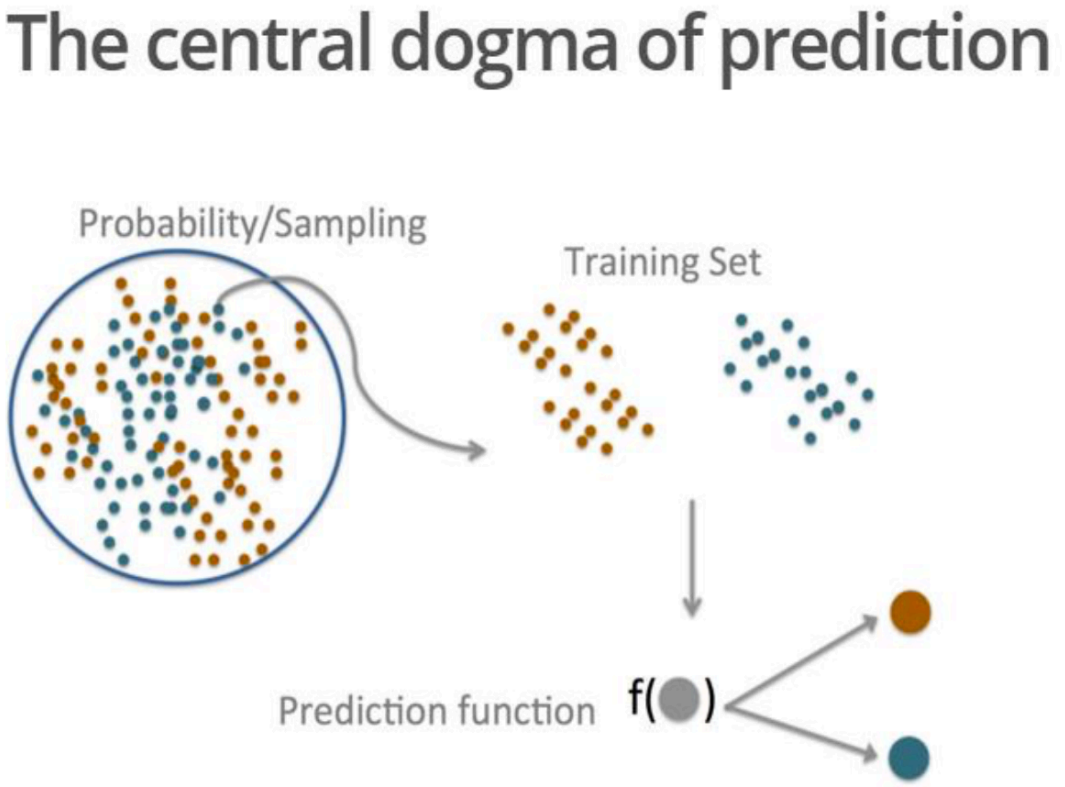
Introduction to Machine Learning

Aswadi Abdul Rahman

Introduction

- Prediction can be understood to be the following things by different groups of people:

Prediction	Types of Statisticians
Statistical Learning	Statisticians
Machine Learning	Computer Scientists
Forecasting	Atmospheric scientists/Bankers
Artificial Intelligence	Members of the press

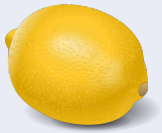


Machine Learning: Categories

- Machine Learning basically can be classified into 2 categories:
  - Supervised Machine Learning: Learn to predict target values from labelled data
    - Classification: Target values are discrete values
    - Regression: Target values are continuous values

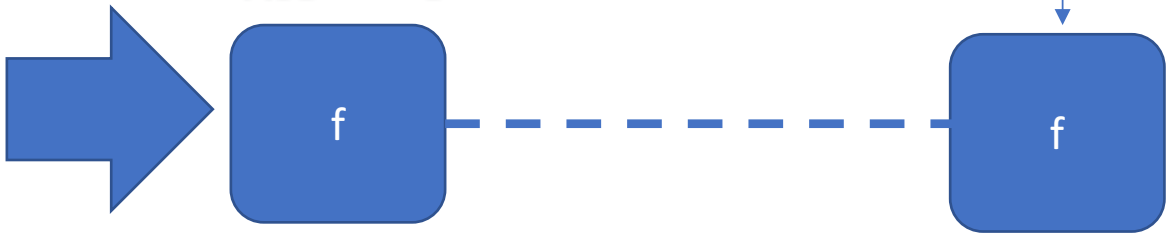
Supervised Learning: Classification Example

Training set

X sample	Y target value (Label)
 x_1	Apple y_1
 x_2	Orange y_2
 x_3	Apple y_3
 x_4	Lemon y_4

