



NUI Galway  
OÉ Gaillimh

# Machine Learning

## Week 7: Evaluating Classifier Performance; Practical Advice; Some Machine Learning Tools

Prof. Michael Madden

Chair of Computer Science

Head of Machine Learning & Data Mining Group

National University of Ireland Galway



# Learning Objectives

After successfully completing this, you will be able to ...

- Discuss distinctions between training, testing & validation data sets
- Perform cross-validation
- Describe and compute performance measures, learning curves and ROC graphs
- Select and perform appropriate statistical tests
- Give guidance on compiling a data set and selecting & comparing methods for a task
- Discuss related issues, including being able to decide how many test data sets are enough
- Discuss and select machine learning tools





# Why Measure Performance?

How do we determine if hypothesis is sufficiently good?

- Need ways to measure performance
- Need to define acceptable performance for task

We always want to know:

How well will this model work on future data?

- Since we can't look into the future, we have to evaluate the model on data already available to us
- IMPORTANT:  
All forms of evaluation assume that data available to us is representative of future data on which we will apply models

No one ML algorithm is always best

- Need to be able to compare algorithms/variations/tweaks
- Wolpert's "No Free Lunch" theorem





# Basic Performance Measures – Classification

For Classification:

- Basic measure is **Classification Accuracy** ( $1 - \text{Error Rate}$ )
- Proportion of test cases classified correctly
- Multiple choice test: some may be right by chance
- Can easily estimate lower bound on performance:  
**How? (See next slide)**



# Basic Performance Measures – Classification

Test Cases			
Outlook	Temp	Humidity	Windy
sunny	hot	high	false
overcast	hot	high	false
rainy	mild	high	false
overcast	cool	normal	true
sunny	mild	high	false
rainy	mild	normal	false
rainy	mild	normal	false

4/7 predictions correct = accuracy of 57.14% = error rate of 42.86%

In training dataset, 9 of 14 are “yes” class

=> guessing “yes” every time would give accuracy of 64.3%



# Basic Performance Measures – Regression

For Regression:

- **Root Mean Squared Error** (error = Actual - Prediction)
- Square each error, average over test cases, get square root
- Also Maximum Absolute Error, Mean Absolute Error, ...



# Basic Performance Measures – Regression

Test Cases			
Size (m <sup>2</sup> )	# Beds	# Floors	Age (yrs)
195	5	1	40
130	3	2	35
140	3	2	26
80	2	1	30
180	5	2	38



# Confusion Matrix

More informative than Accuracy for multi-class problems

- If some errors are more serious than others, can multiply it by a **mis-classification cost matrix**
- If Category A = Positive ,B = Negative, can see **true positives**, **false negatives**, etc.

		Predicted	
		Category A	Category B
Actual	Category A	$w$	$x$
	Category B	$y$	$z$





# Confusion Matrix - Example

		Actual	
		Positive	Negative
Predicted	Positive	3	1
	Negative	2	1

Actual Class
no
yes
yes
yes
no
yes
yes

Classifier Predicted
no
yes
no
yes
yes
no
no
yes

Classifier Performance
✓
✓
✗
✓
✗
✗
✗
✓



# Confusion Matrix: Other Examples

		Predicted		
		Goat	Horse	Sheep
Actual	Goat	10	0	6
	Horse	0	15	1
	Sheep	5	1	9

Predicted:				Nursery
NOT_RECOM	RECOMMEND	VERY_RECOM	PRIORITY	Actual:
1464	0	0	0	NOT_RECOM
0	16	0	0	RECOMMEND
0	91	1264	50	VERY_RECOM
0	0	143	1292	PRIORITY



# Training, Testing and Validation (1)

What data do we use for testing?

- Performance on training data is not sufficient:

Why not?

If we have **unlimited** data ...

- Randomly sample enough to construct hypothesis
  - too little: bad hypothesis
  - too much: slow!
  - how much? use a **Learning Curve** [coming up soon]
- Randomly sample more to test hypothesis
- If you change hypothesis, re-sample more data

In practice, have **limited** amount of labelled data...

- Need to divide into statistically identical subsets



## Training, Testing and Validation (2)

If we have large amount of labelled data, but not unlimited:

- Randomly sample a **testing set** (a.k.a. holdout set) and **hide it**
- Divide remainder into **training sets** and **validation sets**

Using these datasets:

- **Training data:** used to construct hypotheses
- **Validation data:** used to evaluate/compare hypotheses, while selecting algorithms and parameter settings
- When you're happy you've found the best algorithm and parameter settings you can, construct final hypothesis using **training & validation data** together
- **Finally**, unlock the drawer with the **testing data:** evaluate objectively using it; publish/communicate these results



# Training, Testing and Validation (3)

To be confident train/validation sets are representative

- Often need to re-divide multiple times and average results
- Check standard deviation of results as well

A pitfall of random sub-sampling

- All the train/validation/test sets come from the same source and are not truly independent of each other
- Implications for statistical tests: see later



# Learning Curves

Repeat for various values of  $X$  from 0% to 100%:

- Repeat several times:
  - Train on  $X\%$  of data (randomly sampled)
  - Test on the rest
- Average the results at  $X\%$

