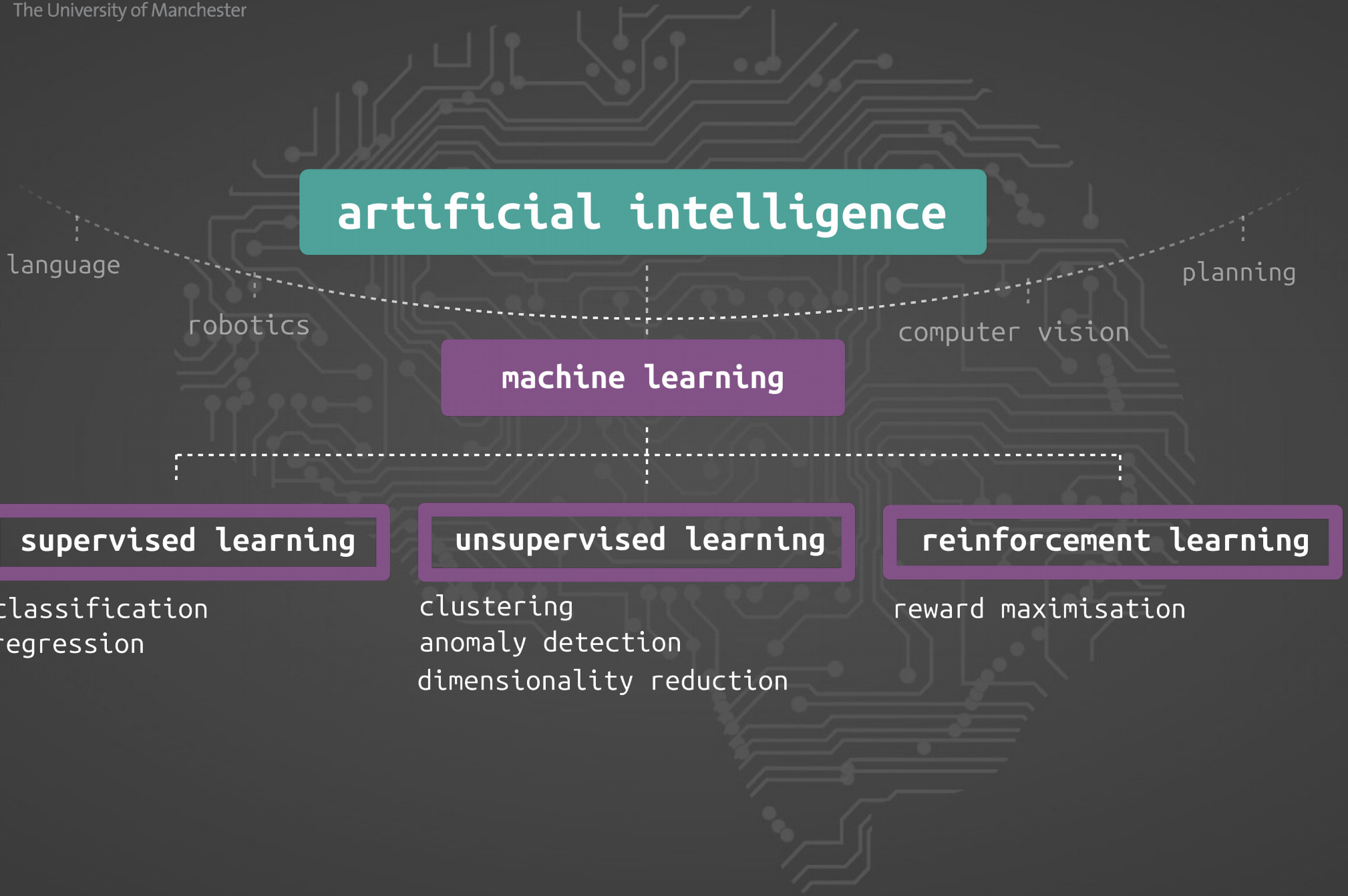
MANCHESTER 1824
The University of Manchester

JACS

Machine learning: best practice

Philippa Hartley

Jodrell Bank Centre for Astrophysics


artificial intelligence
language
robotics
machine learning
computer vision
planning
supervised learning
unsupervised learning
reinforcement learning
classification
regression
clustering
anomaly detection
dimensionality reduction
reward maximisation

MANCHESTER 1824
The University of Manchester

Machine learning terminology

Sample X

A single item to process: image, document, data row etc

Label y

True category or value assigned to each sample, if known

"apple"

Feature

Set of attributes associated with each sample

red
round

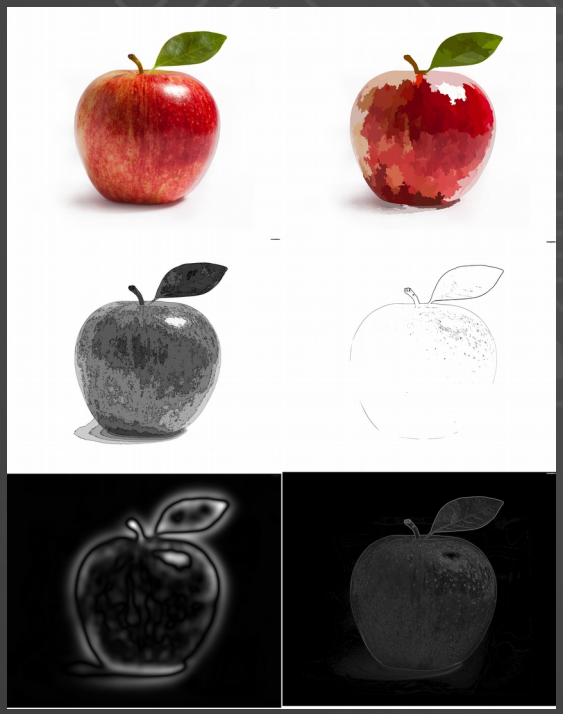
Model

Output obtained after training which can be used to make predictions about new data

f(x)

