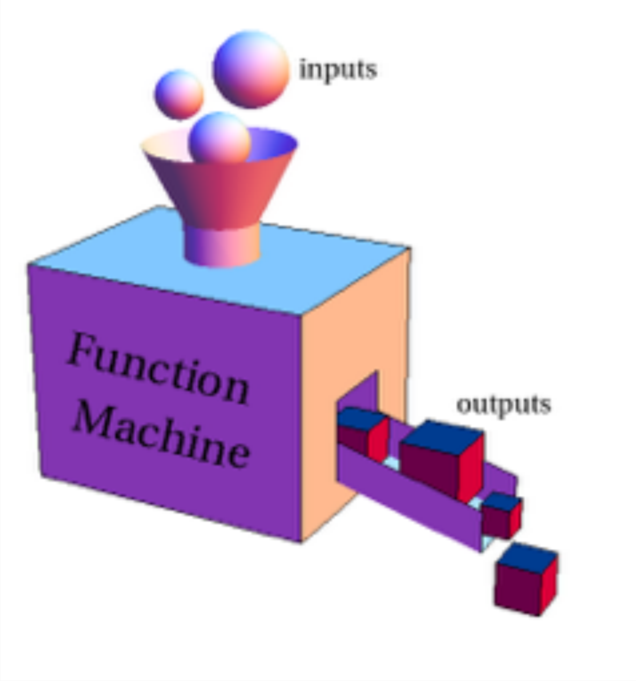
# An Introduction to Machine Learning


**Luisa Lucie-Smith**

Credit: Michelle Lochner

# What is machine learning?

- In machine learning, statistical learning techniques are applied to identify patterns in data.
- These techniques can be used to make highly accurate predictions.
- Different algorithms use different prescriptions for building the model



# When to use machine learning

*Unsupervised* learning – clustering problems

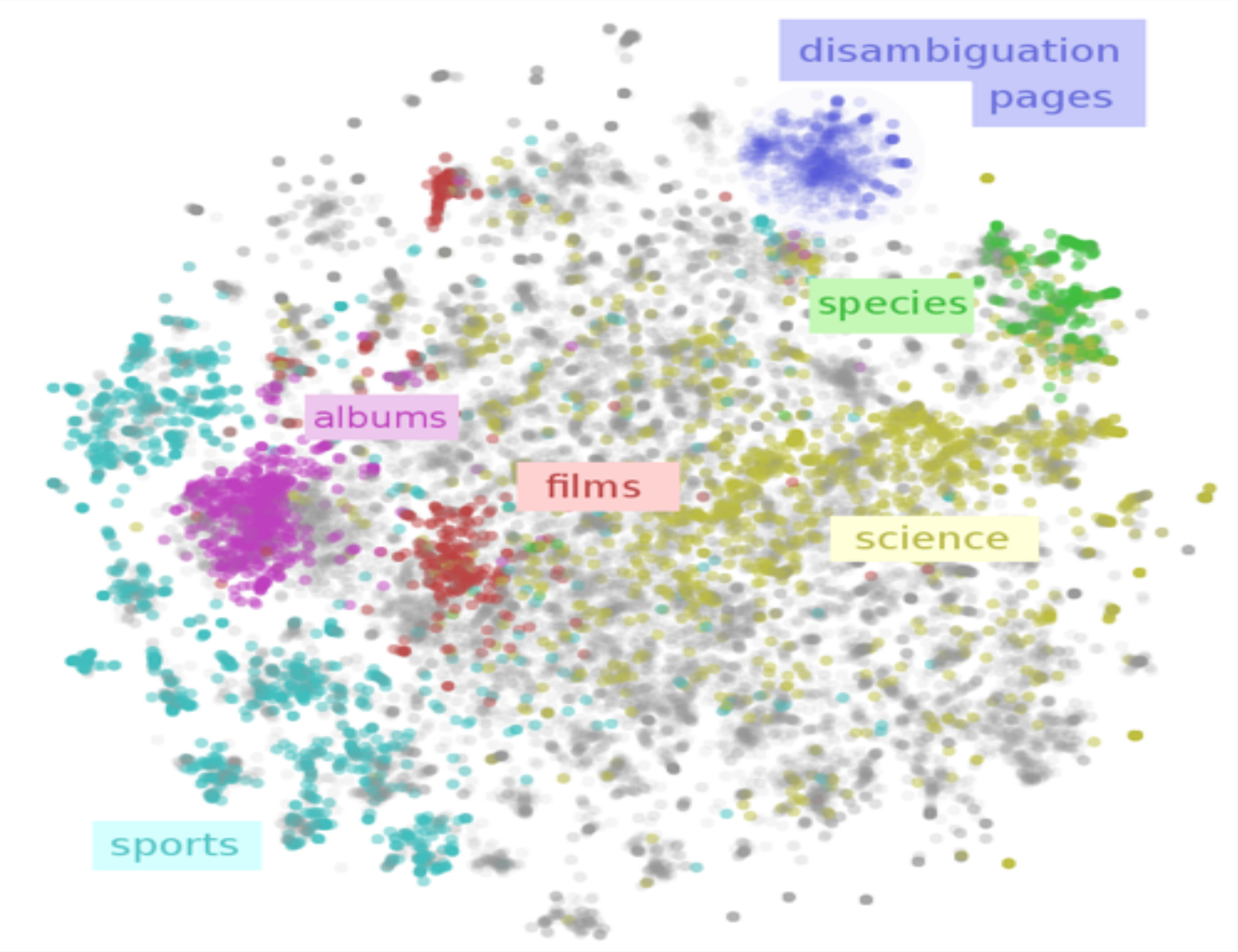
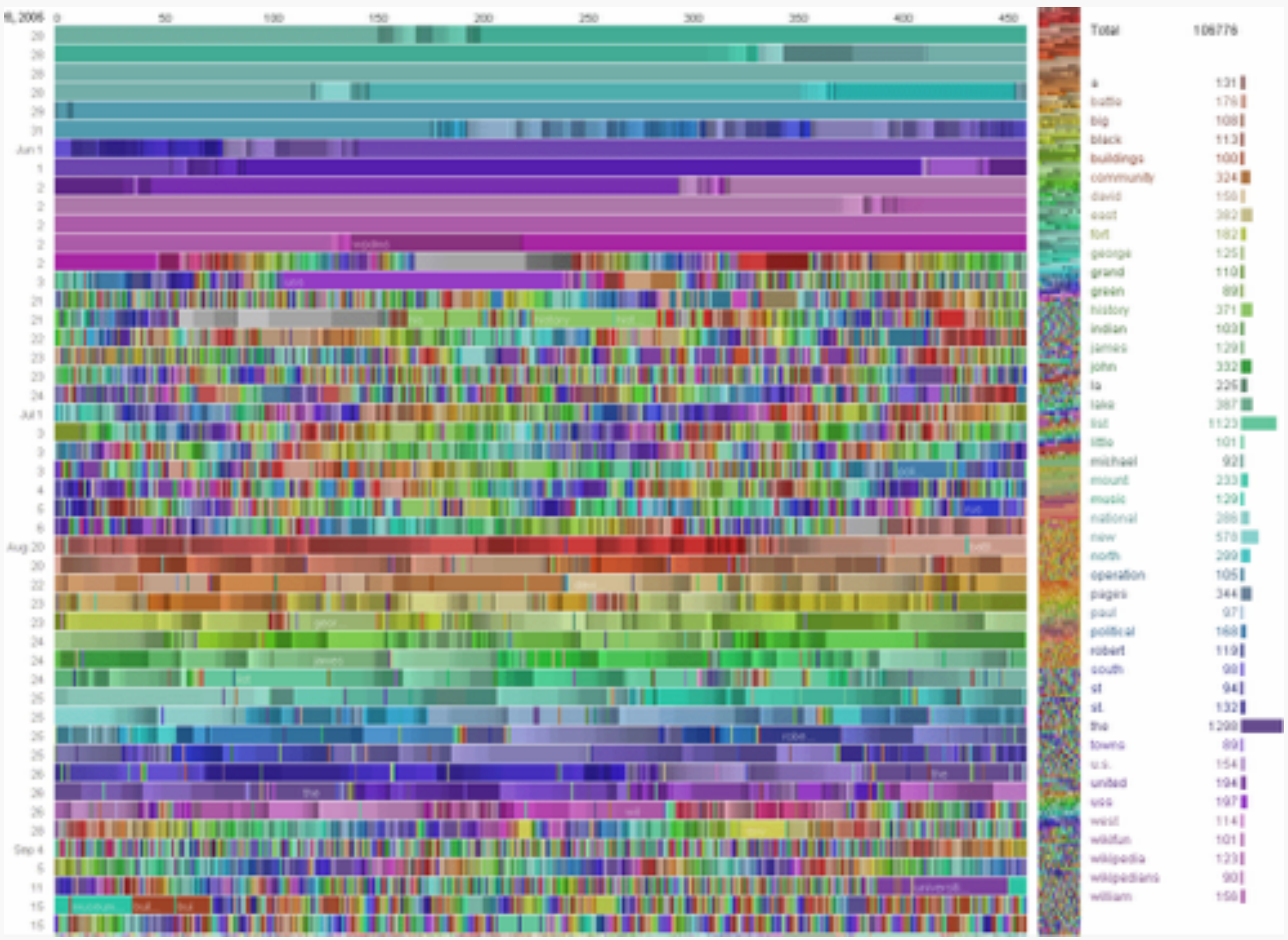