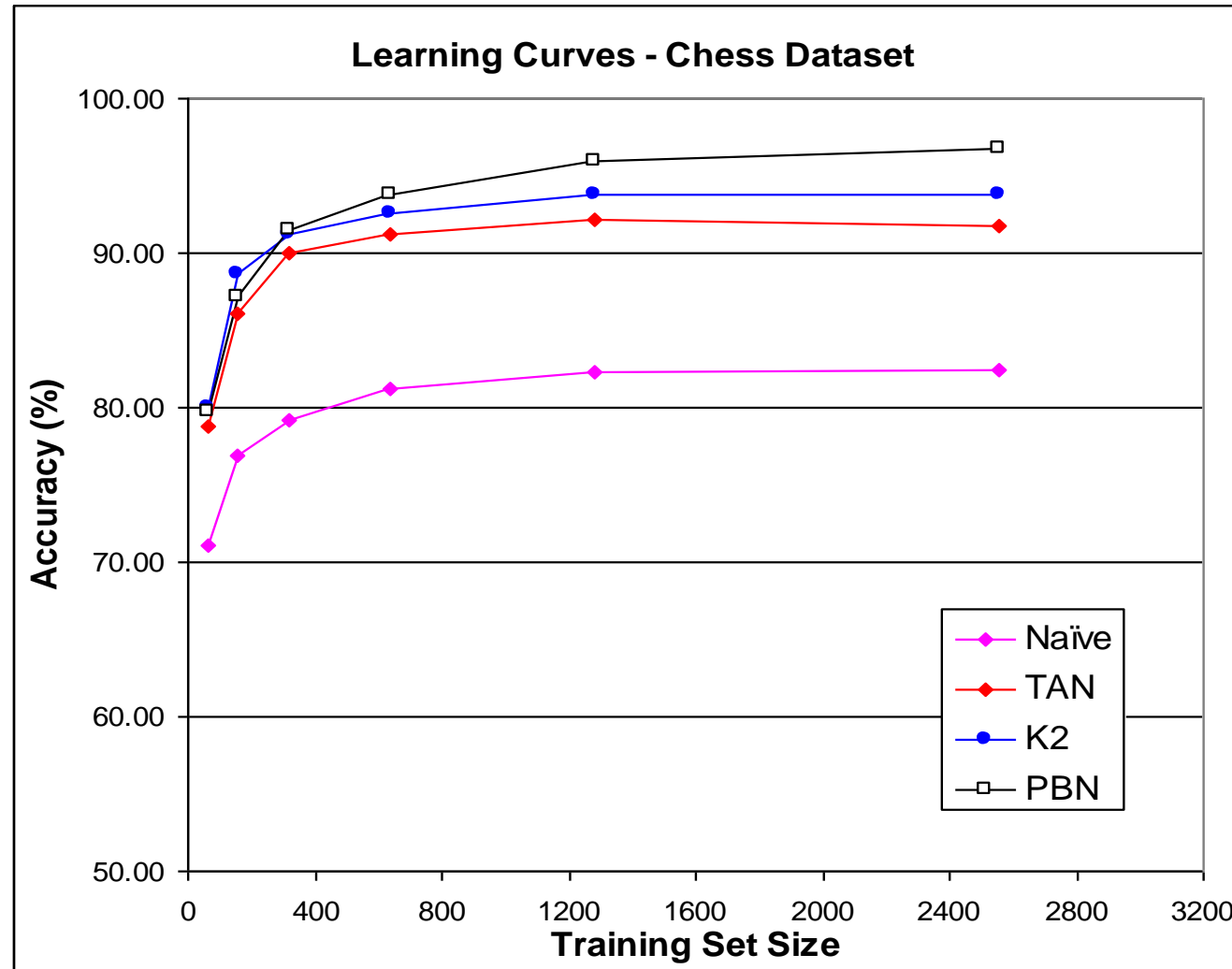
Popular because...

- Useful for comparing techniques
- Insight into how much data is sufficient for a technique
- Indicative of error in the limit





# Learning Curves





# Cross-Validation (1)

Useful for **training/validation** if data is somewhat scarce ...

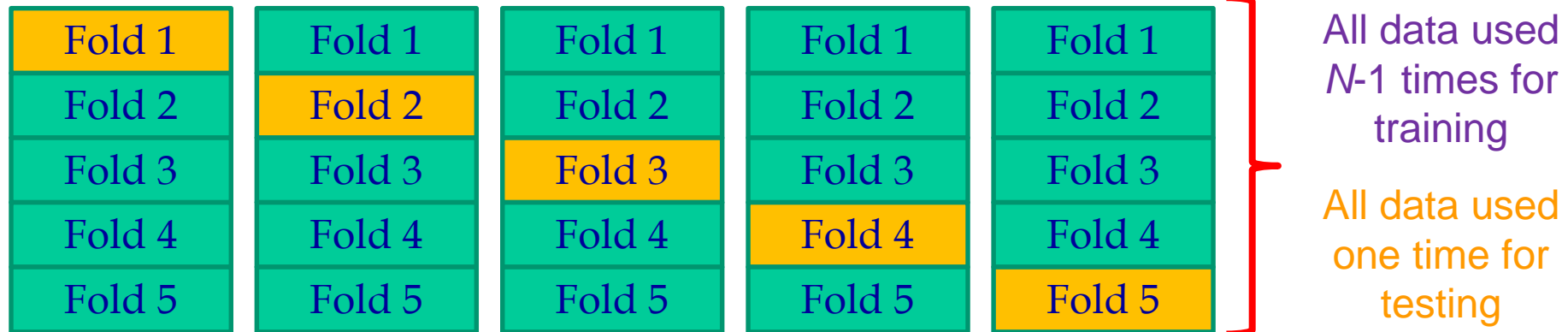
- Divide into  $N$  'folds': subsets of near-equal size
- Often use  $N=10$ : want fair no. of folds, with  $>30$  examples each
- **Stratified**: similar distribution of data in each fold

Repeat for each fold  $i$  from 1 to  $N$ :

- keep fold  $i$  for testing
- construct hypothesis using the rest for training
- test on fold  $i$

Test

Train





## Cross-Validation (2)

### Result:

- Have constructed (and discarded)  $N$  hypotheses, using all data  $N - 1$  times for training
- Have used **all data for testing** once, even though we have never trained and tested on the same data!
- Trade-off: computation vs. data

NOTE: **all** training data used to build a **final** hypothesis

- The cross-validation result constitutes an estimate of its likely future performance
- For 10-CV, 90% overlap between final hypothesis and those in folds, and all training cases have operated as independent test cases
- Still should test this final hypothesis on a **held-out test set**



# Cross-Validation: Variations

## **Repeated N-fold Cross-Validation:**

- Shuffle the data, do cross-validation, shuffle again
- Ensure you have sufficient data that shuffles are meaningful

## **Leave-One-Out Cross-Validation**

- No. of folds = no. of samples
- Not as good statistically, but used with datasets of limited size



# Cross-Validation: Some Caveats

Repeated cross-validation can take a lot of CPU time:

- E.g. do 5 runs of 10-fold cross-validation:  
Each fold involves building a full new model on nearly all data (90%)  
Repeated 5 times with the data shuffled each time
- Result: takes **50 times as long** as if we had done a single train/validation split
- **Of course, even if training times are slow, performing analyses with the new model will still be fast**

Testing on a validation set is quicker, but ...

