
k

Pros
Simple; easy to understand; used widely

Cons
Sensitive to local patterns in the data, rather than accounting for wider trends; lack of interpretability



Classifier
True label

https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html

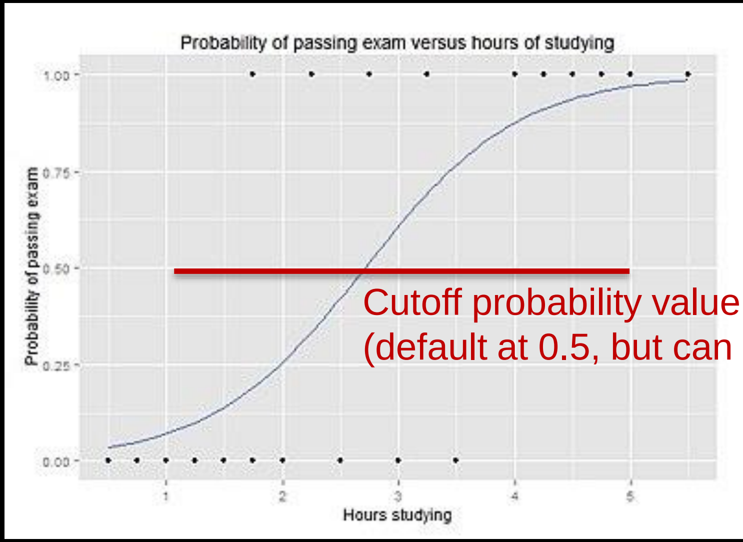
Logistic Regression (or logit regression)

Use a logistic function to model a binary response variable. Applications include predicting risk of developing disease and discrete choice models in economics.

Logistic function: $y = \dfrac{exp(\sum_{i=0}^{n} \beta_i x_i)}{1 + exp(\sum_{i=0}^{n} \beta_i x_i)}$

$\boxed{x} \rightarrow \boxed{y} \rightarrow \boxed{\text{binary class}}$

(-∞,+∞) [0,1] 0 or 1



Cutoff probability value
(default at 0.5, but can be predefined)

Pros
Simple; easy to understand; used widely

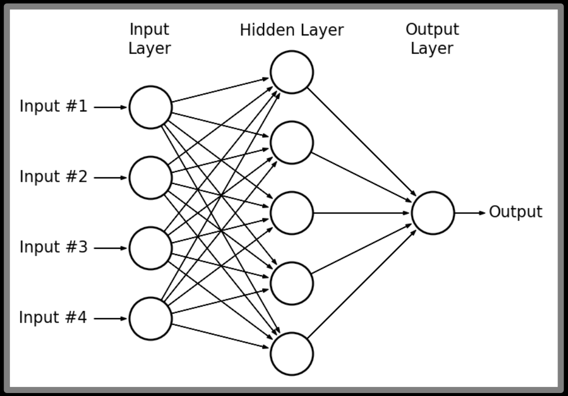
Cons
Subject to variable selection (like linear regression)

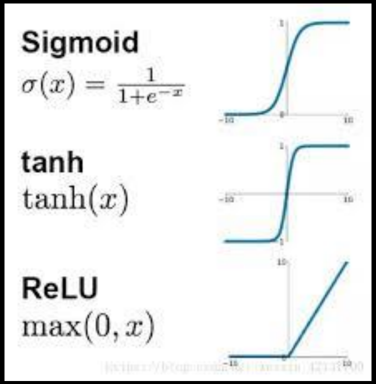
Types
- ***Binomial***: fail *vs* pass
- ***Multinomial*** (more than two classes): bus *vs* car *vs* cycling
- ***Ordinal*** (ordered multiple categories): fail *vs* pass *vs* merit *vs* distinction