*Figure: http://colah.github.io/*

# When to use machine learning

*Supervised* learning – Classification or regression

When your data are too complex for traditional model development and fitting with statistics



# A simple example

Are these houses in **San Francisco** or **New York**?

- House elevation – 100 metres?
- Price - $1000 per square metre?
- Year built – 1900?

These are called *features*!

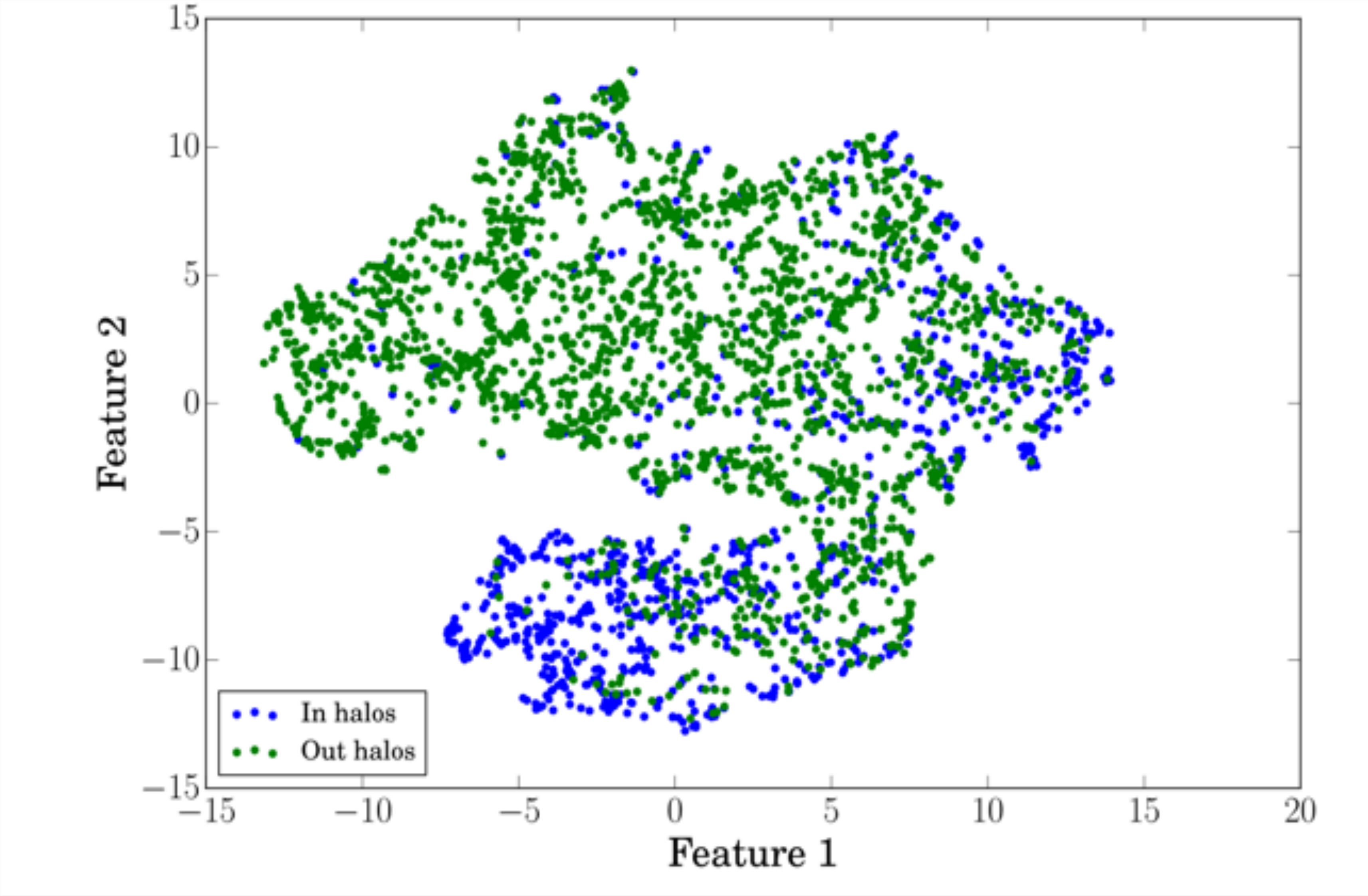
# Optimising your features

The choice of features is crucial to the machine learning algorithm.