- Must ensure that validation set is sampled from same population as the training data
- If you don't have very much data, and you use some for a validation set, you will have less for use in model building



# ROC Curves

Particularly useful for comparing probabilistic classifiers

- Classifier outputs relative probability of each class
- Pick the one with highest probability?
- What about if classification is whether/not to do cancer biopsy?

ROC Curve: perform comparisons independent of misclassification costs

- Family of classifiers: one for every possible probability threshold
- Suitable for two-class problems

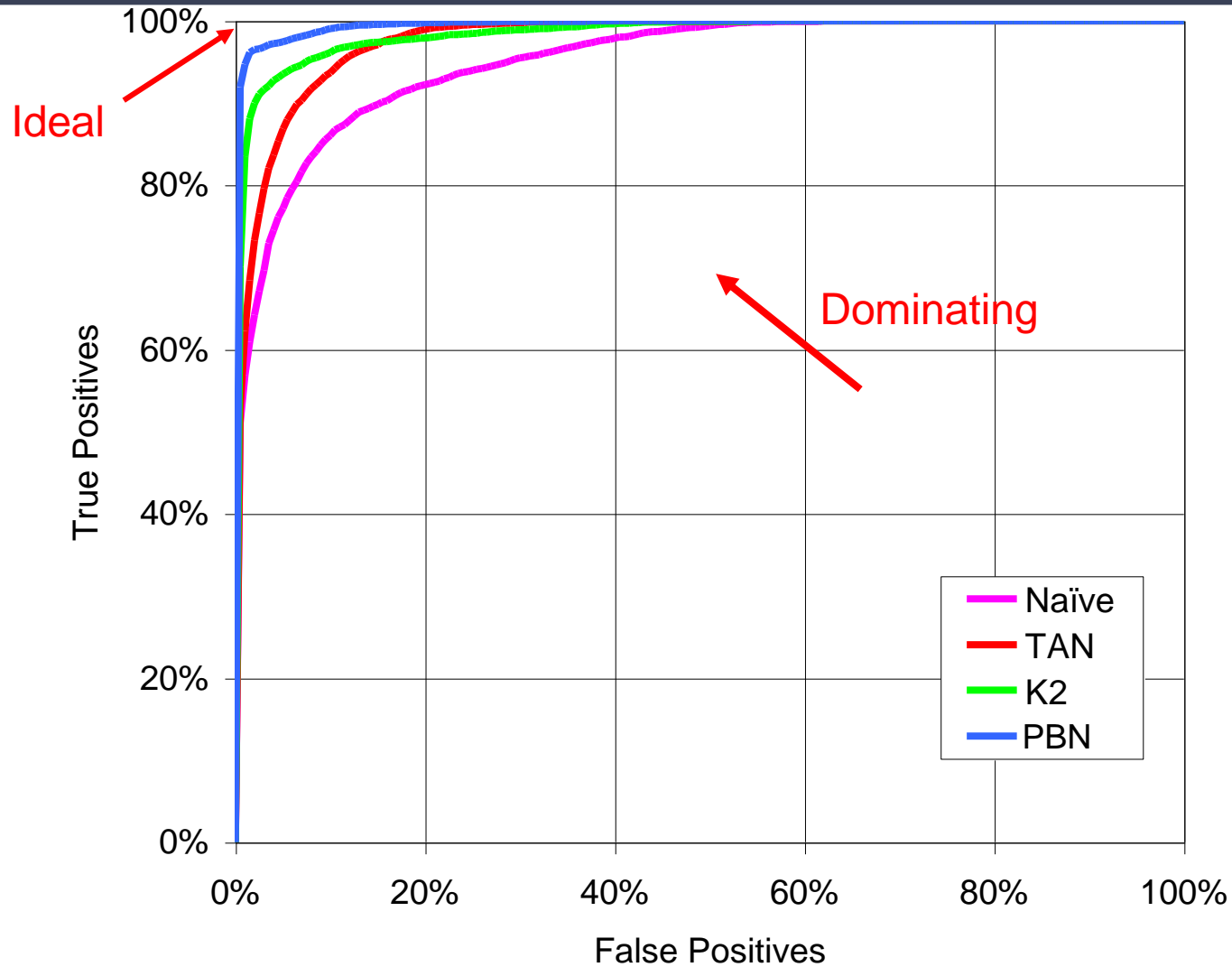
Can also calculate Area Under ROC (AUROC)

- Single-number measure of performance under all possible decision thresholds (multi-class definitions exist)
- Alternative to RMSE