Artificial Neural Networks
ANN

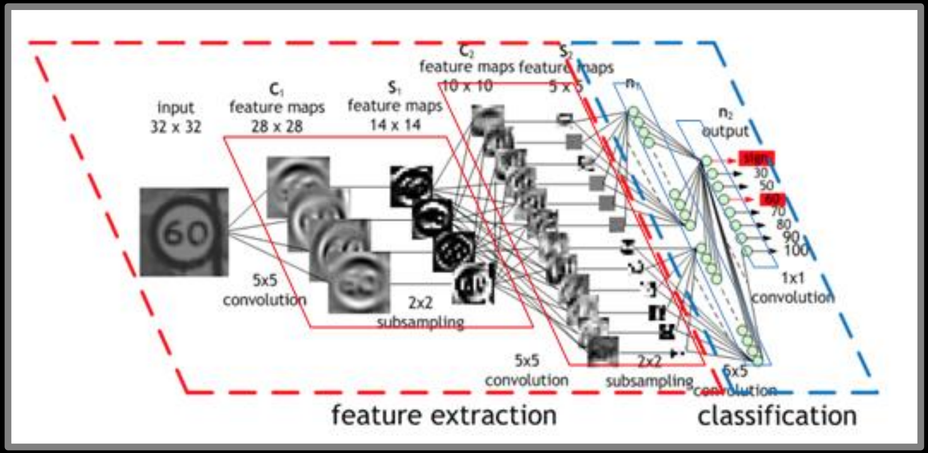
Informal: you can think of ANN as a multilayer logistic regression

ANNs link combinations of attributes to the activation of an artificial neuron





logistic function



Method
Combine weighted attribute values to form hidden layer. Combine weighted hidden functions to generate response. Class selected if response is strong enough.

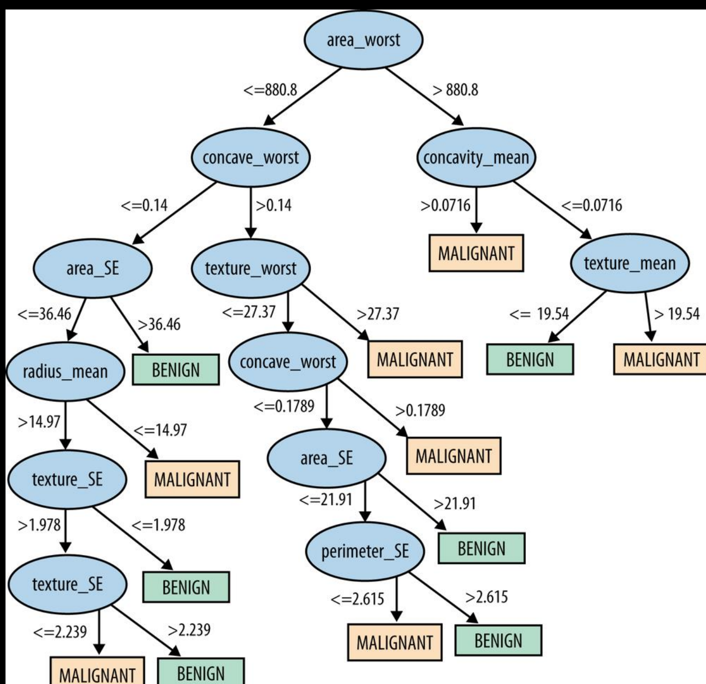
Pros
Easy to understand, used widely, and easy to implement, integrated with image and graphs (unstructured data)

Cons
Hard to understand association between inputs and outputs (non-linear function and many layers)

Decision Trees

Single Trees

Creates a tree of **probabilities** or **value ranges** linking attribute values to classes or outcomes



Method
Different methods (CART, ID3, C4.5). The common idea is to iteratively split a node based on some criteria.

Pros
Easy to understand and interpret; able to extract structure; able to handle numerical and categorical data; limits influence of poor predictors

Cons
Prone to overfitting (complex trees)

Decision Trees

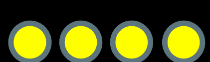
Example of CART

- CART (Classification and Regression Trees)

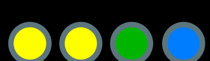
- Choose the best split that maximises the purity of the two new groups, or minimises the ***Gini impurity***

- ***Gini impurity:*** measures the impurity of a group containing different classes (where p_i is the probability of a class)

$$I_G(p) = \sum_{i=1}^{J} p_i(1 - p_i)$$



- Gini = 0. (if and only if only one class in the set)



- Gini = 0.5*(1-0.5) + 0.25*(1-0.25) + 0.25*(1-0.25) = 0.625

Decision Trees

Example of CART

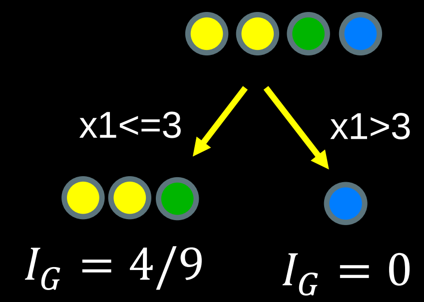
x1	x2	y
2	3	Yellow
3	4	Yellow
3	4	Green
5	3	Blue