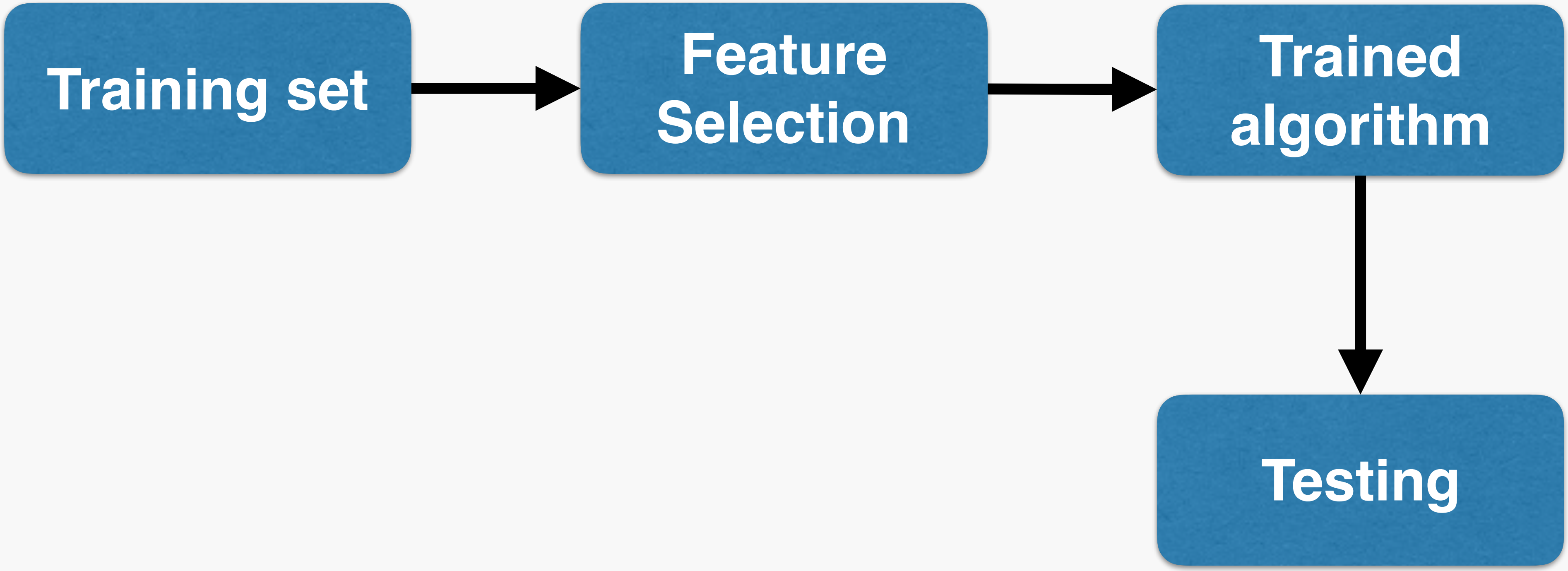
**Feature extraction –** projects original high-dimensional space into a new low-dimensional feature space

**Feature selection -** select only a subset of informative features

# Visualising your features with t-SNE



# The Main Steps



# Machine Learning Algorithms

- Naive Bayes (NB)
- K-nearest neighbours (KNN)
- Support vector machines (SVM)
- Artificial neural networks (ANN)
- Random Forests (RF)

# Naive Bayes

Likelihood (probability of feature given class)

Posterior probability

Class prior probability

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

All features are assumed to be drawn from independent Gaussian distributions. Mean and standard deviation are learned from the training set.

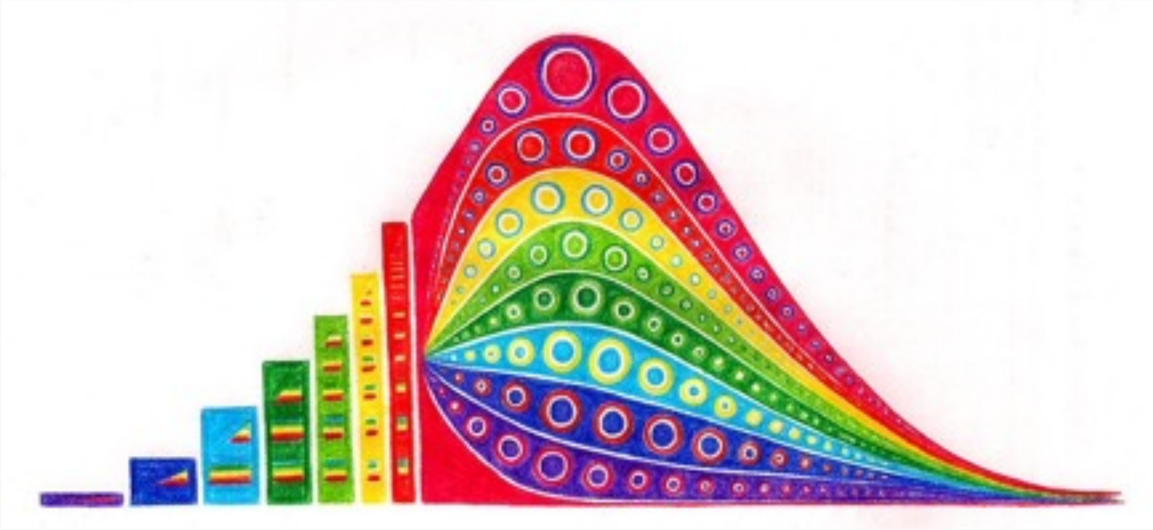
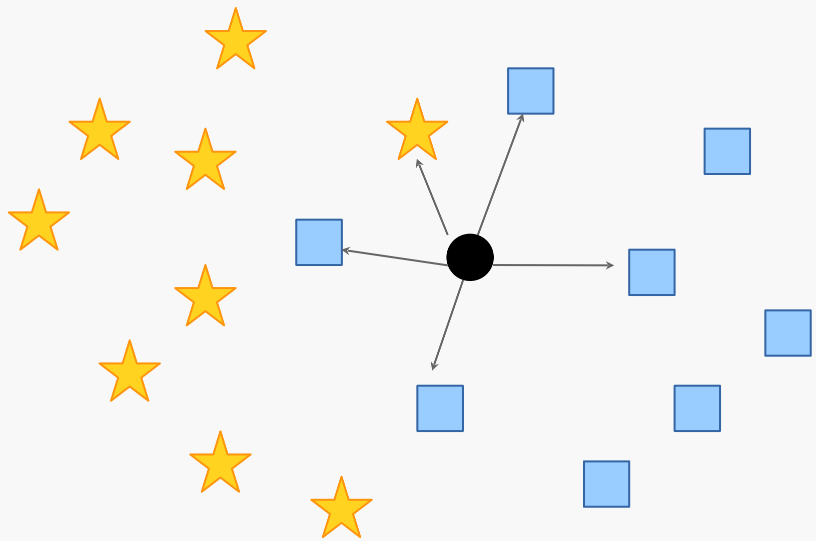


*Figure: LadyLeibniz, Deviant Art*

# K-nearest Neighbours

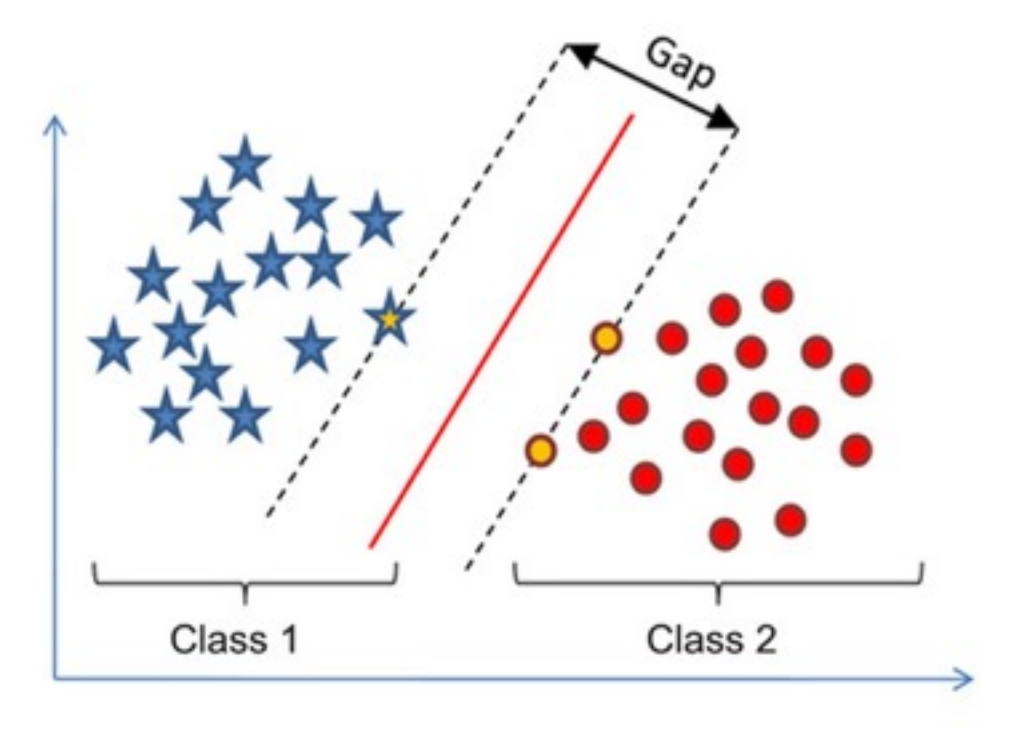
A new test point is classified based on the classes of its *k* nearest neighbours (from the training set)



