Applied Statistical Learning in Python

KKU Datathon 2018

Khon Kaen, Thailand

Mon, 17 September 2018 (1-4PM)

Calvin J Chiew

https://github.com/calvinjchiew/kku18

Today’s Agenda

- Basics of Python and Jupyter Notebook
  - Hands-on Exercise
- Applied Statistical Learning in Python
  - Sample Code Review
  - Hands-on Exercise

Basics of Python and Jupyter Notebook

KKU Datathon 2018

Khon Kaen, Thailand

Mon, 17 September 2018 (1-4PM)

Calvin J Chiew

https://github.com/calvinjchiew/kku18

Terminology

- A **Jupyter notebook** consists of cells, which can contain either Markdown or code.
  - Each cell can be executed independently, but once a block of code is executed, it lives in the memory of the kernel.
- A **comma-separated values (CSV)** file stores tabular data in plain text.
  - Each record consists of values (can be numeric or text) separated by commas.
- **pandas** (typically imported as `pd`) can load csv data into dataframes which optimize storage and manipulation of data. Dataframes have useful methods eg. `head`, `shape`, `merge` etc.
- The pyplot module (typically imported as `plt`) in **matplotlib** contains useful functions for generating simple plots eg. `plot`, `scatter`, `hist`, `title`, `show` etc.

Basic Built-in Data Types

Integers	`7`
Floats	`7.0`
Booleans	`True, False`
Strings	`‘Hi’, “7.0”`
Lists	`[], [‘Hello’, 70, 2.1, True]`

- You can use either single or double quotation marks to enclose strings.
- **Lists** are collections of items, which can be of different types. They are indicated by square brackets, with items separated by commas.
- You do not need to declare the types of your **variables**. The type is inferred from the value assigned to the variable.

Beginner's Cheat Sheet

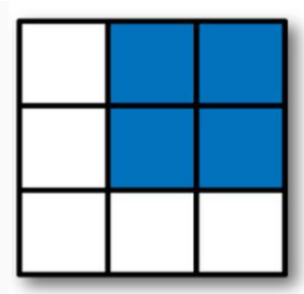
Arithmetic	`+, -, *, /, %, **, //`
Comparison	`==, !=, >, <, >=, <=`
Boolean logic	`and, or, not`
Indexing lists/strings	`[n], [n:m], [n:], [:n]`
Selection	`if, elif, else`
Iteration/loop	`for, in, range`
Create function	`def, return`
Call function	`function(arg1, arg2, …)`
Call object's method or library's function	`object.method(arg1, arg2, …)` `library.function(arg1, arg2, …)`
Get length of list/string	`len(…)`
Import library	`import … as …`
Print	`print()`

Indexing 2D Arrays

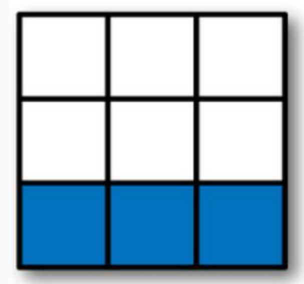
- **Shape** of 2D array is written as

  (number of rows, number of columns)

- The element at the n-th row and the m-th column is indexed as `[n, m]`. Just like lists, you can also get multiple array values at a time.
- Remember that Python is zero-indexed, ie. counting starts from **zero**, not one!

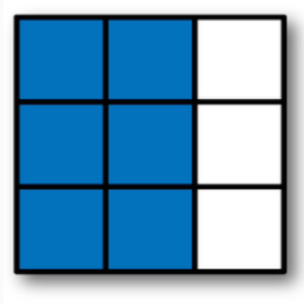


`arr[:2, 1:]`



`arr[2]`

`arr[2, :]`



`arr[:, :2]`

Best Practices

- Code readability is key. Python syntax is close to plain English.
- Variables should be given descriptive names.
- Intersperse your code with **comments**, which are indicated by `#`.
- Proper **indentation** is non-negotiable in Python. Code blocks are not indicated by delimiters eg. `{ }`, only by indentation.
- When in doubt, Google for help and read the documentation of libraries used.

Hands-on Exercise

https://github.com/calvinjchiew/kku18 >> ‘python’ folder >> Python.ipynb

Applied Statistical Learning in Python

KKU Datathon 2018

Khon Kaen, Thailand

Mon, 17 September 2018 (1-4PM)

Calvin J Chiew

https://github.com/calvinjchiew/kku18

Important Concepts for Today

1. **Model fit**
2. **Random forest**
3. **Support vector machine**
4. **Cross-validation**