Gini score

$$I_G(p) = \sum_{i=1}^{J} p_i(1 - p_i)$$

Splitting a node to minimise the Gini score by comparing all splits

A. [x1, 2] 2/3
B. [x1, 3] 4/9
C: [x1, 5]
C. [x2, 3] 1
D: [x2, 4]



The output of a decision tree can be the predicted class or a probability over all classes

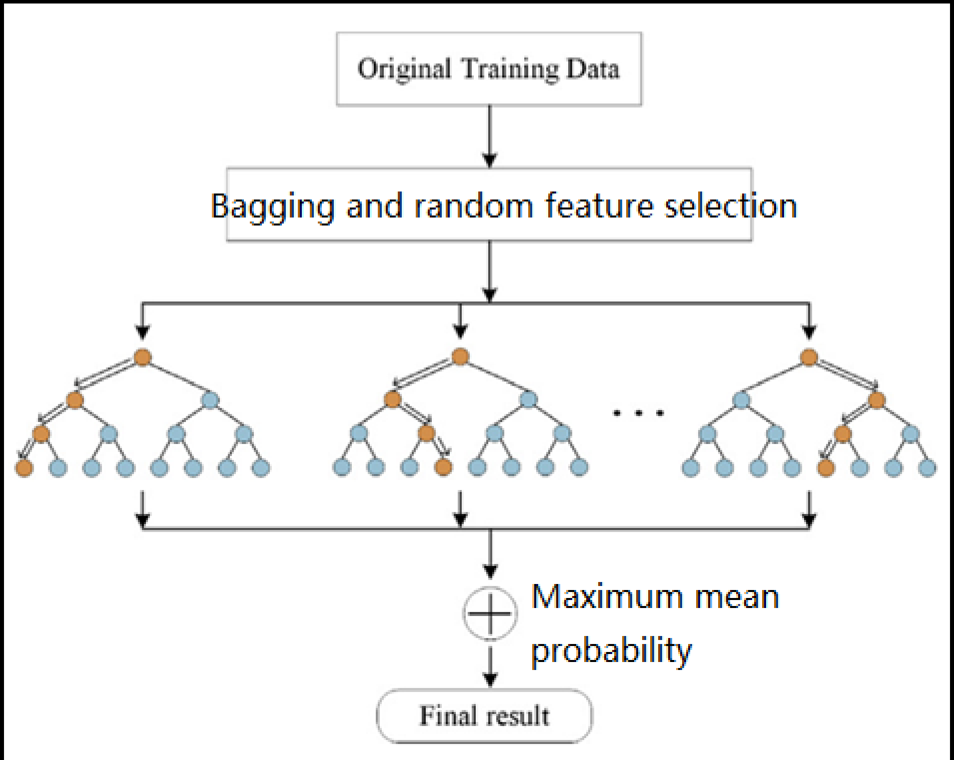
Example: given the input of ***(x1=2.5, x2=4000)***

- Predicted class: yellow
- Predicted probability (in the left leaf):
    - yellow = 2/3
    - green = 1/3
    - blue = 0

Random Forest

Creating many trees and combining their response

- A single tree may be overfitting and does not perform well on new datasets.
- A good solution is to randomly grow a bunch of random and different trees.



- Given an input, the response is a combination (e.g. maximum mean probability) of the output of all trees.

Amended from source image:
Cheng, L., Chen, X., De Vos, J., Lai, X., and Witlox, F. (2019) Applying a random forest method approach to model travel mode choice behavior. *Travel Behaviour and Society*

Random Forest

Growing a different tree

- Each tree is grown by a random subset of data
    - Resampling training data with replacement
    - Sampling a random selection of predictors

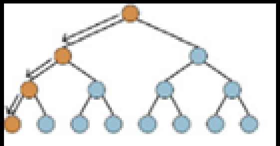
- This guarantees that each tree is different.