# ROC Curve Example

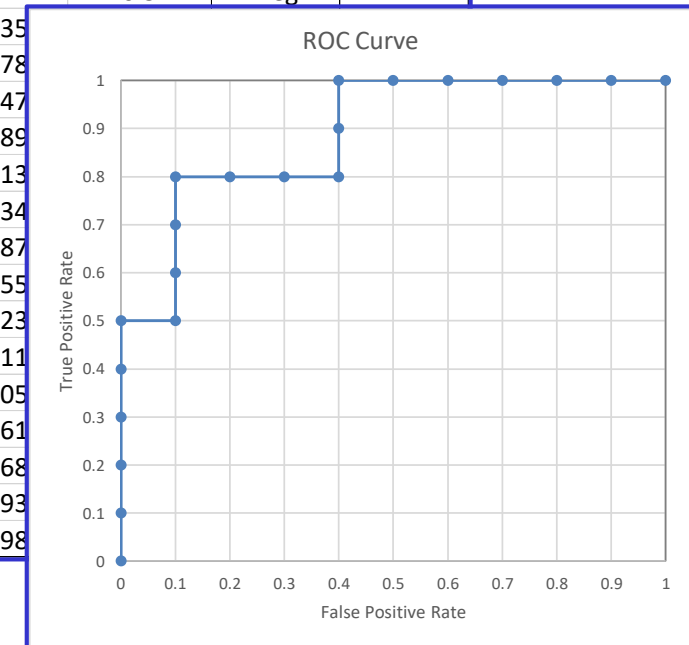




# Tutorial on Computing Confusion Matrices & ROC Curves

- See [ROC-Curve-Tutorial-MMadden.xlsx](#)
- A single ROC curve can have a stepped shape
  - With large amounts of data, steps become small
  - If we average over multiple ROC curves from random subsets of data, steps tend to go away
  - We can also join the curves peak-to-peak: equivalent to replacing points that are dominated by weighted averages of points that dominate them

ID	Actual Class	p(Pos)	Threshold	Predicted Class	Result
A	Neg	0.09	0.5	Neg	TN
B	Neg	0.52	0.5	Pos	FP
C	Neg	0.41	0.5	Neg	TN
D	Pos	0.91	0.5	Pos	TP
E	Neg	0.12	0.5	Neg	TN
F	Pos	0.35	0.5	Neg	FN
G	Neg	0.78	0.5	Neg	TN
H	Neg	0.47	0.5	Neg	TN
I	Pos	0.89	0.5	Pos	TP
J	Neg	0.13	0.5	Neg	TN
K	Pos	0.34	0.5	Neg	FN
L	Pos	0.87	0.5	Pos	TP
M	Pos	0.55	0.5	Pos	TP
N	Neg	0.23	0.5	Neg	TN
O	Neg	0.11	0.5	Neg	TN
P	Neg	0.05	0.5	Neg	TN
Q	Pos	0.61	0.5	Neg	FN
R	Pos	0.68	0.5	Neg	FN
S	Pos	0.93	0.5	Pos	TP
T	Pos	0.98	0.5	Pos	TP





# Comparing two Classifiers/Algorithms (1)

If A has lower error (RMSE, AUROC, ...) than B

- it might be better
- it might have happened by chance
- How can we tell if difference is statistically significant?

## Paired t-test

- To decide whether average results of A and B are same
- **Null Hypothesis**: no statistical difference between results
- **Paired**: use same training/testing data for A and B  
[random seed]

– Calculate **t-statistic**:

– Pick confidence level

– Look up **t-critical** (DOF=k-1)

– See if **t-statistic > t-critical**

$$t = \frac{\bar{d}}{\sqrt{\sigma_d^2 / k}}$$

Annotations for the equation:

- $\bar{d}$ : mean of difference between pairs of measurements
- $\sigma_d^2$ : variance
- $k$ : no. of measurements



## Comparing two Classifiers/Algorithms (2)

A tutorial on how to do this in Excel: see [T-Test.xls](#).

Caveat:

- Paired t-test assumes each individual pair of results is independent of each other pair
- If we did random sub-sampling, this is not true
- High risk of identifying a difference where none exists:  
**Type I Error** [Dietterich 1997]

Modified version of t-statistic to account for this:

- **Corrected Resampled T-Test** [Nadio & Bengio, 2003]
- Also applicable to cross-validation runs



## Comparing two Hypotheses/Algorithms (3)

Should include significance tests when presenting results of accuracy comparisons ...

- Here, figures are accuracy +/- standard deviation
- Statistically best results (in each row) highlighted

	Naïve	TAN	K2
Chess	87.63 ± 1.61	91.68 ± 1.09	94.03 ± 0.87
Breast Cancer	<b>97.81 ± 0.51</b>	<b>97.47 ± 0.68</b>	<b>97.17 ± 1.05</b>
LED-24	<b>73.28 ± 0.70</b>	<b>73.18 ± 0.63</b>	<b>73.14 ± 0.73</b>
DNA Splice	94.80 ± 0.44	94.75 ± 0.42	<b>96.22 ± 0.64</b>
Lymphography	83.60 ± 9.82	<b>85.47 ± 9.49</b>	81.47 ± 10.37
Nursery	90.48 ± 0.41	<b>94.16 ± 0.33</b>	92.63 ± 0.67
SPECT	71.70 ± 6.56	<b>81.25 ± 4.78</b>	<b>80.19 ± 4.66</b>
TicTacToe	70.69 ± 1.94	75.08 ± 1.86	74.04 ± 3.51