- Summarized in handout

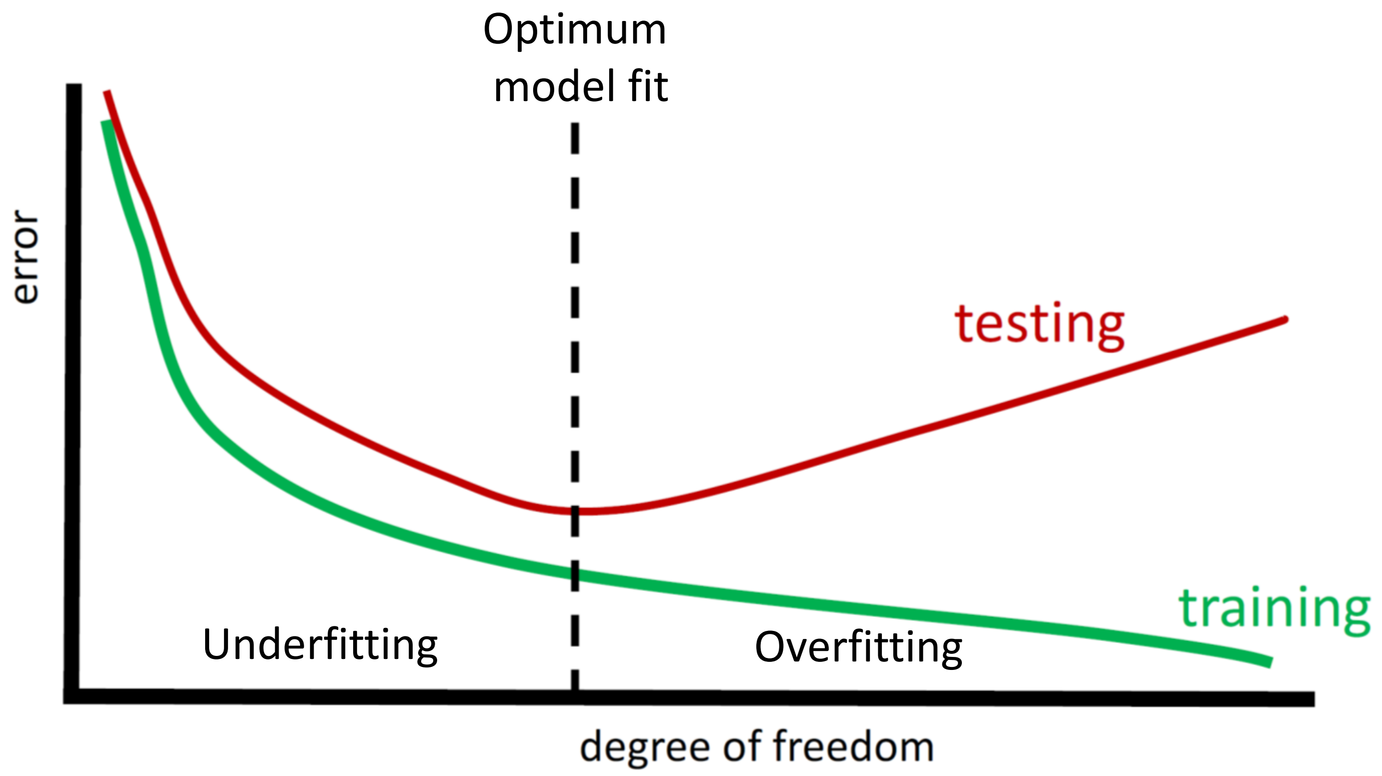
  https://github.com/calvinjchiew/kku18 >> 'lecture' folder >> handout.md

Model Fitting

- We want to estimate f where

$$Y = f(X_1, X_2, X_3 \ldots) + \varepsilon$$

  - X: feature, predictor, independent var
  - Y: outcome, response, dependent var
  - ε: error
- Data is split into distinct **training** and **testing** sets to prevent **overfitting**
- Loss (error) function depends on prediction task