Index	x1	x2	x3
1	2	1.0	2
2	3	1.5	3
3	5	2.0	4
4	4	2.6	6

Sampling:
2 features,
7 samples

Index	x1	x3
1	2	2
2	3	3
3	5	4
4	4	6
2	3	3
4	4	6
2	3	3

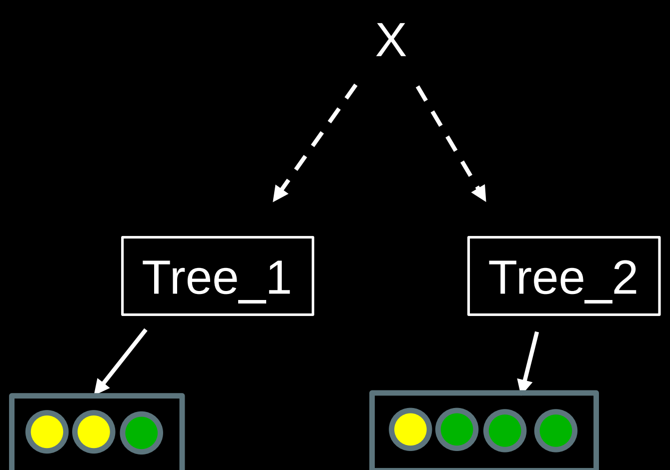
Train a
tree



Random Forest

Combining response

- Combining response from many trees into one output (**sklearn package**)
  - predict(X): the predicted class is the one with highest mean probability estimate across the trees
  - predict_proba(X): the mean predicted class probabilities of the trees in the forest. The class probability of a single tree is the fraction of samples of the same class in a leaf.
  - Note that other packages may differ on the calculation of output class/probability of a random forest



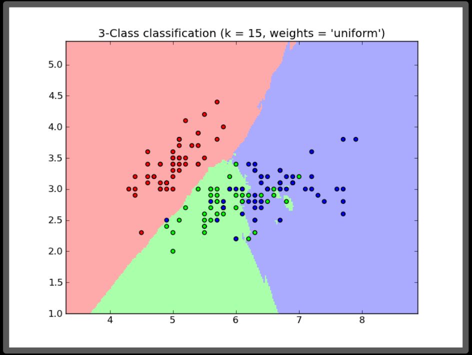
class	Tree_1	Tree_2	output_proba	output_class
Yellow	2/3	1/4	11/24	Green
Green	1/3	3/4	13/24	

Validation and Testing

Types of Error
Metrics

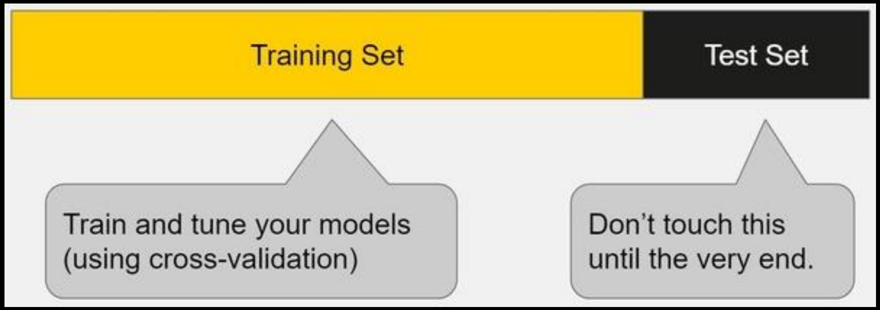
Hyperparamter & parameter

- We limit the discussion to machine learning.
- A **hyperparameter** is a parameter whose value is used to control the learning process. It should be predefined before model training. Many algorithms provide default value of hyperparameters.
- The values of other parameters (typically node weights) are derived via training.
- Example of KNN
  - Hyperparameters: k
  - Parameters: the class of each sub-areas.
- Another example of random forest
  - Hyperparameters: n_estimator (number of trees), max_depth (maximum depth of a tree), etc.
  - Parameters: the splits.

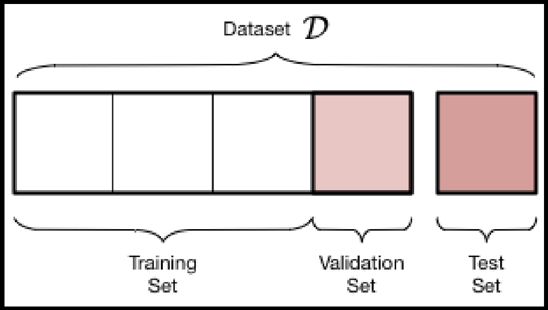


Data split (training & testing)

- Fair validation of model performance should be based on training/testing split. Don't touch the test set until the model is finalised.