Feature extraction

engine size	colour	height	width	weight
0.9	red	1.7	1.2	900
2.2	red	2.4	1.4	1800
1.1	black	1.7	1.1	1200
1.8	white	1.6	1.2	1600
1.6	blue	1.6	1.2	1600
3.0	yellow	1.2	1.5	1100



n features

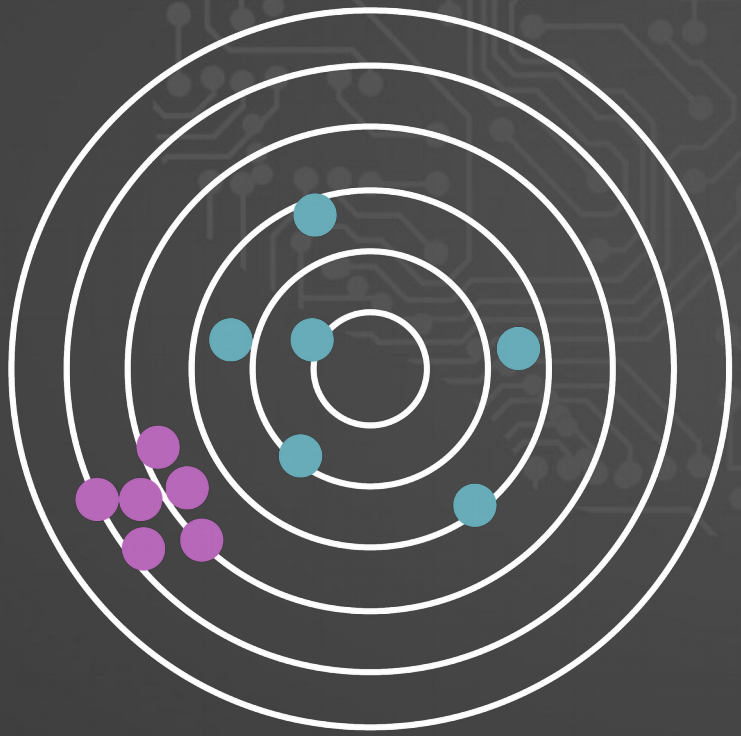
m samples

$$\begin{bmatrix} X_{00} & \cdots & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \cdots & X_{mn} \end{bmatrix} \begin{bmatrix} y_0 \\ \vdots \\ y_m \end{bmatrix}$$

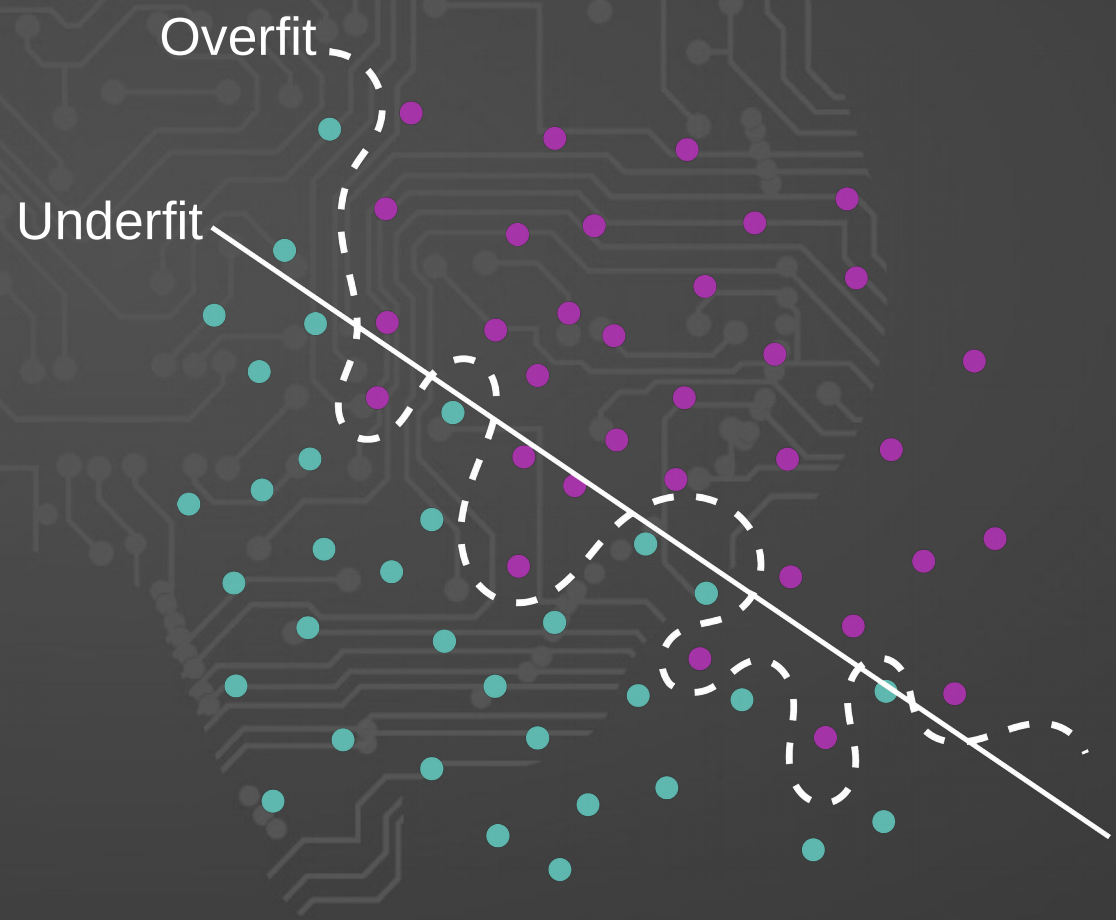
Python API reference: *http://scikit-learn.org/stable/modules/feature_extraction.html*
Python image processing library: *http://scikit-image.org/*