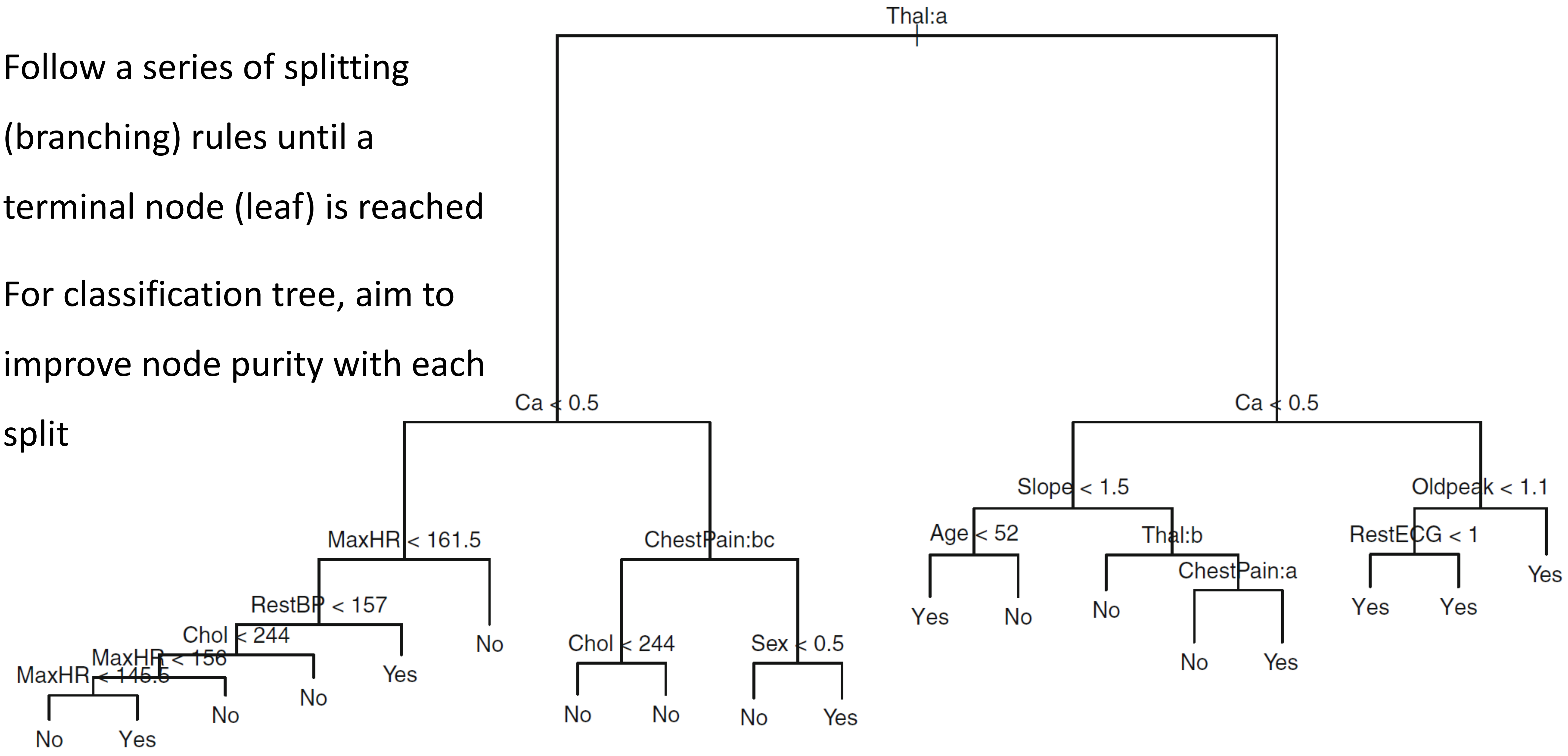


Popular ML Methods

- **Supervised Learning**
  - K-nearest neighbours
  - Regression (linear, logistic, polynomial, spline etc.) $\pm$ regularization
  - Linear/quadratic discriminant analysis
  - Tree-based approaches: decision tree, bagging, random forest, boosting
  - Support vector machine
  - Neural network
- **Unsupervised Learning**
  - Principal components analysis
  - Clustering
  - Neural network