Probability of belonging to a particular class is simply the normalised number of votes for that class (inversely weighted by distance)

# Support Vector Machines

Linear SVM finds the hyperplane in feature space that best separates classes




The hyperplane is described by the "support vectors", which are vectors between the data points on the margin (yellow points) and the hyperplane.

*Figure: Ankit Sharma, www.diggdata.in*

# Support Vector Machines

Non-linear SVM can be used by transforming to a higher dimensional (linear) feature space using the kernel trick.

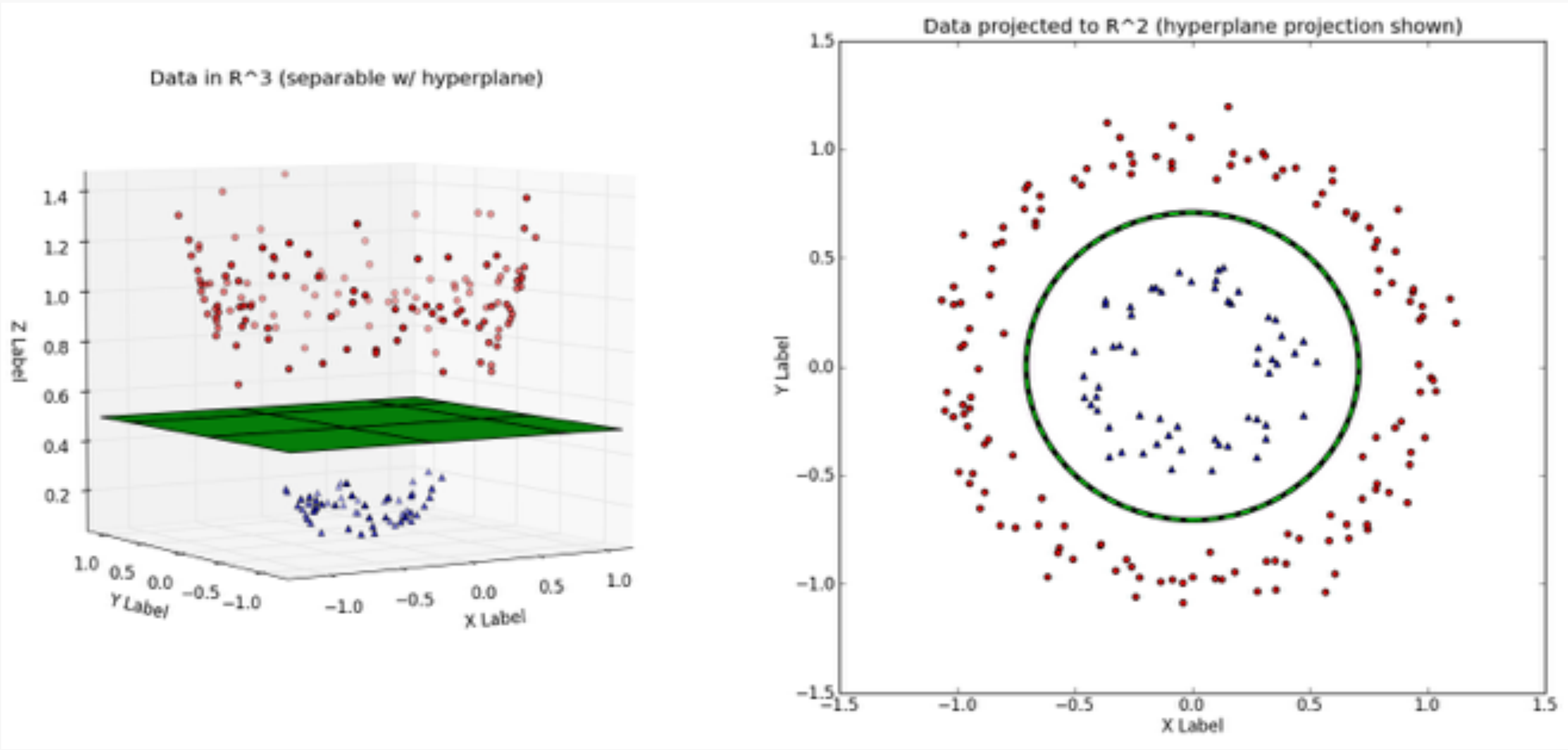



*Figure: Eric Kim, www.eric-kim.net*

# Artificial Neural Networks

Based on how the human brain learns (probably), ANNs are constructed from layers of connected neurons.