Feature selection

We would like the model to generalise well for use with new data



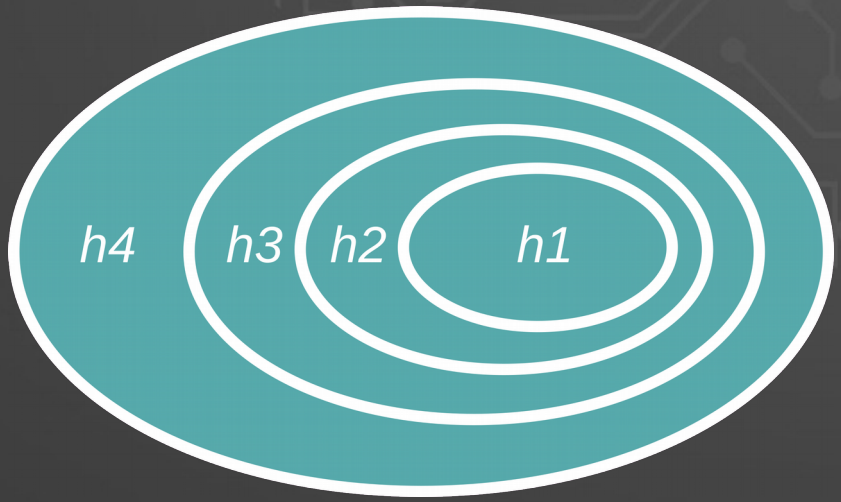
Bias vs variance tradeoff



Feature selection

We can evaluate the following bound:

test error = train error + *f(N,h,p)*



Nested subsets of functions
ordered by complexity

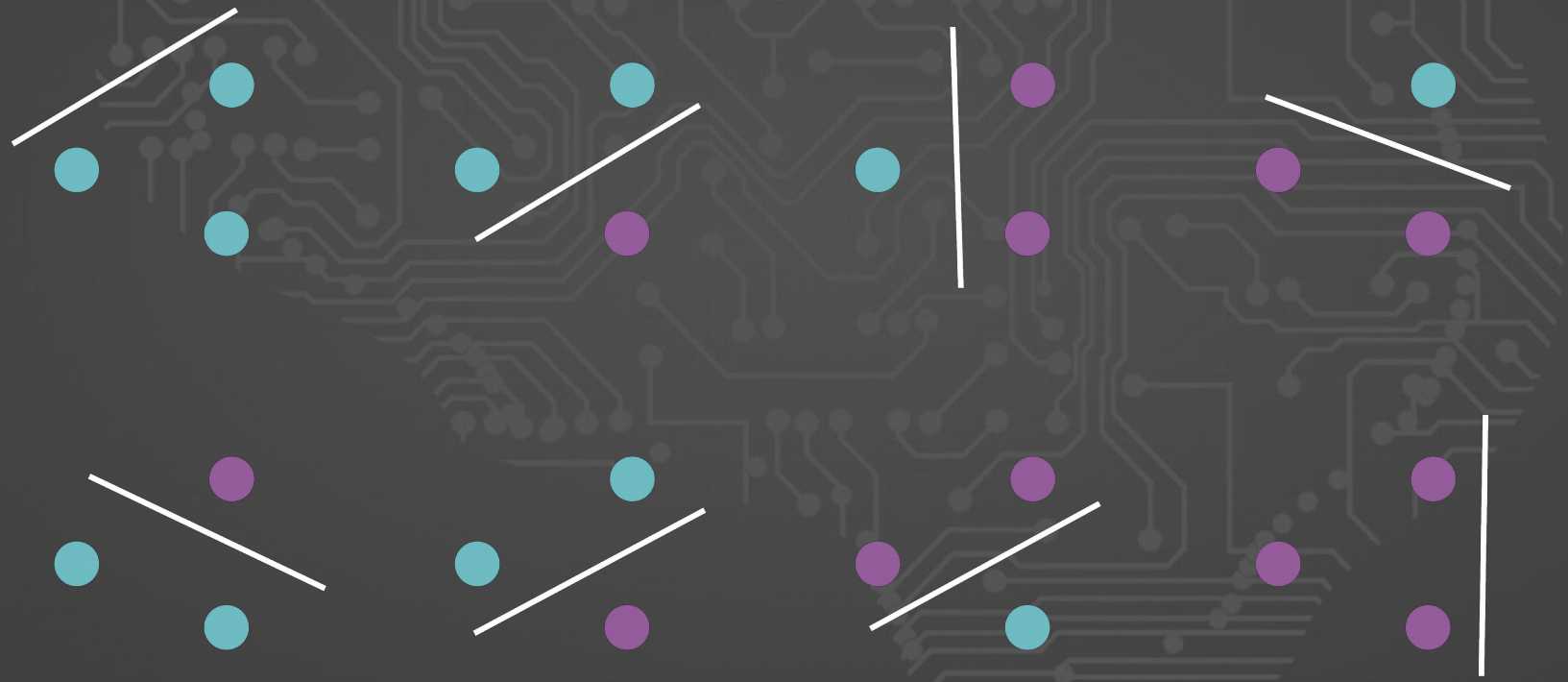
N = number of training samples

h = model complexity

p = probability of the bound failing

Feature selection

VC dimension: A measure of model complexity for classification methods



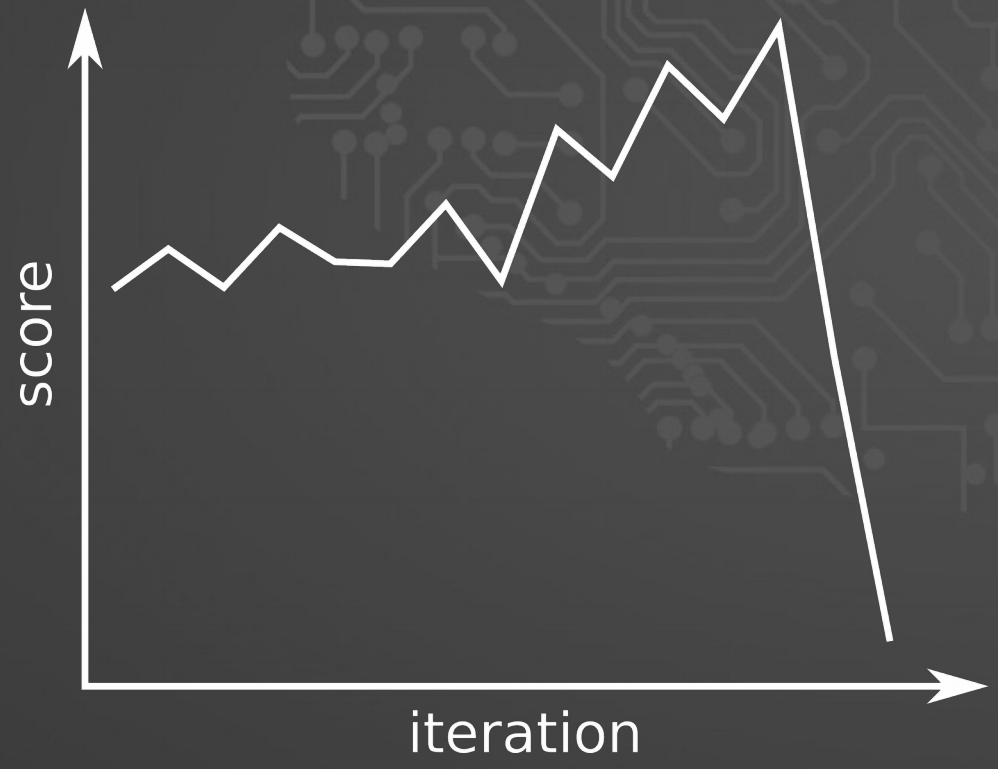
Example: two features/dimensions
three training samples

Burgess 1998: A Tutorial on Support Vector Machines for Pattern Recognition

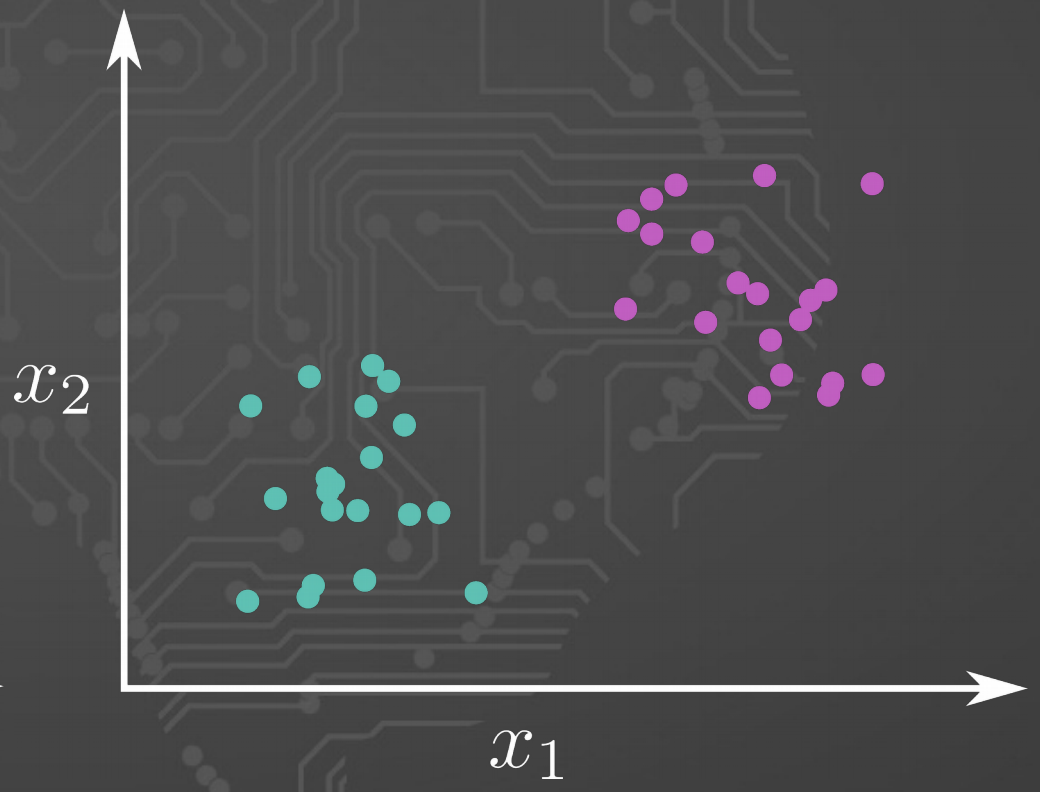
MANCHESTER 1824
The University of Manchester