# Comparing Algorithms: How many Test Data Sets are Enough?

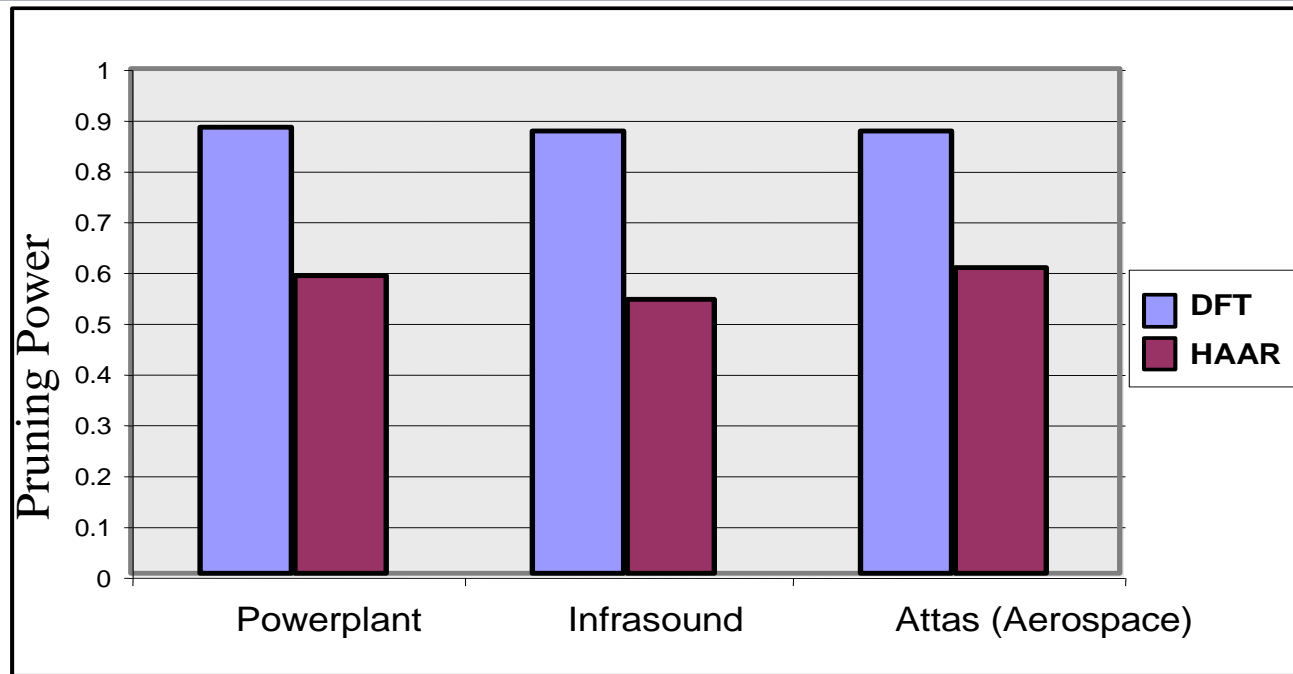
Essential point:

- If we want to determine whether one technique is on average better than another, we typically use several datasets to test them
- If you only use a few different datasets, you can ‘show’ whatever you want!

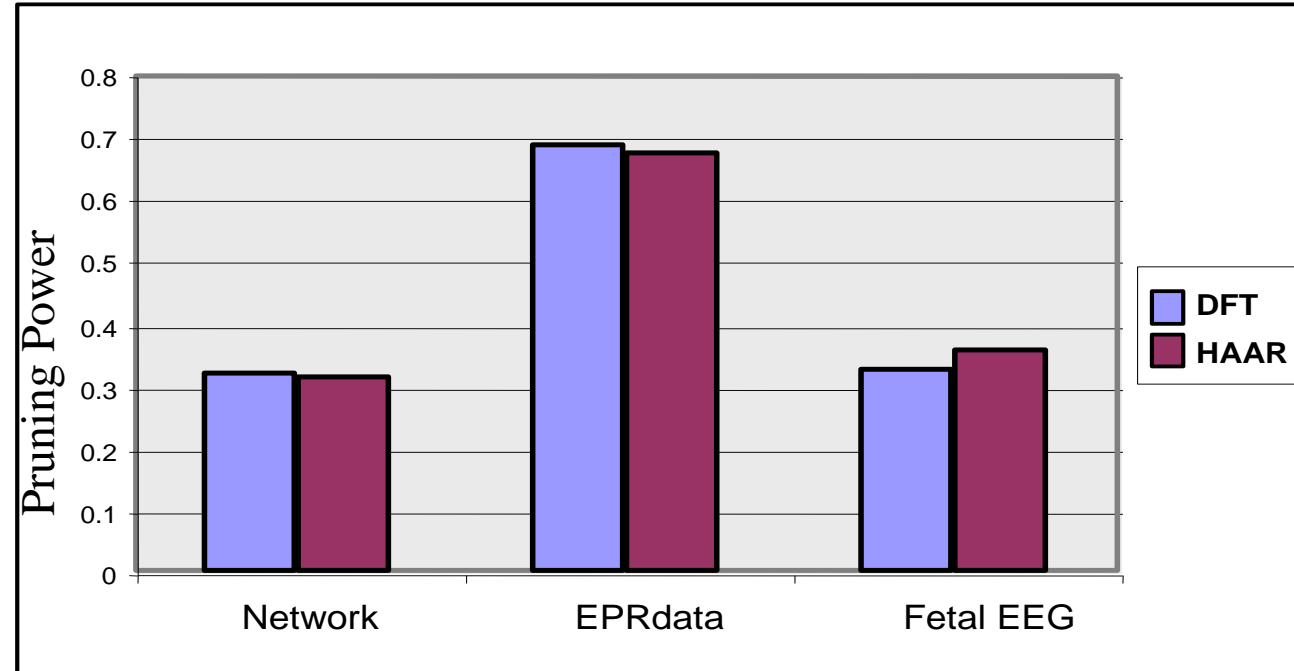
Reference:

- “On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration”, by Keogh and Kasetty, *Data Mining and Knowledge Discovery*, Oct. 2003.
- Paper concerned with time series techniques, but point is generally applicable
- More details: see tutorial from 14<sup>th</sup> ECML conference, at <http://www.cs.ucr.edu/~eamonn/>