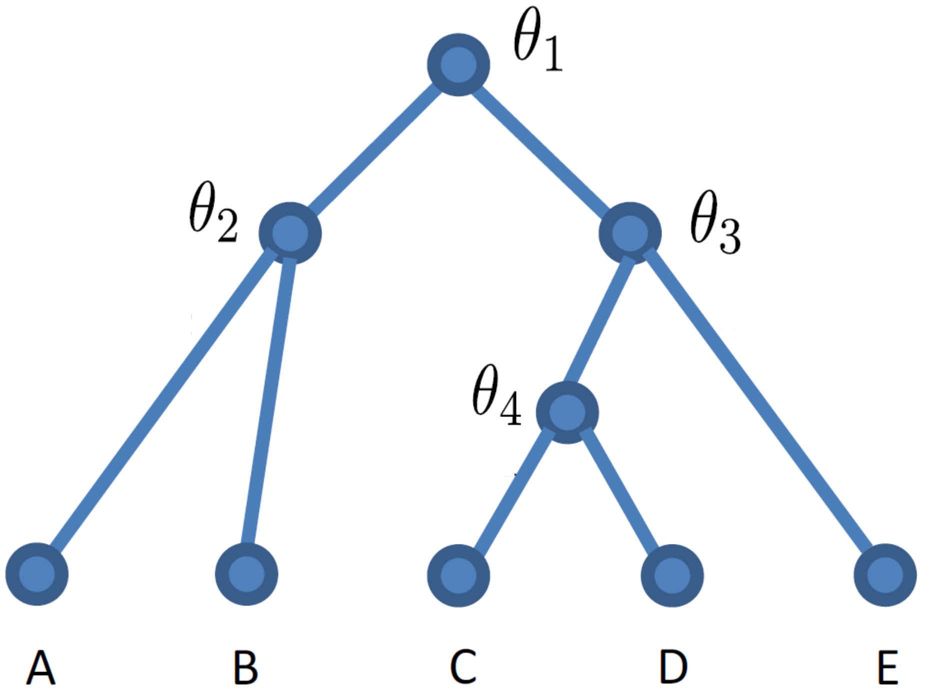
Decision Tree

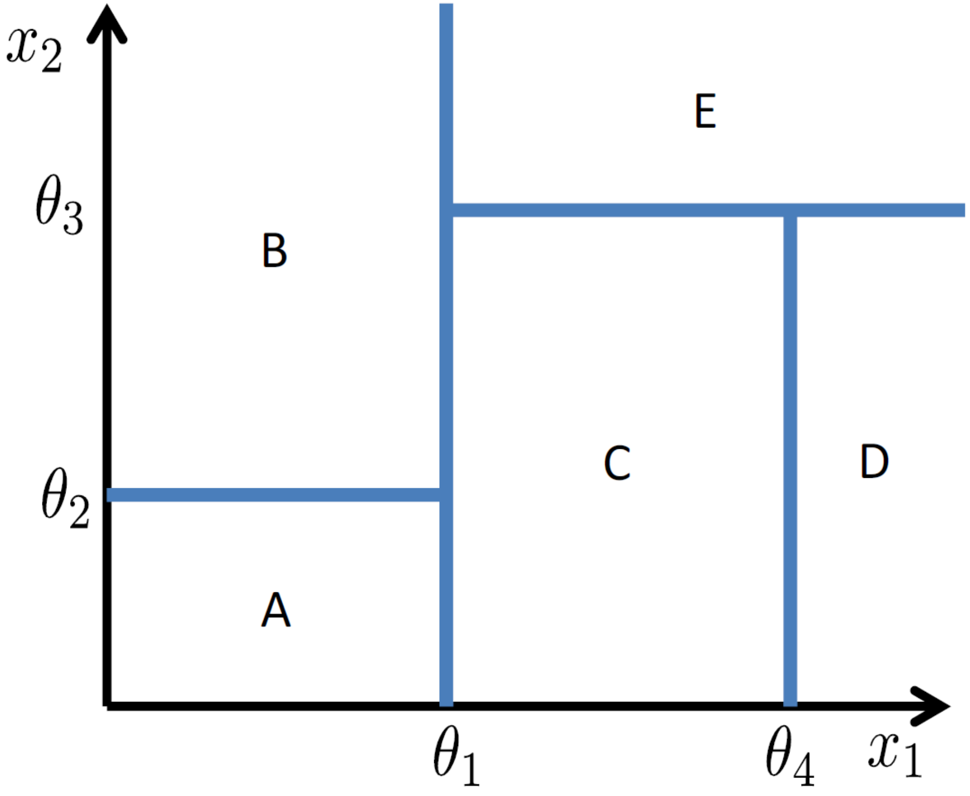
- Follow a series of splitting (branching) rules until a terminal node (leaf) is reached
- For classification tree, aim to improve node purity with each split



Decision Tree

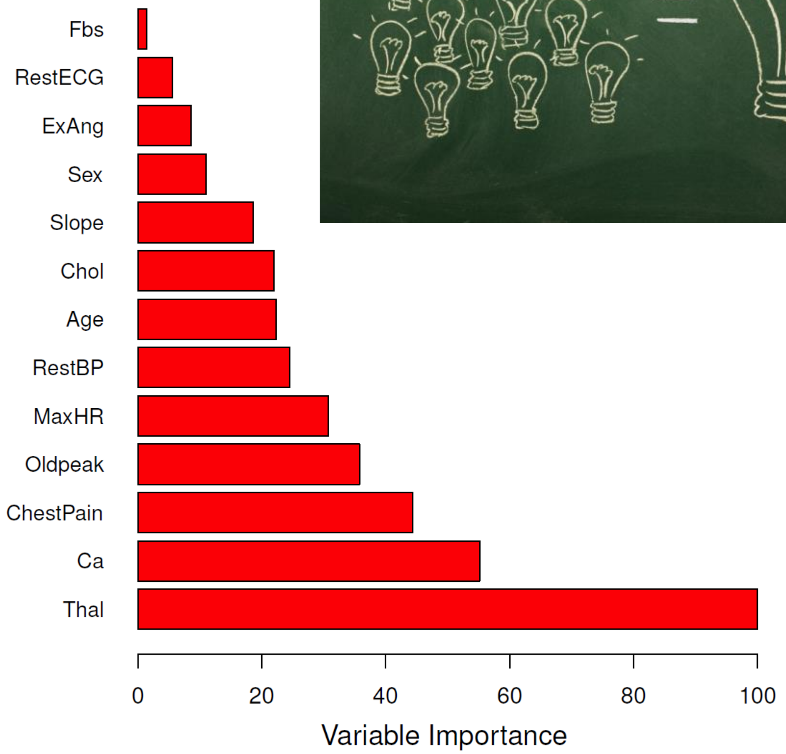
- The feature space is split into rectangular regions (boxes)
- We use the mean or majority class of observations in each region for prediction