Feature selection

Recursive feature selection reduces overfitting and finds optimal number of features



Principle component analysis can be used to reduce features and remove redundancies



Python API: *http://scikit-learn.org/stable/modules/feature_selection.html*
http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

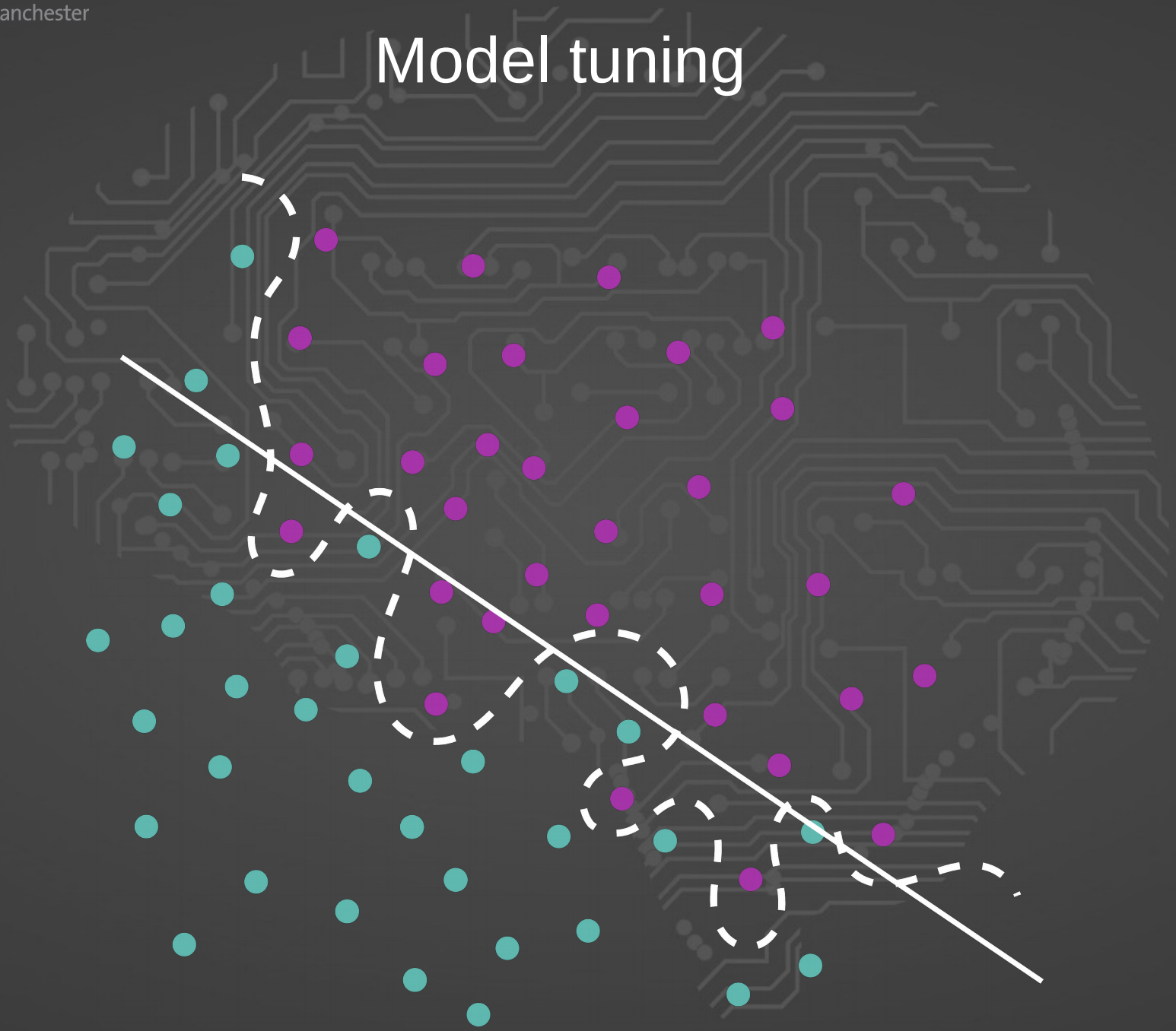
Cross validation

k-fold cross validation splits training set into *k* smaller sets

Average scores from 1-fold test set are calculated

Python API reference: *http://scikit-learn.org/stable/modules/cross_validation.html*