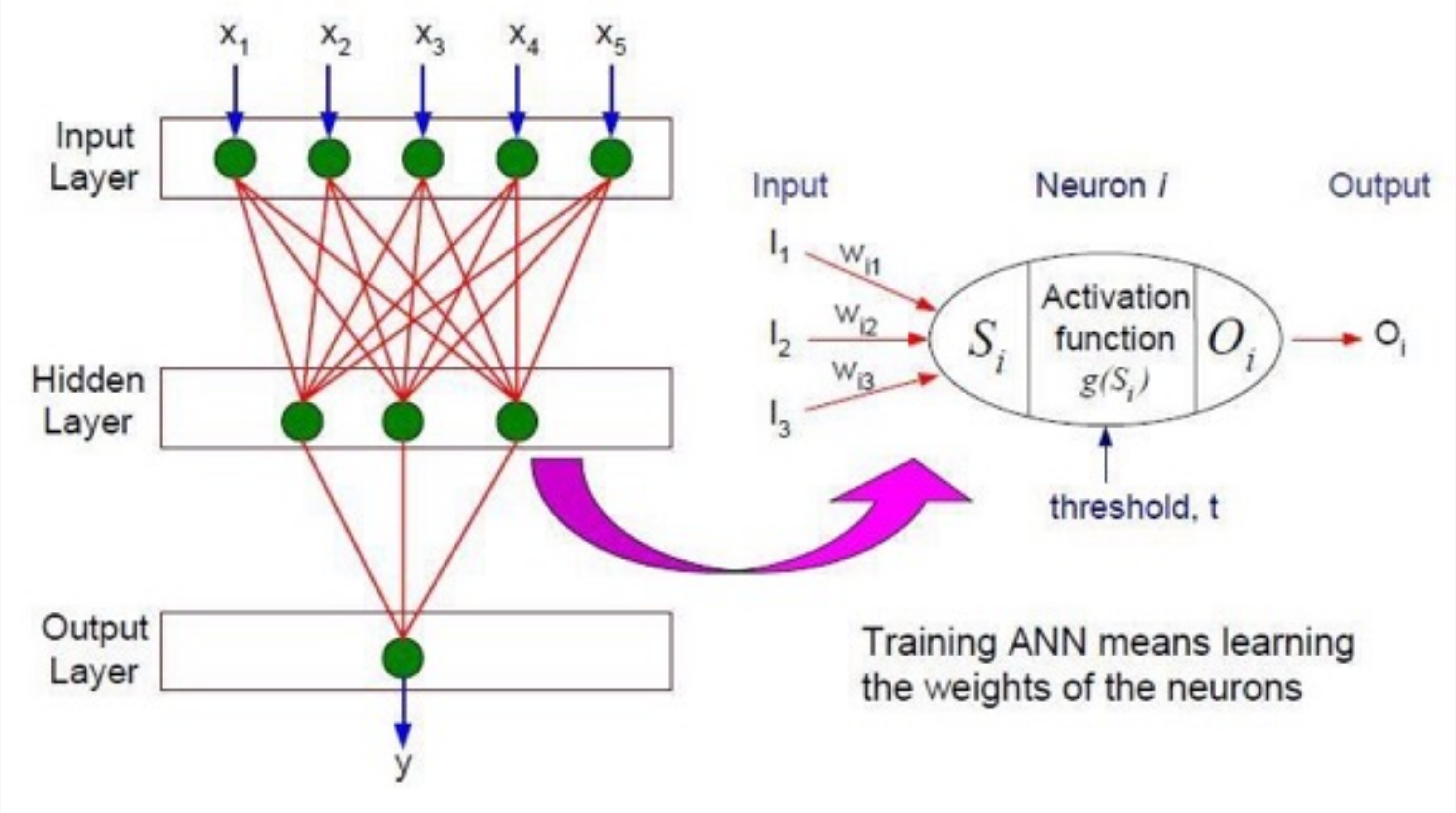
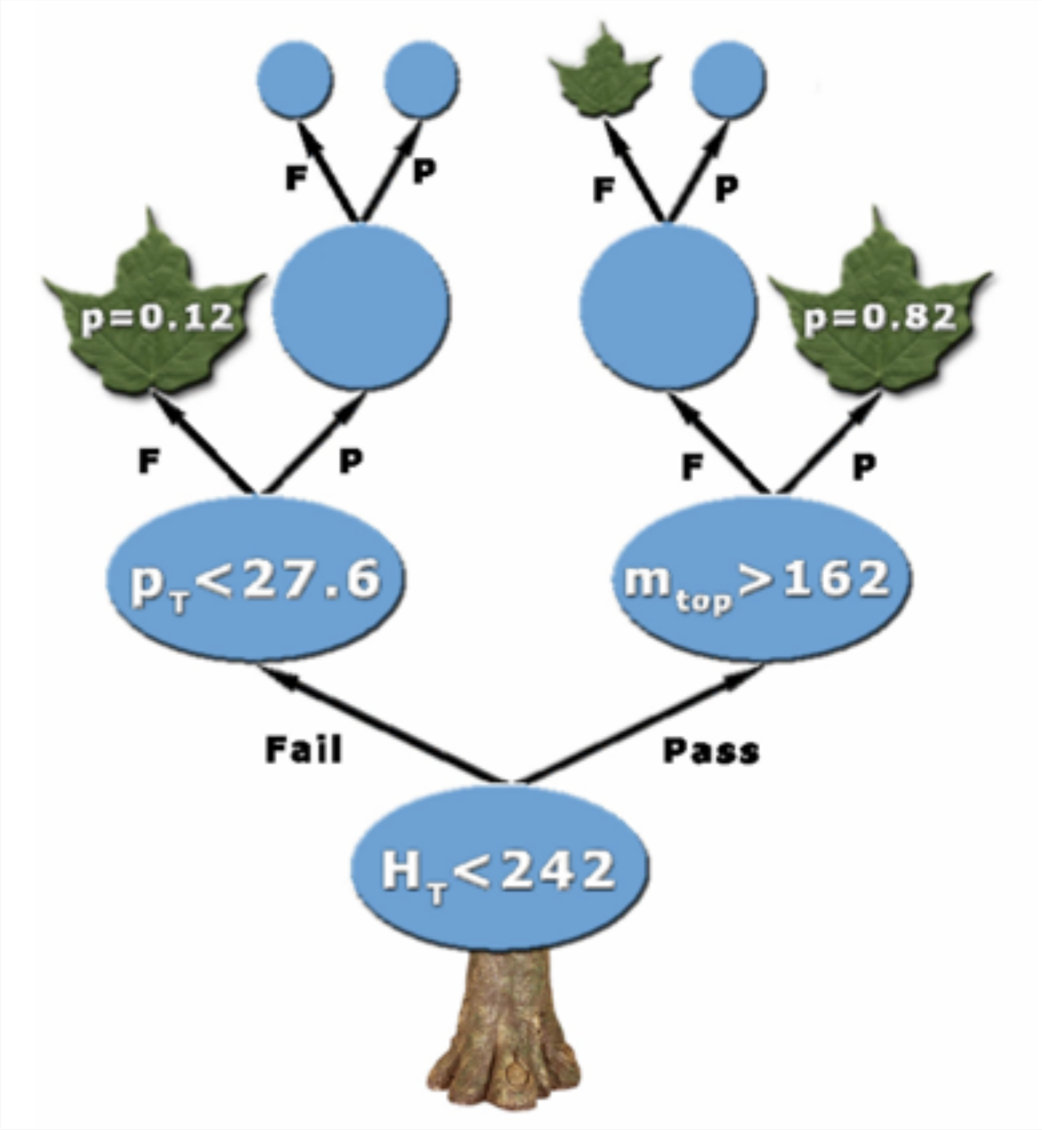