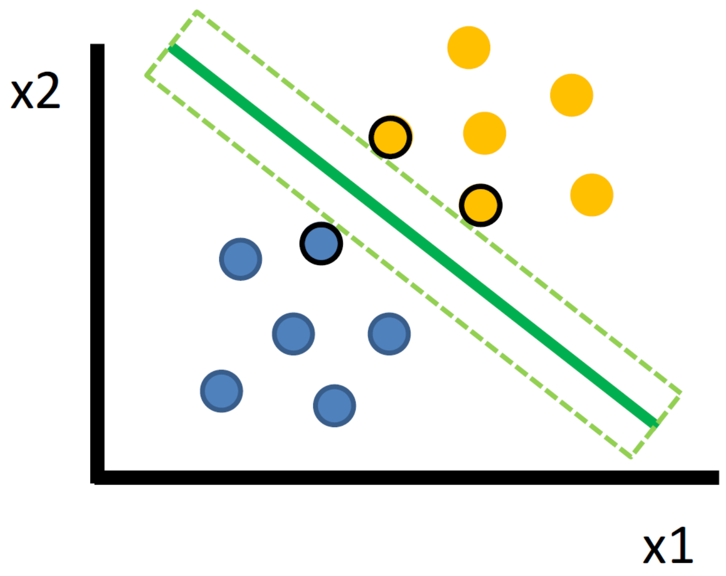
Random Forest

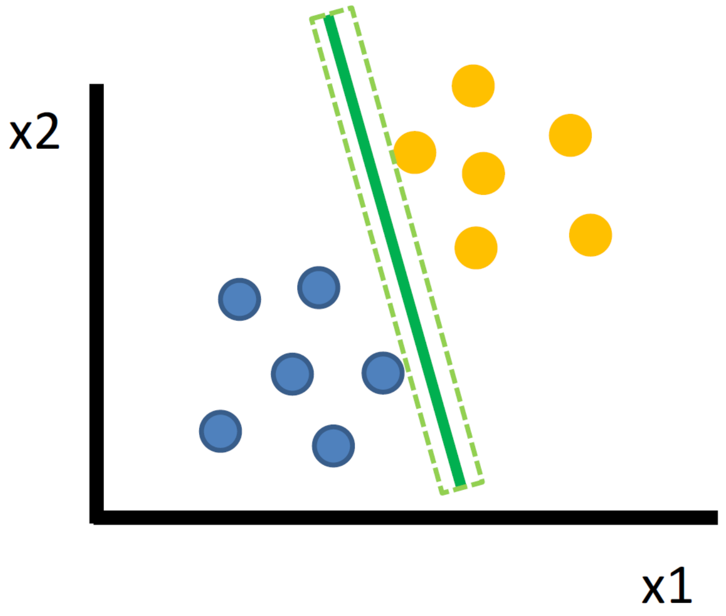
- Multiple trees which are combined to yield a single consensus prediction
  - Averaging multiple onerous predictions produce less uncertain results
- At each branch, only a **random subset** of all the predictors are considered as potential split candidates
  - To obtain trees that are less similar to each other
- Feature importance can be visualized by total decrease in Gini index due to splits over the feature



Support Vector Machine

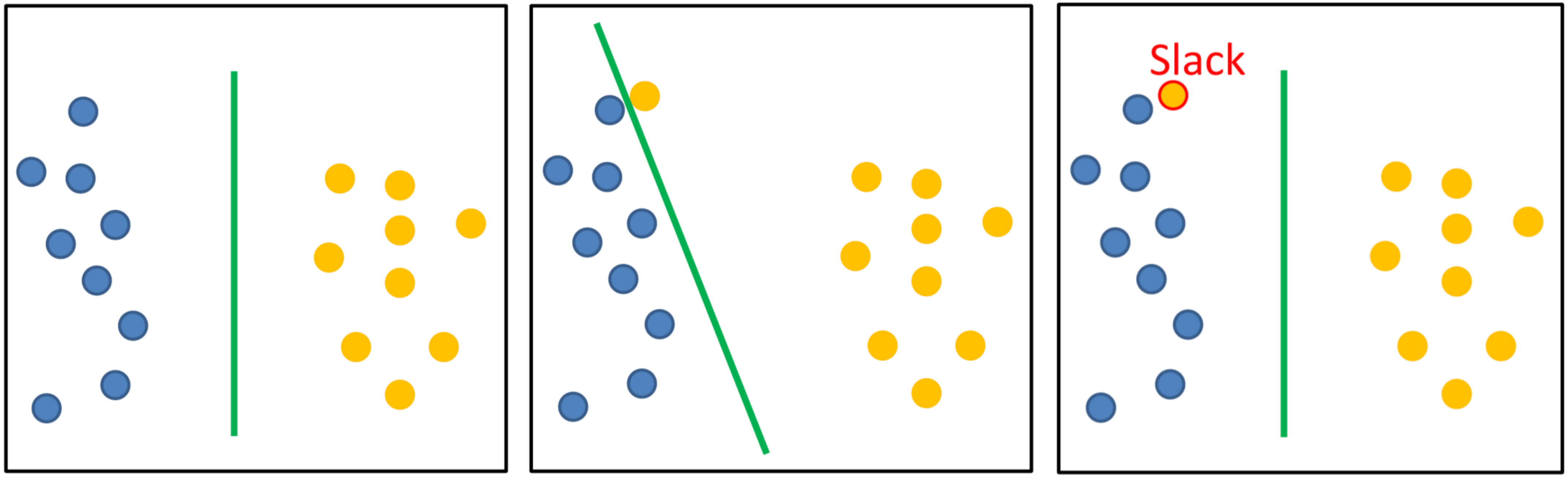
- The wider the **margin**, the more confident we are in the separating hyperplane
- Separating hyperplane depends only on the **support vectors**