Model tuning

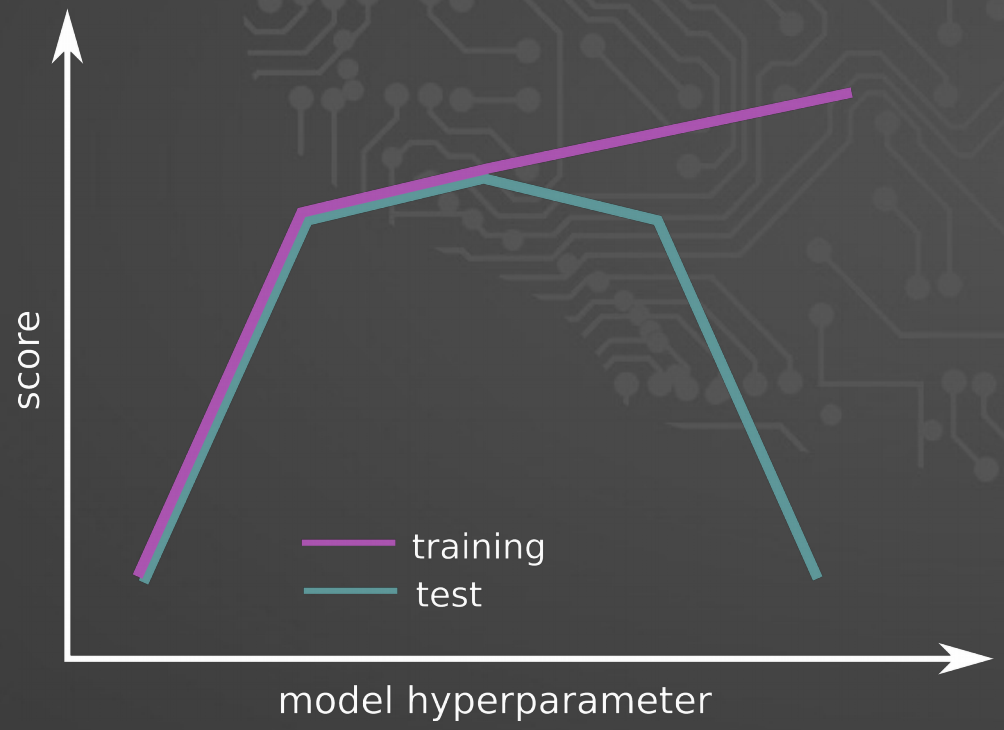


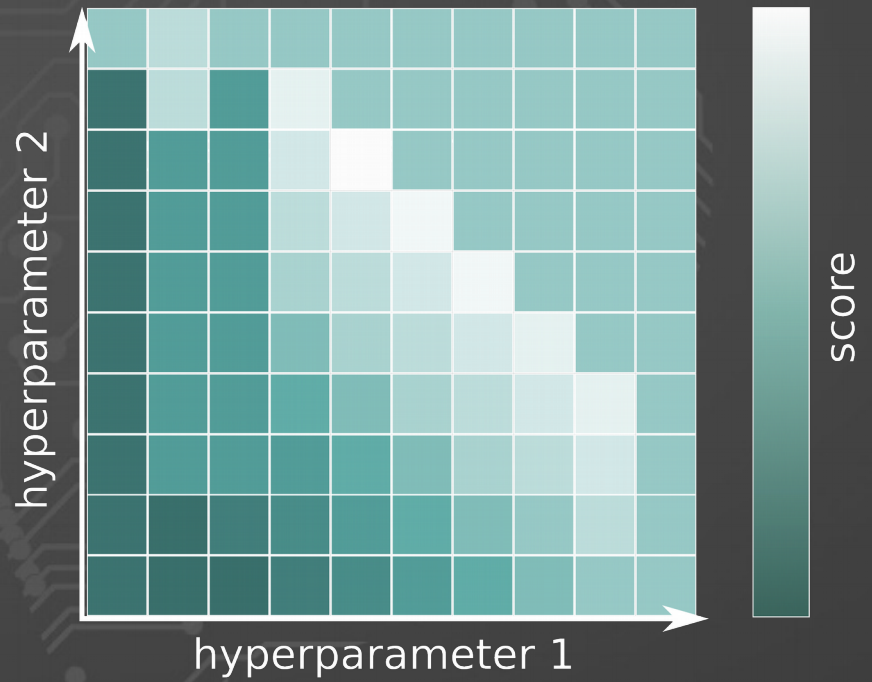
MANCHESTER 1824
The University of Manchester

Model tuning

Models often contain **regularisation hyperparameters** which can be optimised to prevent overfitting

e.g. SVM kernel, Gaussian process kernel, k neighbours in k-NN, complexity C in regression





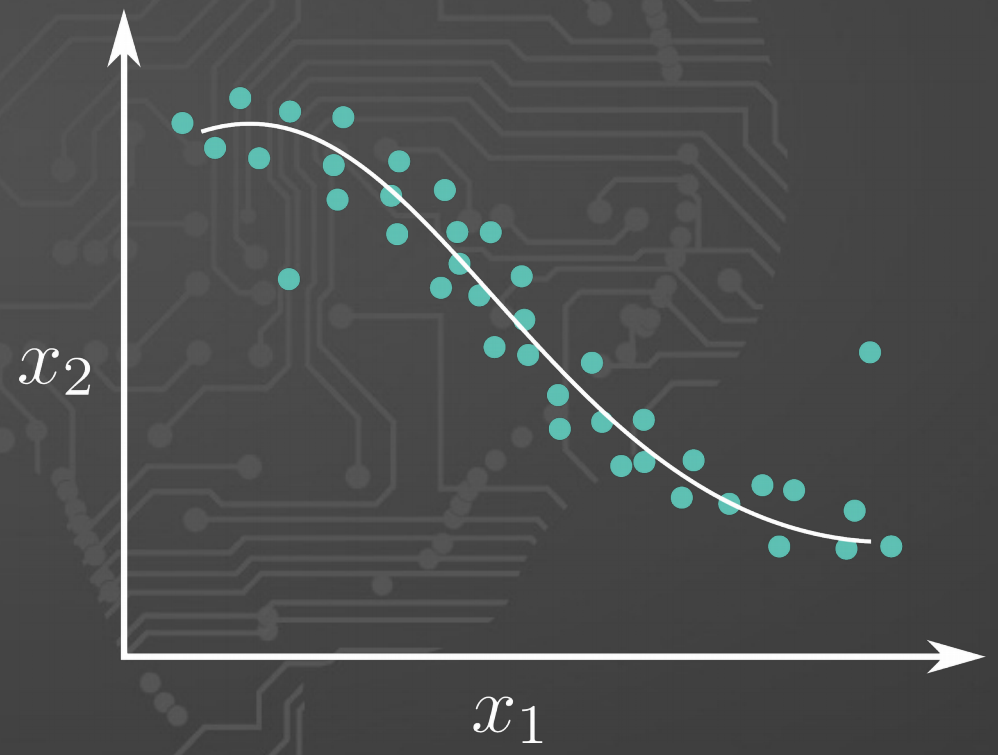
Python API: *http://scikit-learn.org/stable/modules/learning_curve.html*
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Metrics: regression