- If you need to tune the hyperparameters in the model, you would need to use training/validation/test split. Validation set is used to determine the hyperparameters.



Cross Validation

A more complicated validation process

Cross Validation executes model training and validating on multiple subsets of the data

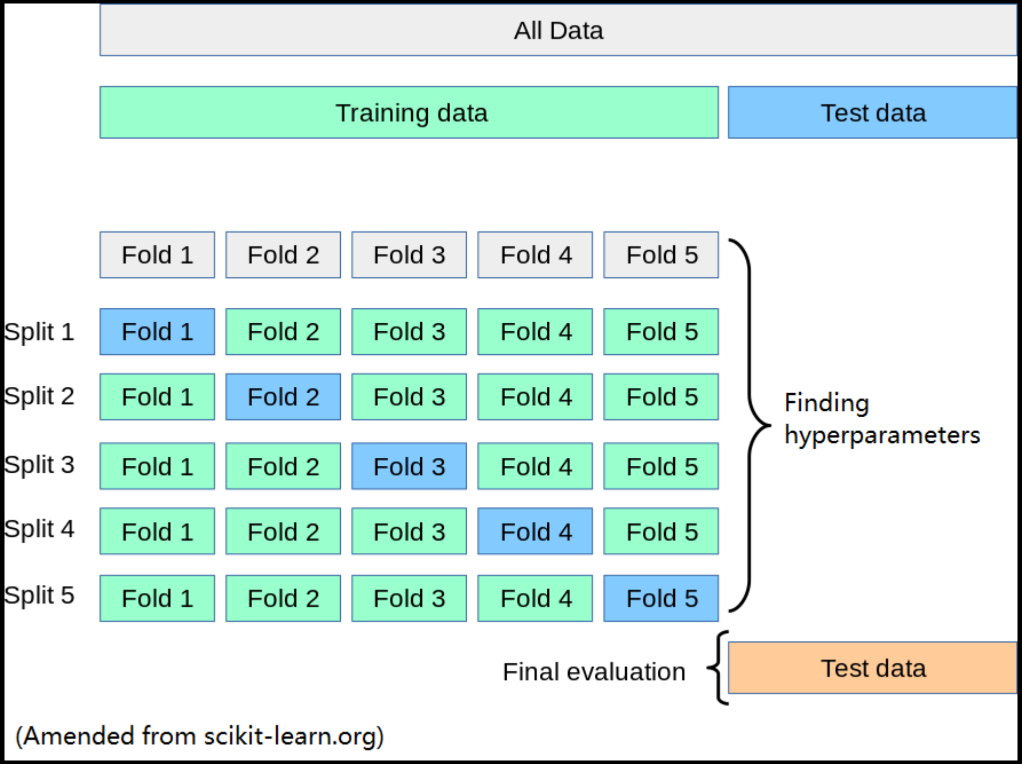
-- To test the model's ability to predict new data that was not used in training it
-- To avoid problems of overfitting or selection bias

K-Fold Cross Validation

Extreme case (K = #Training data)
Leave-one-out cross-validation

Accuracy = Mean(Split 1, Split 2, etc.)
Model hyperparameters = Best fitting (e.g. Split 5)

Note: K is a hyperparameter.



Validation Measures

Classification (suitable for two-class and multi-class)

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Confusion Matrix

Classification measures compares predicted against observed classes

Classification Accuracy

Proportion correctly classified. Not the be-all metric!

$$\frac{Correct}{Correct + Incorrect} = \frac{tp + tn}{tp + tn + fp + fn}$$

Precision

How many positive predictions are correctly classified?

$$\frac{tp}{tp + fp}$$

tp = true positive (yay)
fp = false positive (incorrectly flagged)

Recall

How many positive classes are correctly classified?

$$\frac{tp}{tp + fn}$$

tp = true positive (woo hoo)
fn = false negative (missed result)

F1

A balance between precision and recall, takes beta attribute which weights precision or recall

$$(1 + \beta^2)\frac{precision \, * \, recall}{\beta^2 precision + recall}$$

https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

An exercise

		Predicted/Classified	
		Negative	Positive
Actual	Negative	998	0
	Positive	1	1

- Accuracy = $\frac{tp+tn}{ALL} = \frac{999}{1000}$
- Precision = $\frac{tp}{tp+fp} = \frac{1}{1+0} = 1$
- Recall = $\frac{tp}{tp+fn} = \frac{1}{1+1} = \frac{1}{2}$
- $(\beta = 1)$: F1 = $(1 + 1^2)\,\frac{1*0.5}{1^2*1+0.5} = \frac{2}{3}$

The precision and recall can be related to Type I and Type II error in statistics.