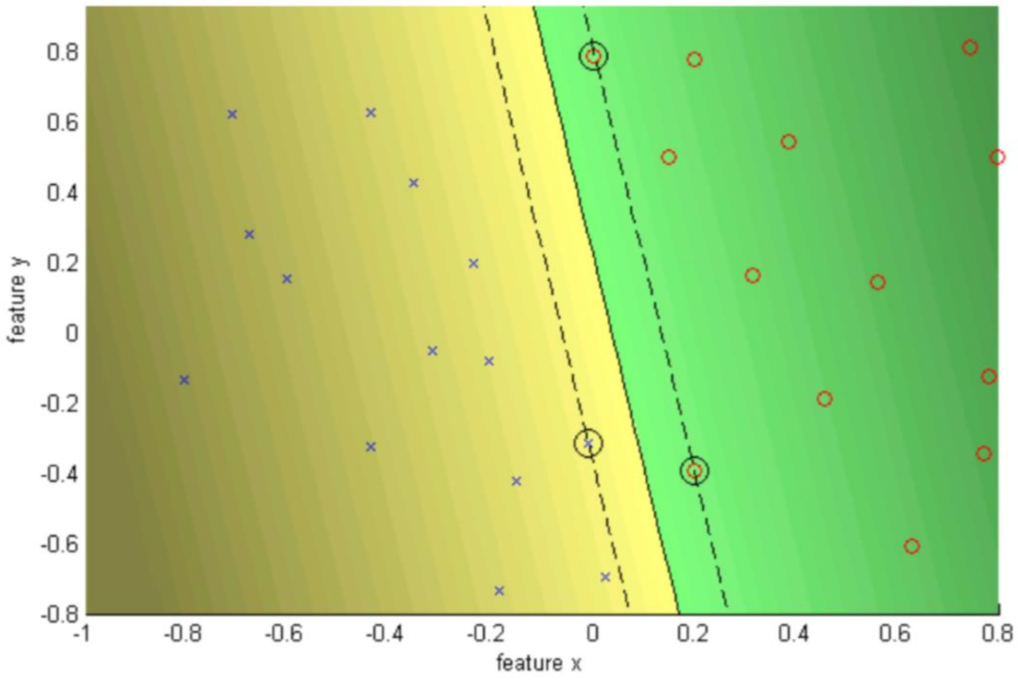
Support Vector Machine

- We allow some **slack** for data points to be on the "wrong" side of the hyperplane in exchange for a more robust hyperplane (against outliers)

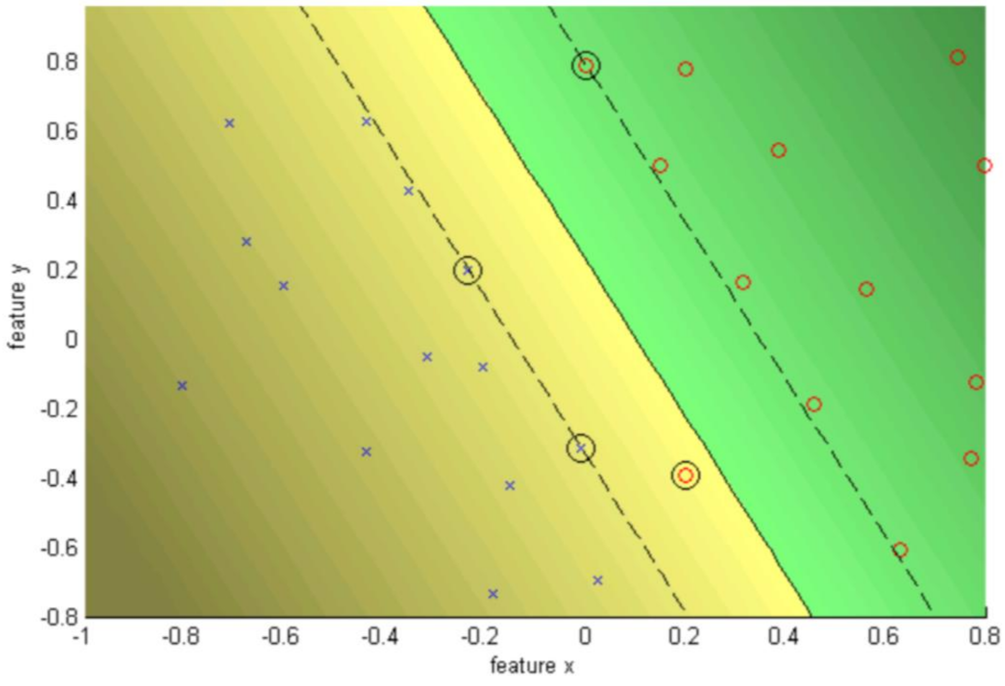


SVM Regularization

- When **C** is small, more slack is allowed, resulting in a softer (but wider) margin



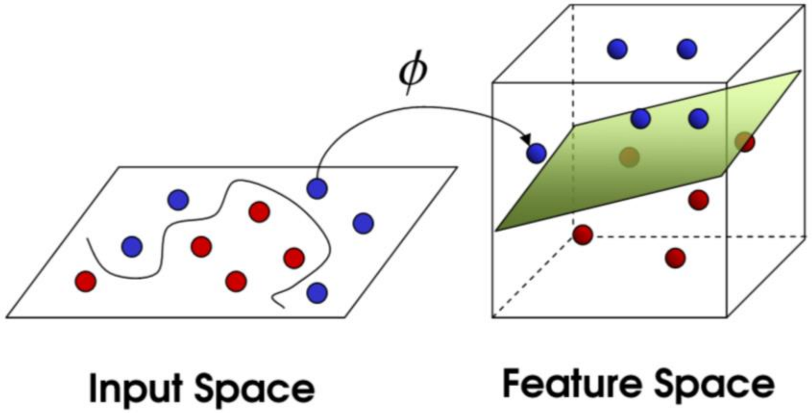
Hard margin
(Large C)