Evaluate performance in order to choose best models

Mean absolute error (MAE) or

Root mean squared error (RMSE)



Python API reference: *http://scikit-learn.org/stable/modules/model_evaluation.html*

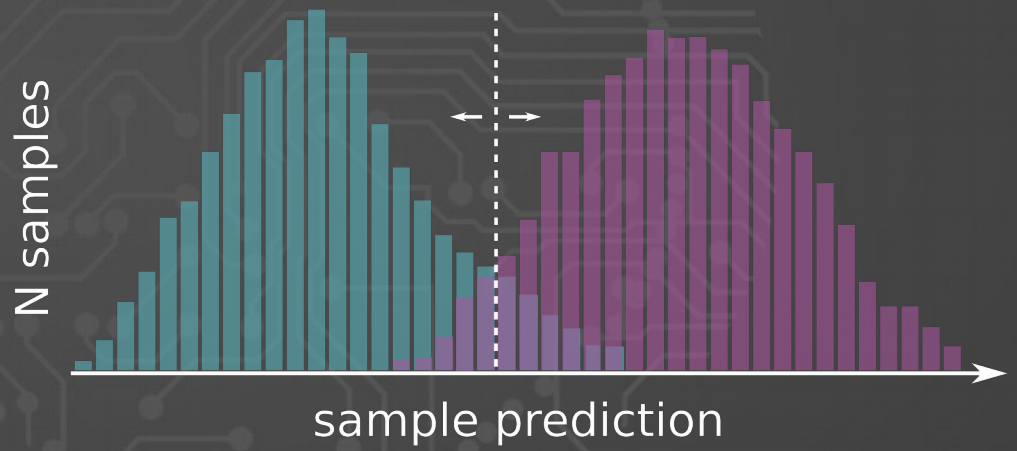
MANCHESTER 1824
The University of Manchester

Metrics: classification

A **confusion matrix** can be used to visualise different types of errors

		True condition	
	All samples	Condition positive	Condition negative
Predicted condition	Predicted condition positive	True positive (**TP**)	False positive (**FP**)
	Predicted condition negative	False negative (**FN**)	True negative (**TN**)
		True positive rate **(recall)** **TPR = TP/TP+FN**	False positive rate **(fall-out)** **FPR = FP/FP+TN**
		False negative rate **(miss rate)** **FNR = FN/TP+FN**	True negative rate **(specificity)** **TNR = TN/FP+TN**

Accuracy = TP+TN / all samples



Python API reference: *http://scikit-learn.org/stable/modules/model_evaluation.html*