*Figure: http://mines.humanoriented.com/*

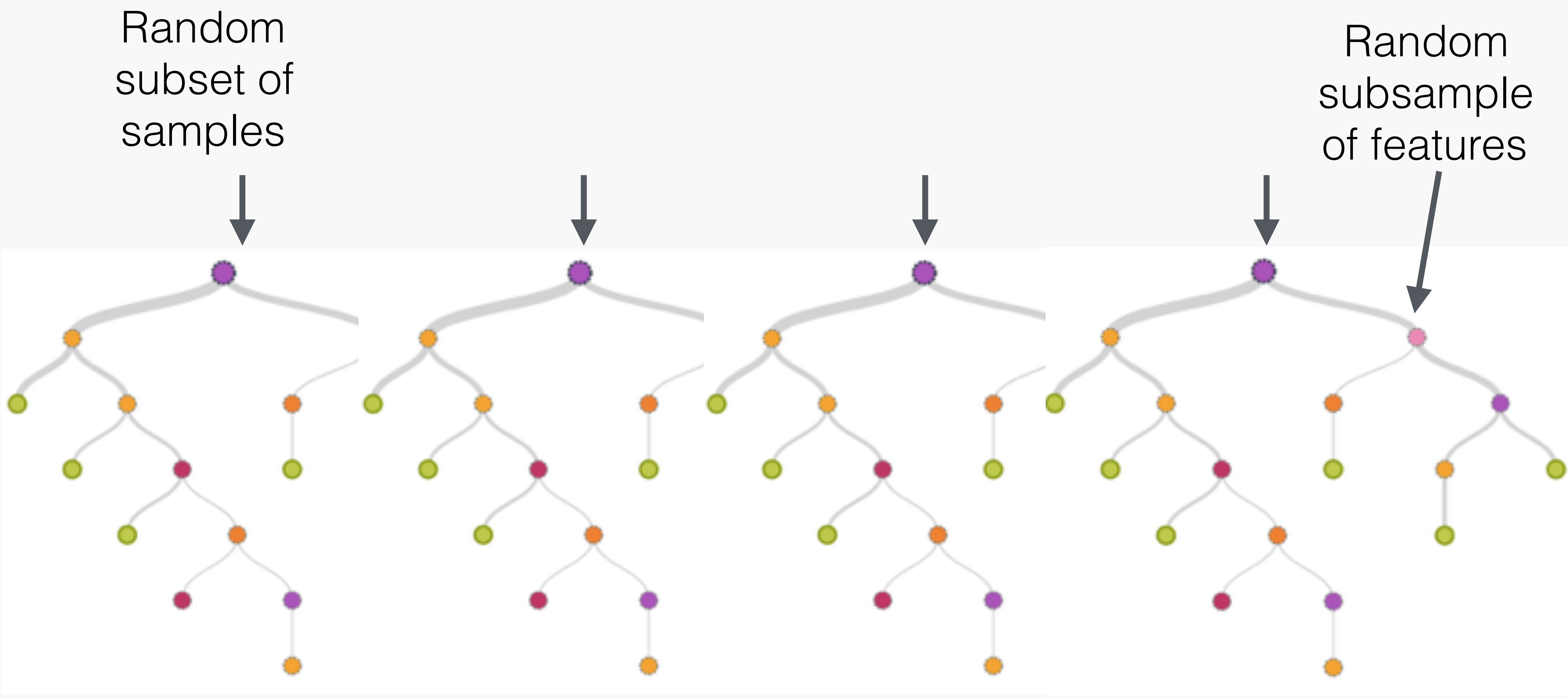
# Random Forests




Decision trees construct a series of nodes which make splits on a particular feature.

Decision trees are in general biased but can be effectively combined in ensemble methods.