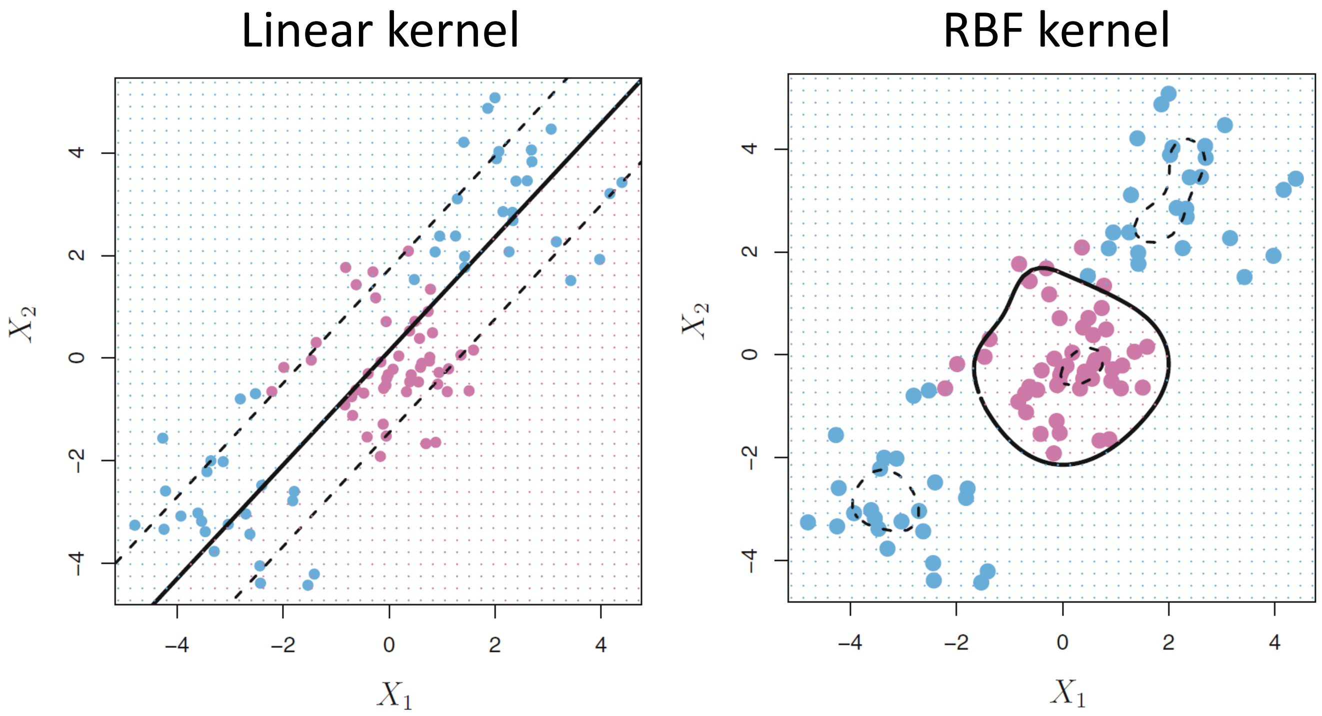


Soft margin
(Small C)

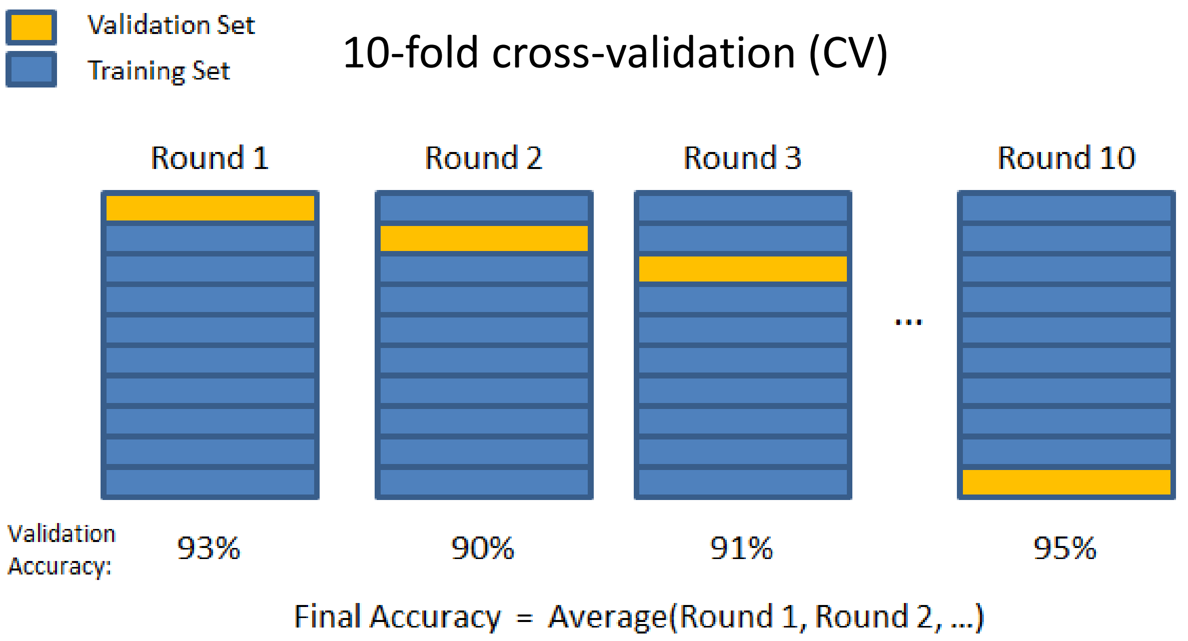
SVM Kernel

- Non-linear kernels allow us to project data points not linearly separable (on the input space) onto a higher-dimensional (feature) space where a linear separating hyperplane can be drawn
- This projection is done by a **kernel function** eg. radial basis function (RBF)
- When projected back onto the input space, the decision boundary is non-linear