MANCHESTER 1824
The University of Manchester

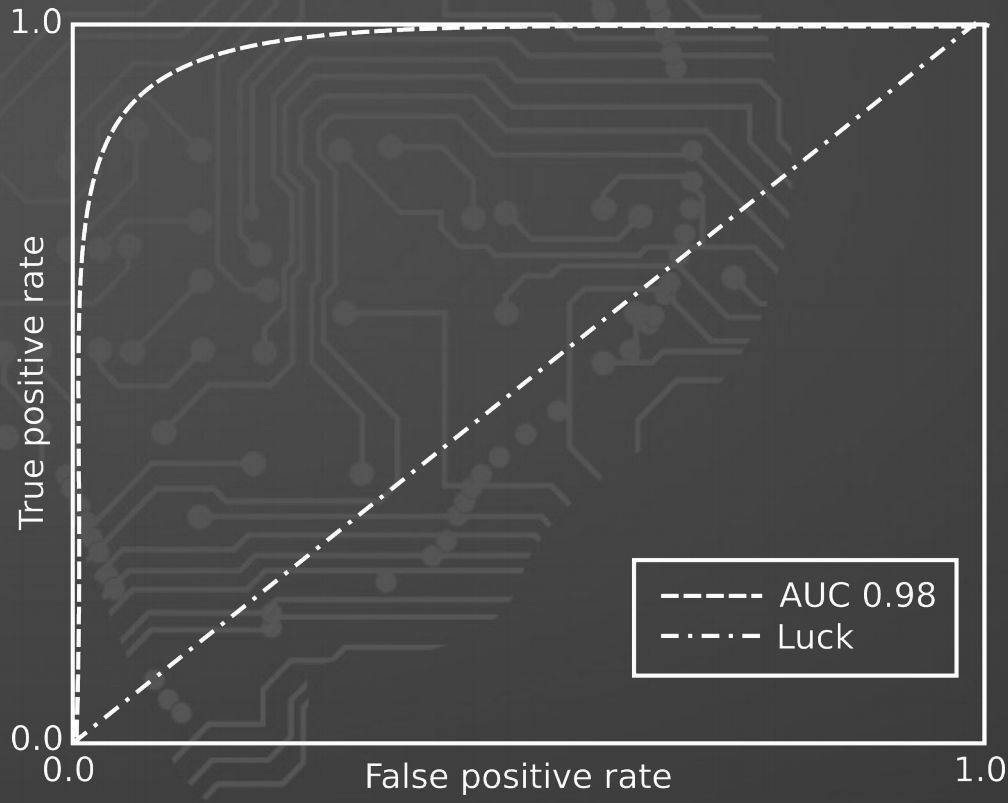
Metrics: classification

A **confusion matrix** can be used to visualise different types of errors

		True condition	
	All samples	Condition positive	Condition negative
Predicted condition	Predicted condition positive	True positive (**TP**)	False positive (**FP**)
	Predicted condition negative	False negative (**FN**)	True negative (**TN**)
		True positive rate **(recall)** **TPR = TP/TP+FN**	False positive rate **(fall-out)** **FPR = FP/FP+TN**
		False negative rate **(miss rate)** **FNR = FN/TP+FN**	True negative rate **(specificity)** **TNR = TN/FP+TN**

A **Receiver Operating Characteristic (ROC)** curve shows the trade-off between recall and fall-out and is insenstive to class distribution

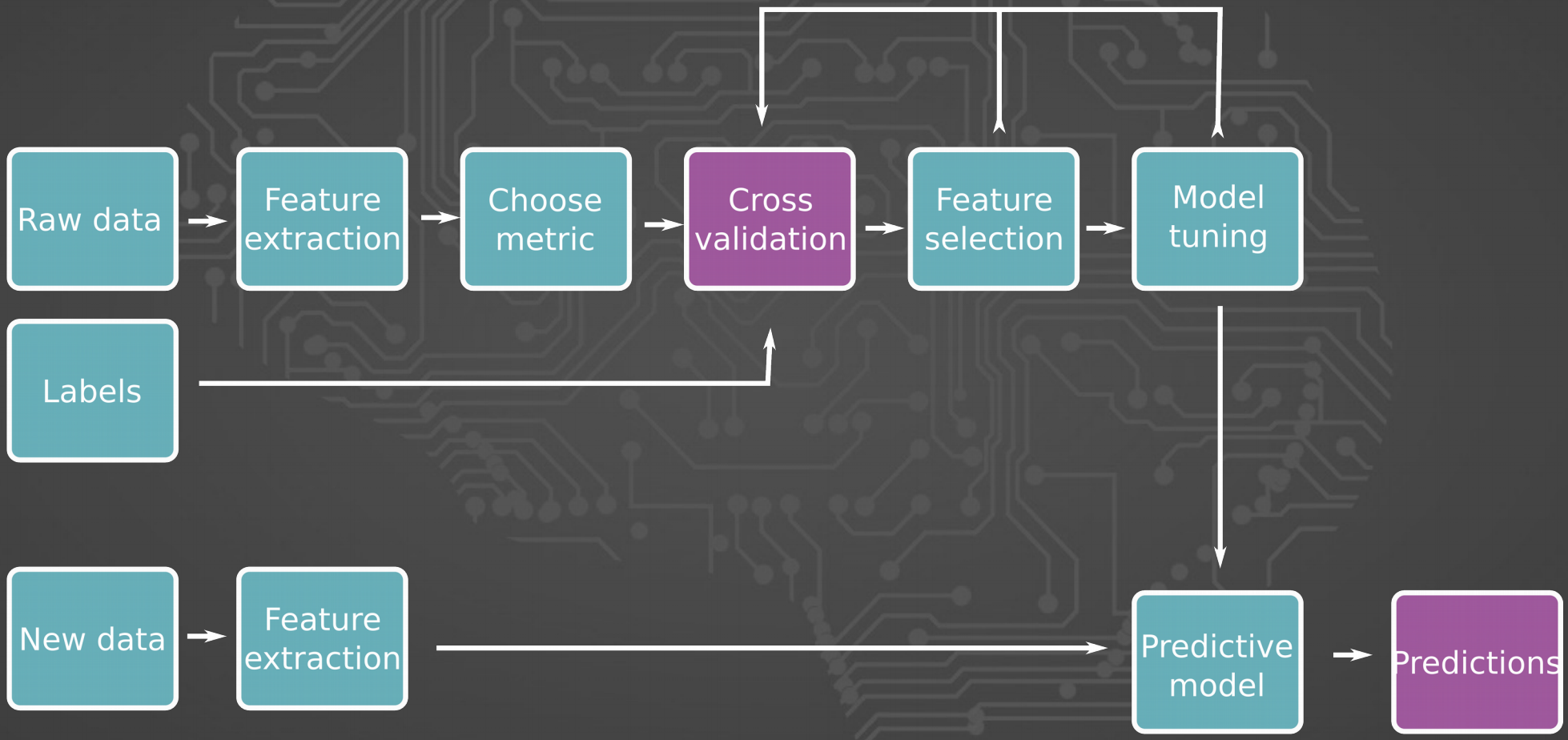
The A**rea Under Curve (AUC)** gives one overall measure of performance



Python API reference: *http://scikit-learn.org/stable/modules/model_evaluation.html*

MANCHESTER 1824
The University of Manchester

Model persistence



Python API:*http://scikit-learn.org/stable/modules/model_persistence.html*