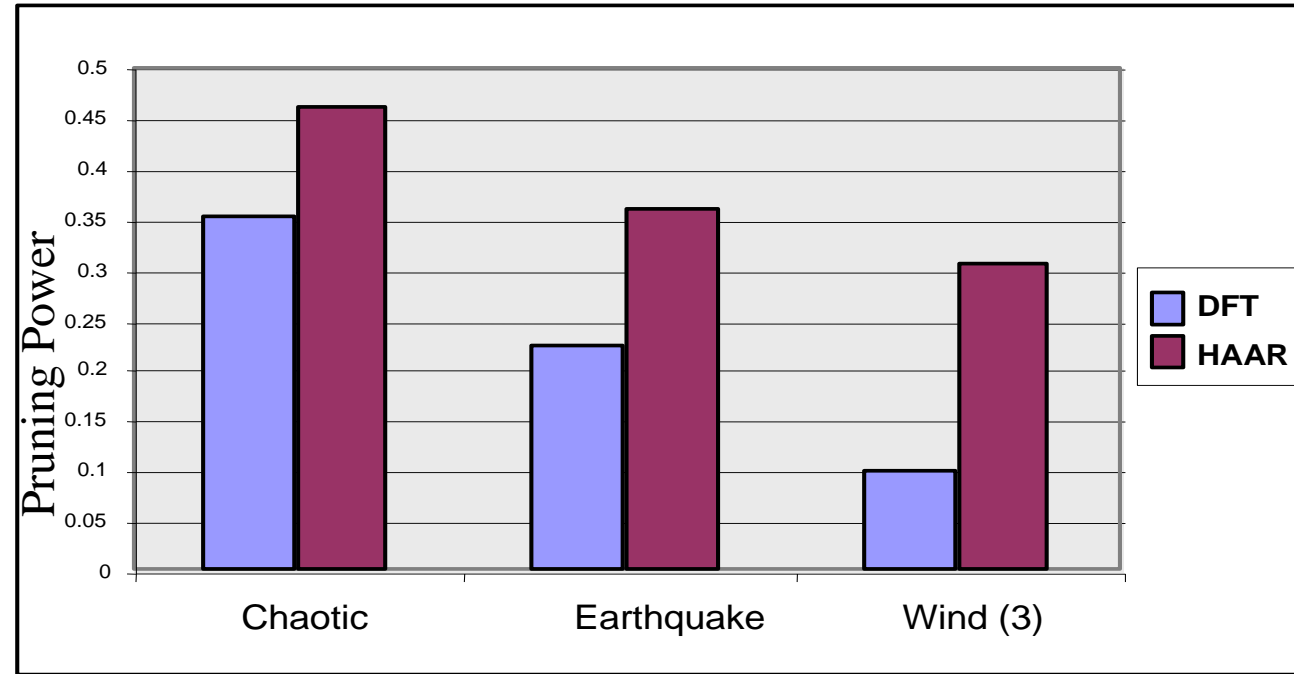




These tests show that DFT is much better than HAAR



These tests show that DFT and HAAR perform similarly



These tests show that DFT is much worse than HAAR



# How to Improve Performance of ML Models

There are several strategies for generating improved ML models with lower error rates

In order of increasing difficulty ...

1. Adjust the parameters of the chosen ML algorithm
2. Select a different ML algorithm to build a new model
3. Use appropriate pre-processing on the data set
4. Improve the training data set; begin by examining any cases that are persistently misclassified



# A Systematic Model Development Procedure (1)

From your dataset, select some as a hold-out set

- Ideally, these should be samples that do not occur in exactly the same form elsewhere in the dataset
- Lock this away!

Using the rest of your data, select a pre-processing method, a model-building method, and settings for it

- Test it on the training data to see if it's promising
- If it is, test it using repeated cross-validation

Repeat the previous step as necessary, until you have found a combination that works for your data

Only then, unlock your hold-out set and test on it



# A Systematic Model Development Procedure (2)

As you follow this procedure, always good practice to keep notes of what you have tried and what results you got

- Learn from these about what methods are good for your domain

Bear in mind that you may need to adjust your dataset

- There may be gaps, inadequacies or errors in it
- Watch out for systematic errors, independent of analysis method

Other issues to watch out for:

- As you adjust a setting of a model-building method, you should find that it ultimately plateaus or starts to get worse: then you know you've gone far enough
- If there is a big divergence between performance on training set and on independent data, it's a sign of **overfitting**





# A Systematic Model Development Procedure (3)

There are no hard-and-fast rules for which algorithm will work well for your data

- Different datasets have different properties
- Different algorithms make different assumptions, that are either well suited or poorly suited to the particular dataset
- Wolpert's “No Free Lunch Theorem”, mentioned earlier

Algorithms that reflect general properties of your application domain tend to work well

- Choose or design algorithms based on experience of the domain to make them “close to the problem”, so that they work better than generic statistical methods



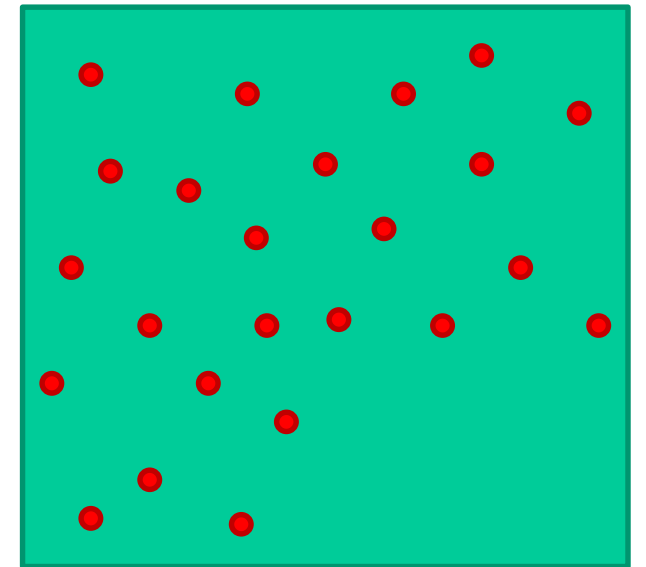
# Training Dataset Considerations (1)

- A training dataset consists of:
  - A set of cases, with the same quantities measured for each
  - For each one, the correct answer to the question to be answered: this is **ground truth**: essential for both constructing and evaluating the model
- This data will be used to build a model that generalizes and interpolates from the data
  - For new, unknown cases, make best estimate of correct answer



## Training Dataset Considerations (2)

- Training data must cover an appropriate range of samples & situations
  - Model-building methods assume that the training data is drawn from the same space as future data on which it will be applied
  - Model evaluation methods are only meaningful with this assumption
- Consider how the model will be used
  - What data collection conditions will apply?
  - Will there be variations, noise, etc?
  - What kinds of data will be presented to the model?
- Aim to get good **coverage** of the space of data on which the model will subsequently be used