# Random Forests



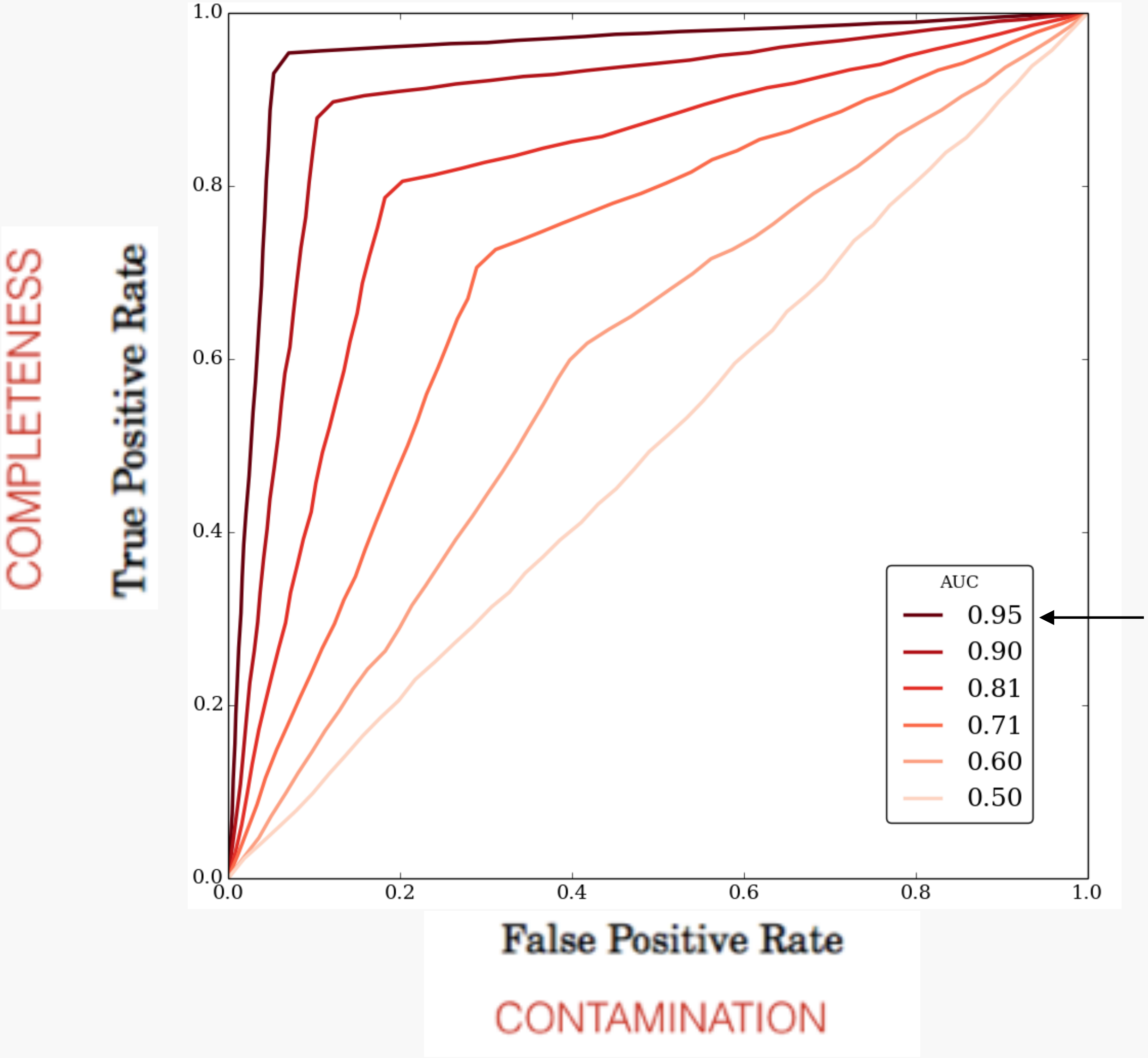

Final prediction =
average probabilistic predictions of individual trees

# Confusion Matrix

**Predicted labels**

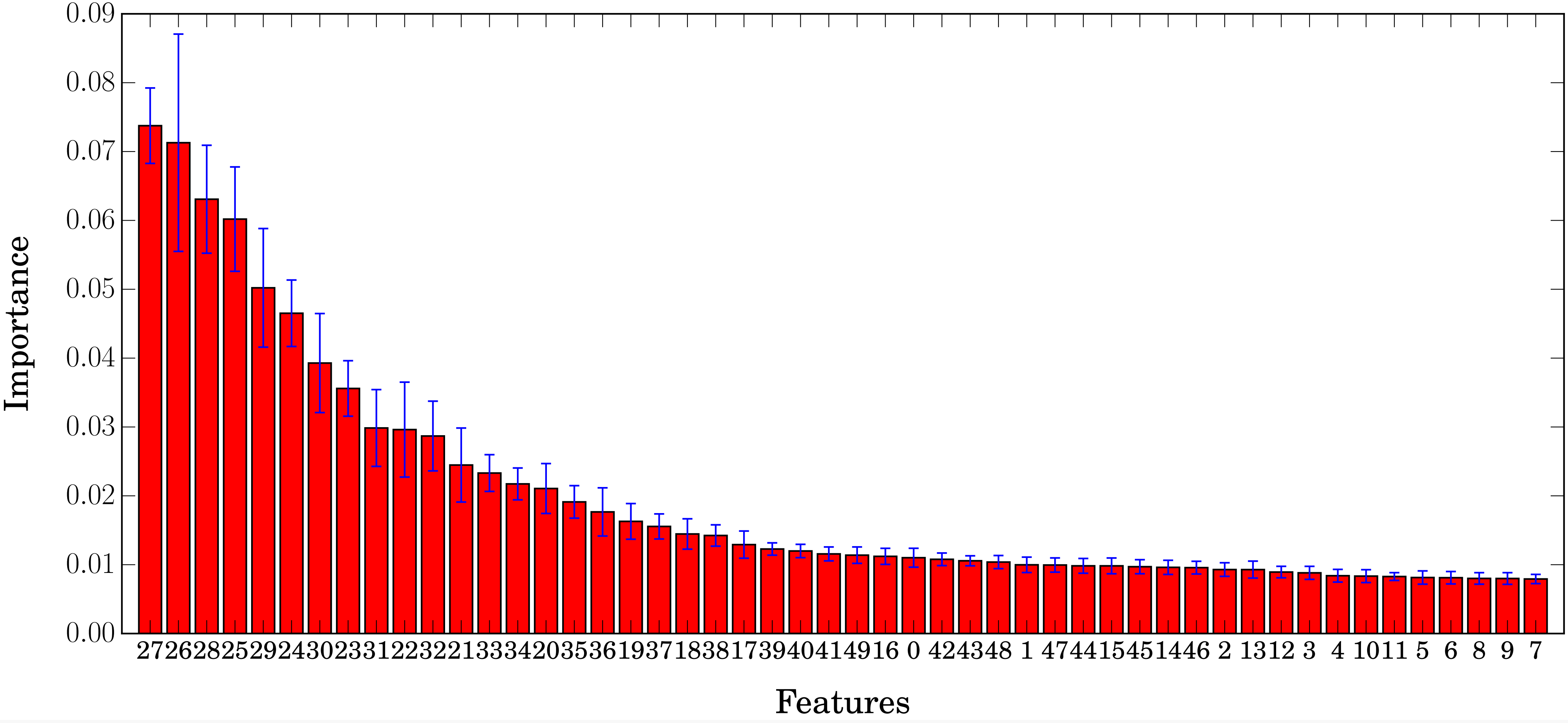
| | | **Positives ("in")** | **Negatives ("out")** |
|---|---|---|---|
| **True labels** | **Positives ("in")** | True Positives (TPs) | False Negatives (FNs) |
| | **Negatives ("out")** | False Positives (FPs) | True Negatives (TNs) |

# Receiver Operator Characteristic (ROC) curves




Area under the curve

# Feature Importances



# Is your training set overfitting?

- Overfitting is very bad
- Split data into three: training, validation and test
- Cross validation is the best weapon against overfitting

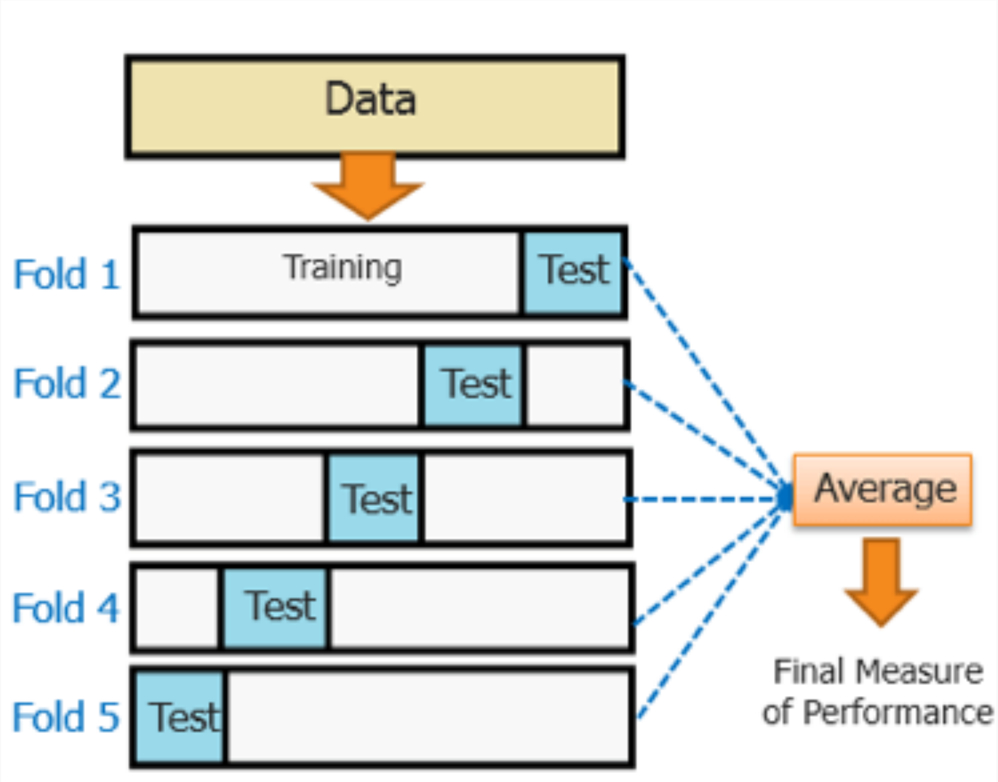
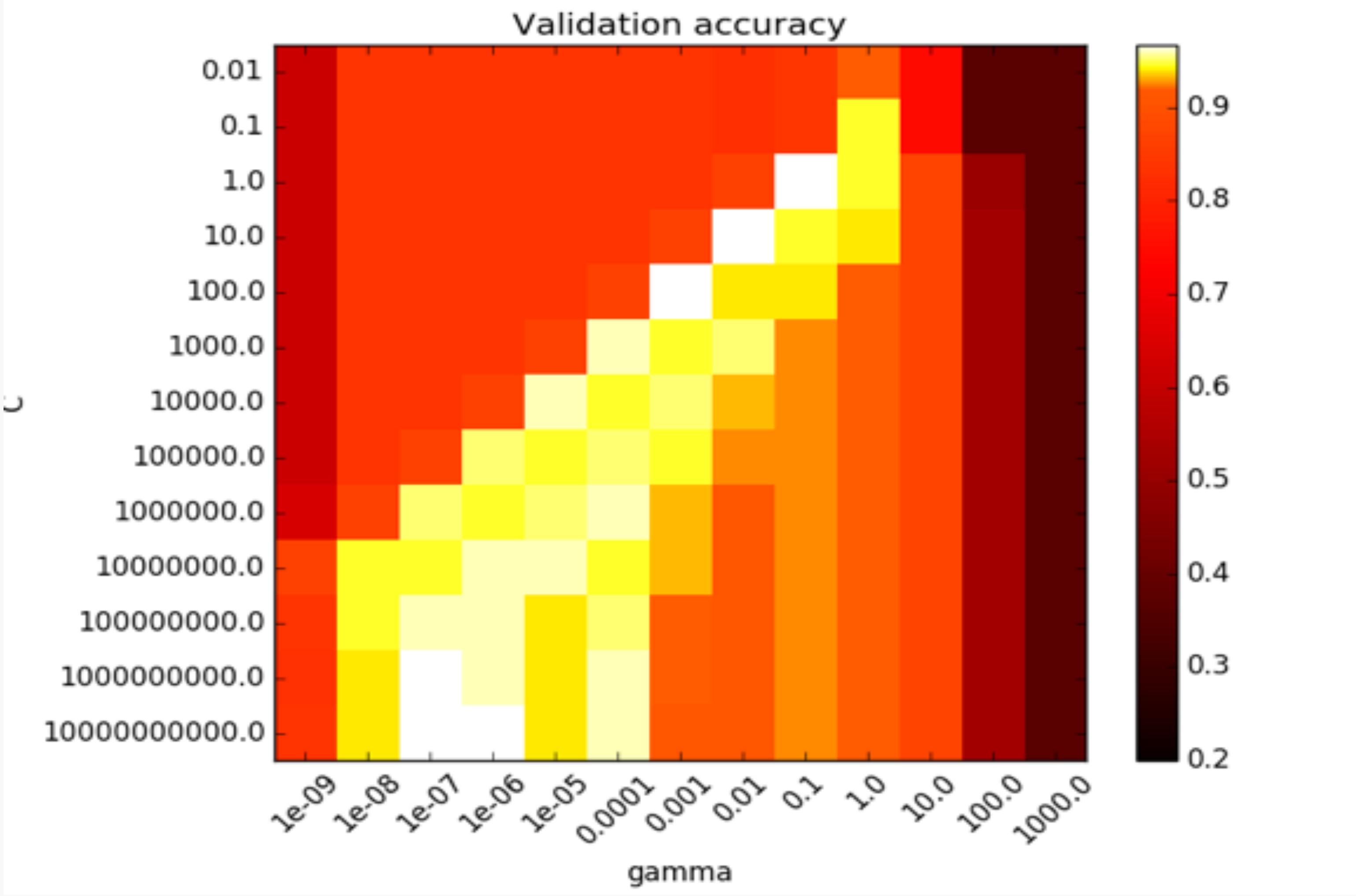