k-fold Cross-validation

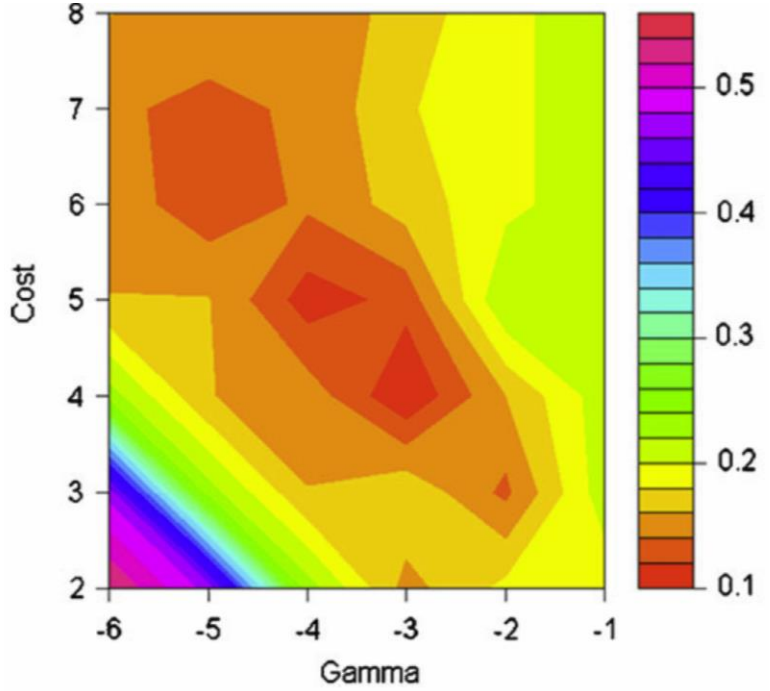


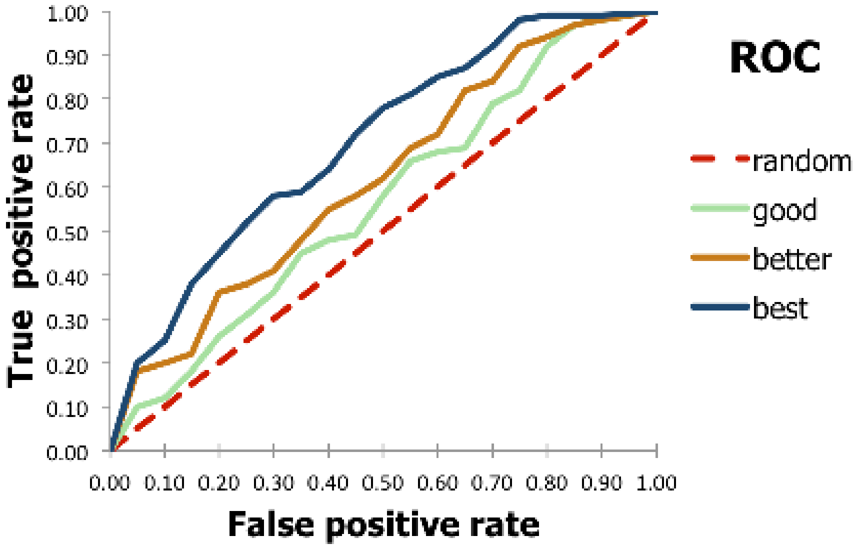
Examples of parameters to optimize:

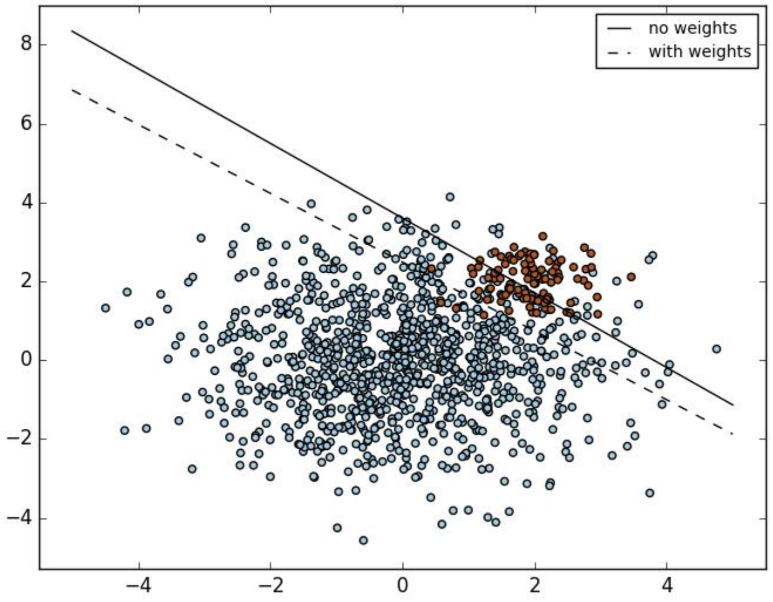
- Random forest
  - Number of trees, B
  - Number of predictors considered at each split, m
  - Max tree depth / min node size
- Support vector machine
  - Penalty / amount of slack tolerated, C
  - Kernel
  - Kernel coefficient, γ

Others

- Grid Search CV, Receiver Operating Characteristic (ROC) curve, class weighting







Sample Code Review

https://github.com/calvinjchiew/kku18 >> ‘sample’ folder >> Lung.ipynb

Hands-on Exercise

https://github.com/calvinjchiew/kku18 >> 'exercise' folder >> Leukemia.ipynb