# Training Dataset: Coverage for Classification

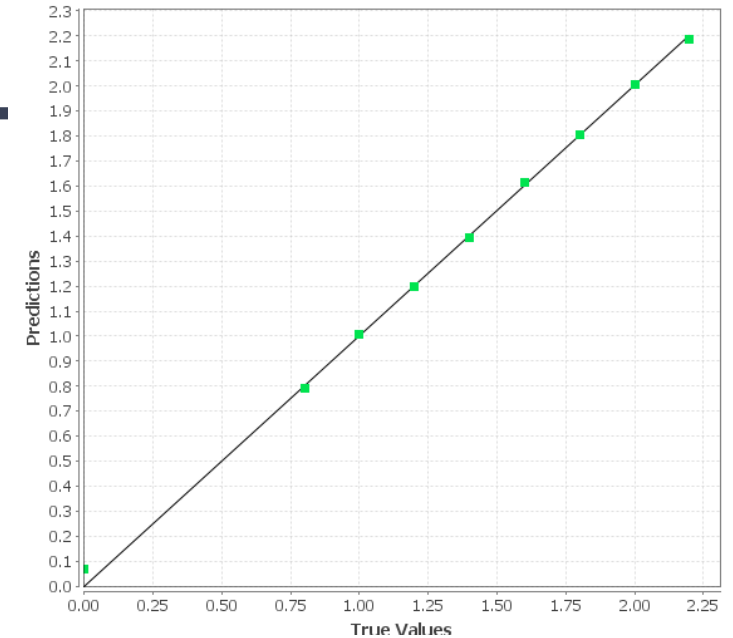
- Example: building a model to detect cocaine in wide variety of cutting agents
- Need a representative sample of cutting agents
- Need **positive and negative** samples
- **Positive:** Pure cocaine; cocaine in a variety of cutting agents
- **Negative:** Cutting agents without cocaine; other drugs
- For classification model, do **not** need exact concentration in each sample, as long as you are certain it is present/absent in each
- Do **not** need every possible combination of cocaine plus all possible cutting agents, but **do** need enough for the model to be able reliably **generalize** from the data





# Training Dataset: Coverage for Regression

- E.g: building a model for concentration of a solvent, for a process monitoring  
The solution is tightly controlled  
Solvent should be between 10% and 15%  
Want to estimate it to within 0.1%
- Need samples covering a good range of concentrations, focusing between 10% and 15%  
Not much point in having lots of samples in the range 20%-90%
- Some pure samples and some 0% are useful to include
- To estimate concentration to 0.1%, must have at least that accuracy in ground truth measurements  
Variations of that amount must manifest themselves in the data!





# Guidelines for Choosing Algorithms

In what order should you try different algorithms?

- Try simpler algorithms before more complex ones
- Occam's Razor: Reduce risk of **overfitting**

Try simpler settings before more complex ones

- E.g. Linear SVM before Quadratic or Cubic

Be aware of the effects of pre-processing:

- Can help or harm
- E.g. normalisation can destroy quantitative information


No amount of algorithmics can fix bad data

- Wrongly labelled, instrument errors, dominated by noise, containing unknown constituents, etc.
- Not representative of the problem
- Not a good sampling of the problem space



# Machine Learning is an Iterative Experimental Process

- Try different algorithms that might be relevant to the application; modify existing algorithms if needed
- Experiment with different settings
- Figure out whether pre-processing might help
- Understand the effects and assumptions of methods
- Be familiar with your data
- If you see systematic errors in models, check whether they arise from the data
- Be patient, be thorough, question everything!



*Practical  
machine  
learning  
advice*



# Some (FOSS) ML Tools

Weka: Command line or GUI; API in Java

The screenshot displays three overlapping windows from the Weka machine learning software:

- Weka GUI Chooser:** The main interface for selecting tools. It features a menu bar (Program, Visualization, Tools, Help), a logo of a kiwi bird, and the text "WEKA The University of Waikato". It includes buttons for "Applications", "Explorer", "Experiment", "KnowledgeFlow", and "Simple CLI".
- Weka Explorer:** A window for data exploration and classification. It has tabs for "Preprocess", "Classify", "Cluster", "Associate", "Select attributes", and "Visualize". The "Classify" tab is active, showing the "Classifier" as "J48 -C 0.25 -M 2". It includes "Test options" (Use training set, Supplied test set, Cross-validation, Percentage split) and a "Result list" showing several files, with "10:06:35 - trees.J48" selected.
- Weka Classifier Tree Visualizer:** A window titled "Weka Classifier Tree Visualizer: 10:06:35 - trees.J48 (weather.symb...)". It displays a decision tree structure for the "trees.J48" model. The tree starts with the root node "outlook", which branches into "sunny", "overcast", and "rainy". The "overcast" branch leads to a leaf node "yes (4.0)". The "sunny" branch leads to a node "humidity", which branches into "high" (leaf "no (3.0)") and "normal" (leaf "yes (2.0)"). The "rainy" branch leads to a node "windy", which branches into "TRUE" (leaf "no (2.0)") and "FALSE" (leaf "yes (3.0)").

Below the tree, the "Classifier output" section shows statistics: "Correctly Classified", "Incorrectly Classified", "Kappa statistic", "Mean absolute error", "Root mean square error", "Relative absolute error", "Root relative standard error", and "Total Number of Instances". It also displays a "Weighted Avg." and a "Confusion Matrix".

```
==== Detailed ====
```

Weighted Avg.	1	0	1	1	1	1
==== Confusion Matrix ====						
a b <-- classified as						
9 0   a = yes						
0 5   b = no						

The status bar at the bottom shows "Status OK" and a "Log" button.