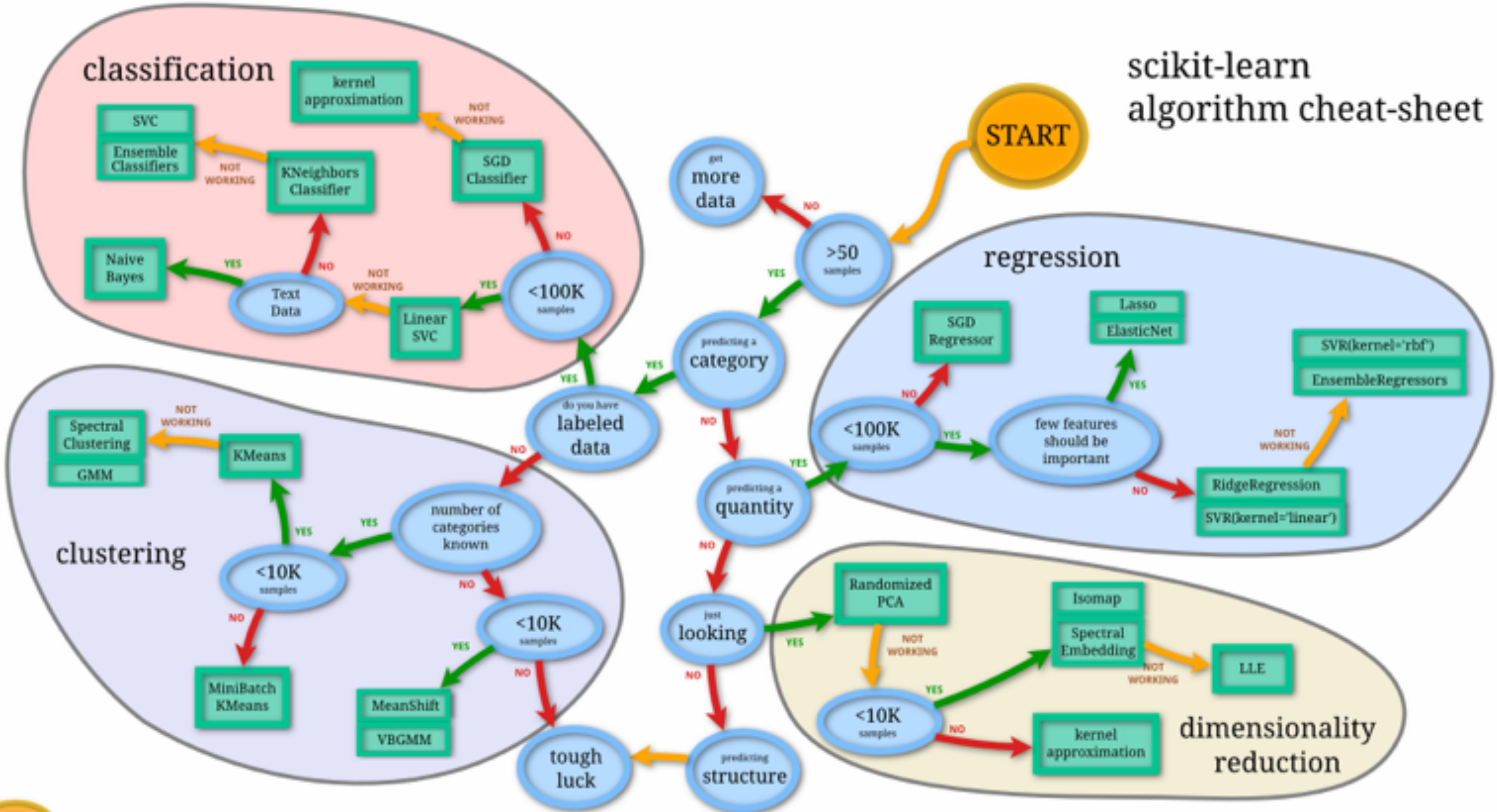

*Figure: www.edureka.in/data-science*

# Optimising the hyperparameters

Hyperparameters must be manually set a priori
– use `GridSearchCV`



# Tips and Tricks




Back

# Tips and Tricks

- Rescale your features (zero mean, unit variance)

# Tips and Tricks

- Rescale your features (zero mean, unit variance)
- Always keep aside a test set of data while you're experimenting with the algorithms, this avoids overfitting

# Tips and Tricks

- Rescale your features (zero mean, unit variance)
- Always keep aside a test set of data while you're experimenting with the algorithms, this avoids overfitting
- Use cross validation when optimising algorithm hyperparameters

# Tips and Tricks

- Rescale your features (zero mean, unit variance)
- Always keep aside a test set of data while you're experimenting with the algorithms, this avoids overfitting
- Use cross validation when optimising algorithm hyperparameters
- Use dimensionality reduction/ feature selection to improve performance by reducing number of features

# Tips and Tricks

- Rescale your features (zero mean, unit variance)
- Always keep aside a test set of data while you're experimenting with the algorithms, this avoids overfitting
- Use cross validation when optimising algorithm hyperparameters
- Use dimensionality reduction/ feature selection to improve performance by reducing number of features
- For classification, think about your choice of performance metric!

# References

https://www.coursera.org/learn/machine-learning

https://github.com/rasbt/python-machine-learning-book

https://github.com/jakevdp/sklearn_tutorial

http://ipython-books.github.io/featured-04/

Lochner et al. (2016) http://arxiv.org/abs/1603.00882

Email:

luisa.lucie-smith.15@ucl.ac.uk

# Tutorial

Using a RandomForestClassifier:

1. Visualise features using a t-SNE plot
2. Use GridSearchCV to cross-validate two hyperparameters
3. Train the random forest
4. Plot the ROC curve of the test set
5. Plot the feature importances