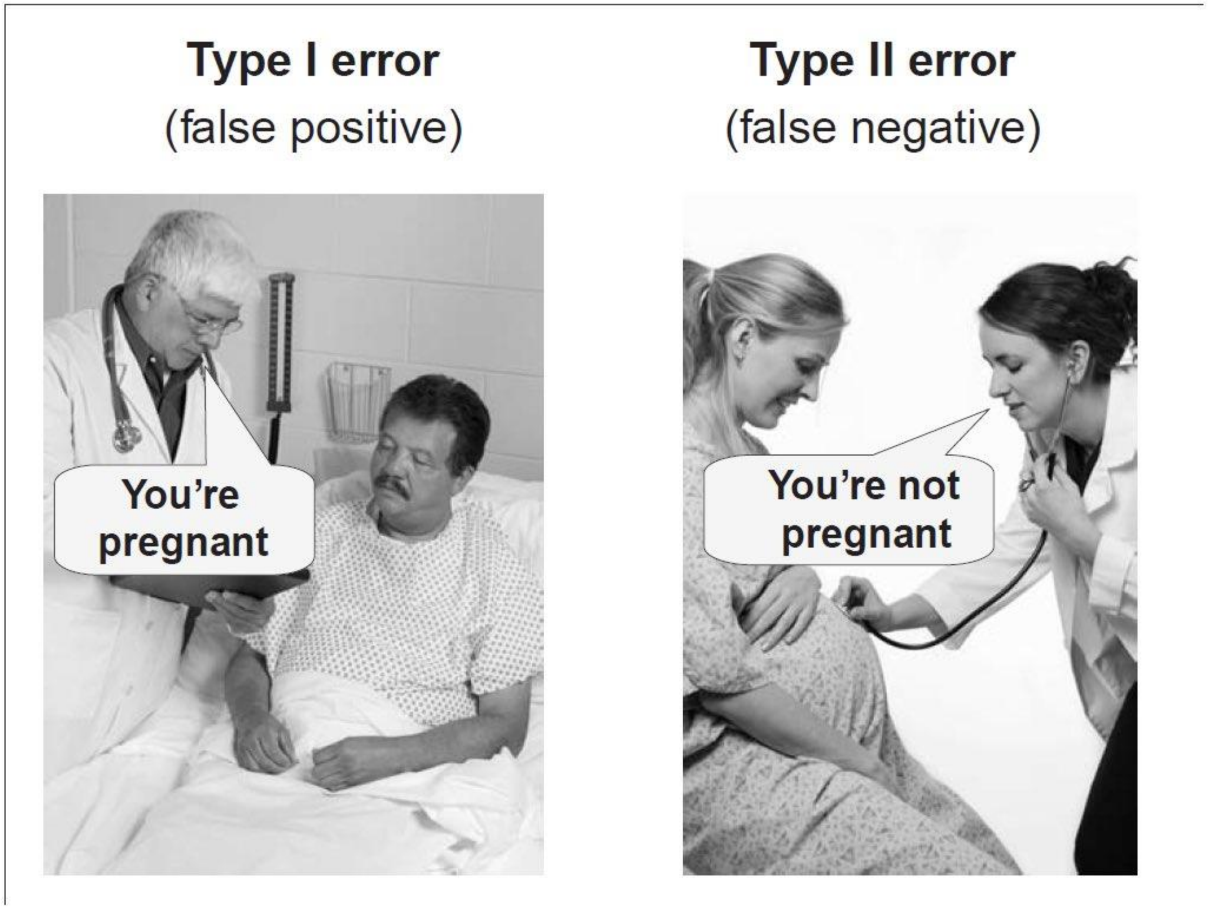


Figure 3.1 Type I and Type II errors

Thank You

Questions?

Huanfa Chen

huanfa.chen@ucl.ac.uk

Workshop
Classification

- In this workshop you will extend your skills in data mining by learning to implement data classifiers
- Once again you'll be using the Python sklearn library, which has a range of easy-to-implement methods for creating data classifiers
- Again, you're not expected to understand all of the maths and computation, only the usefulness and application of these approaches.
- **Download this week's Jupyter Notebook from Moodle, open it in Anaconda and work through**