# Some (FOSS) ML Tools

## Apache Mahout:

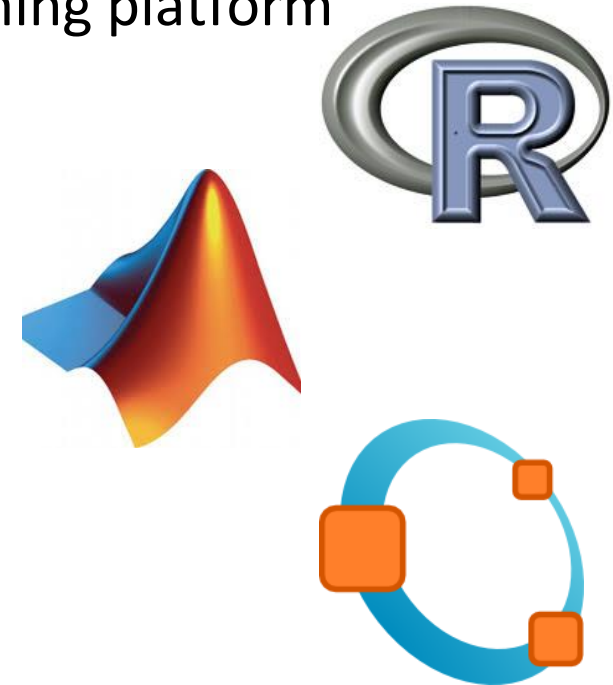
- Scala algorithms implementations built on Spark
- Java implementations of algorithms that are distributed (built on Hadoop) or otherwise scalable: older ones built with MapReduce
- Collaborative filtering, clustering and classification





# Other Tools Often Used

- R
  - General-purpose mathematical programming platform
  - CRAN has good collection ML packages
- Matlab & Mathematica
  - Neither are FOSS! Quite expensive
- Octave
  - Open-source partial clone of Matlab
- Your favourite programming language
  - Lisp
  - Java
  - C++
  - Python

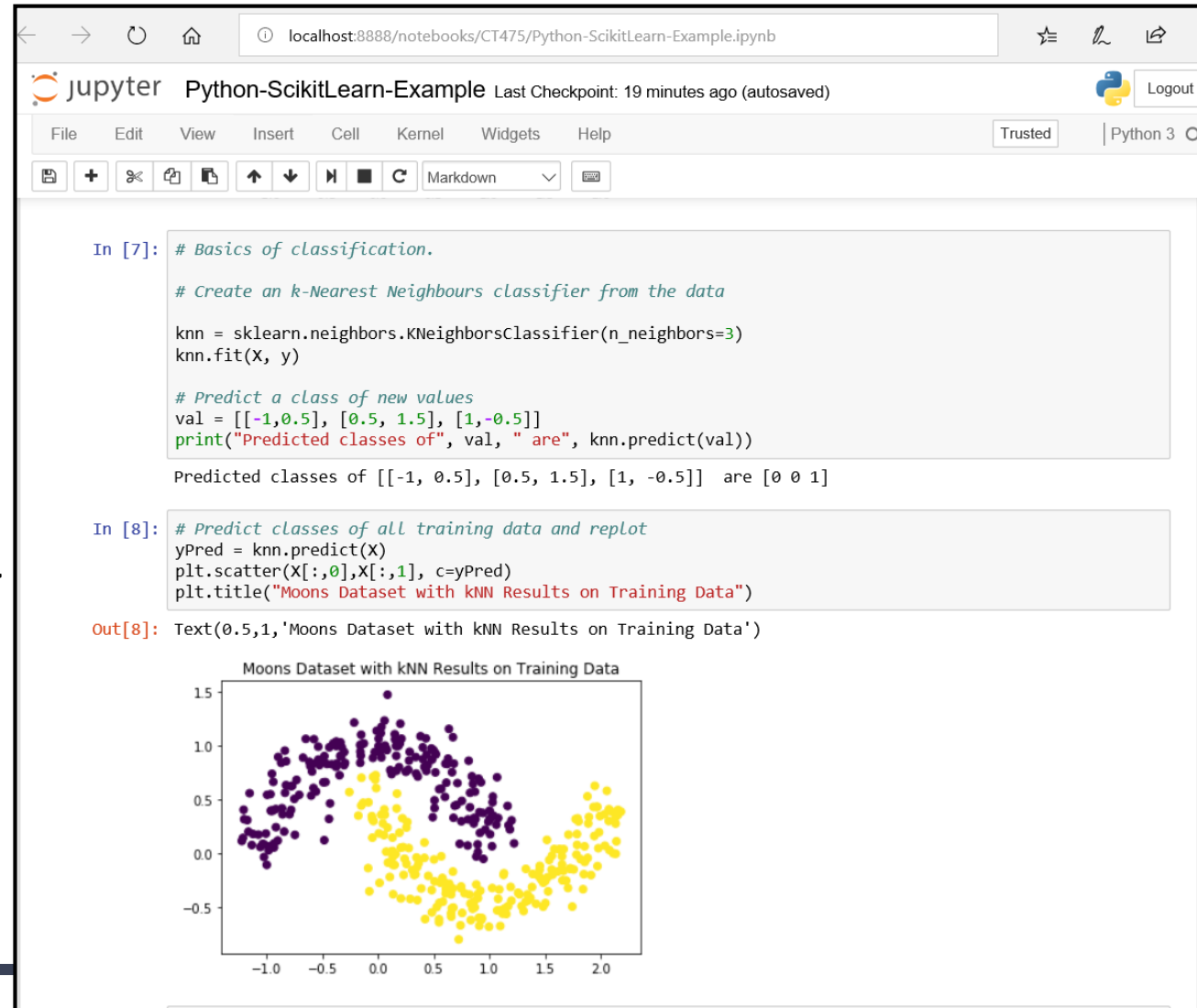




# Other Tools Often Used

## Jupyter Notebook with scikit-learn

- Interactive Python with rich-text web interface
- Other languages also supported
- Keep code, notes & outputs together
- Useful Packages: **scikit-learn** (ML); uses NumPy, SciPy & Matplotlib





# Learning Objectives - Review

Now that you completed this topic successfully, you can ...

- Discuss distinctions between training, testing & validation data sets
- Perform cross-validation
- Describe and compute performance measures, learning curves and ROC graphs
- Select and perform appropriate statistical tests
- Give guidance on compiling a data set and selecting & comparing methods for a task
- Discuss related issues, including being able to decide how many test data sets are enough
- Discuss and select machine learning tools