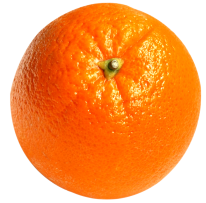
Classifier

f:$X \rightarrow Y$



At training time, the classifier uses labelled examples to learn rules for recognizing each fruit type

Future sample

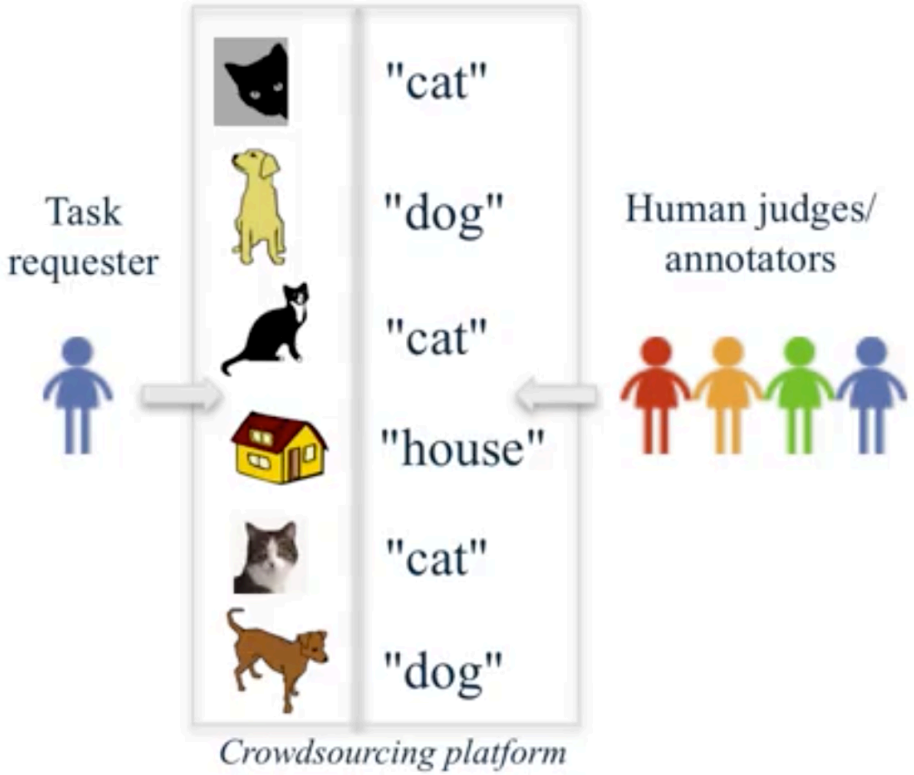


Label: Orange

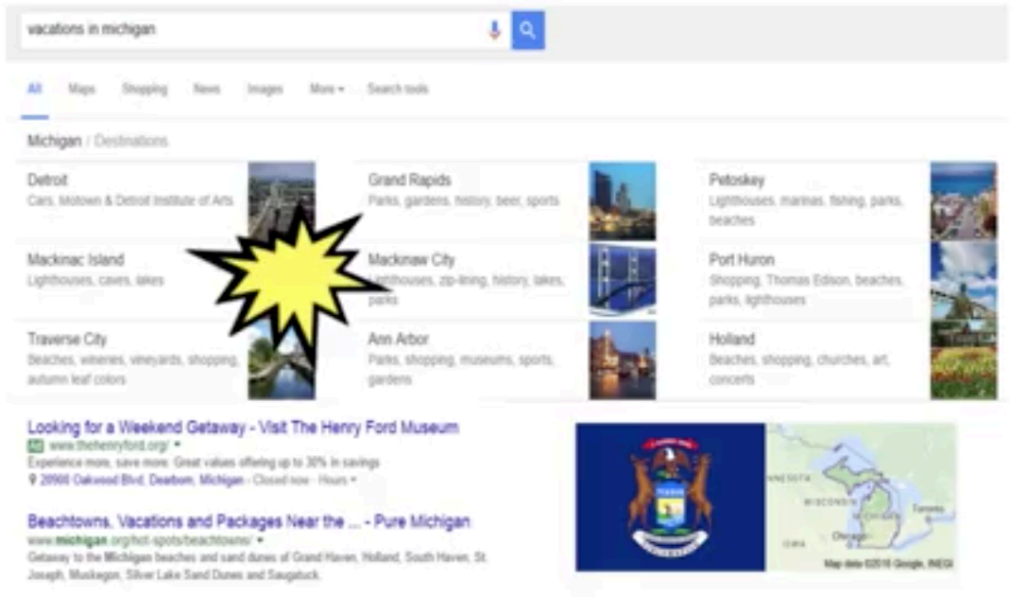
After training, at prediction time, the trained model is used to predict the fruit type for new instances using the learned rules

Examples of explicit and implicit label sources

Explicit labels



Implicit labels



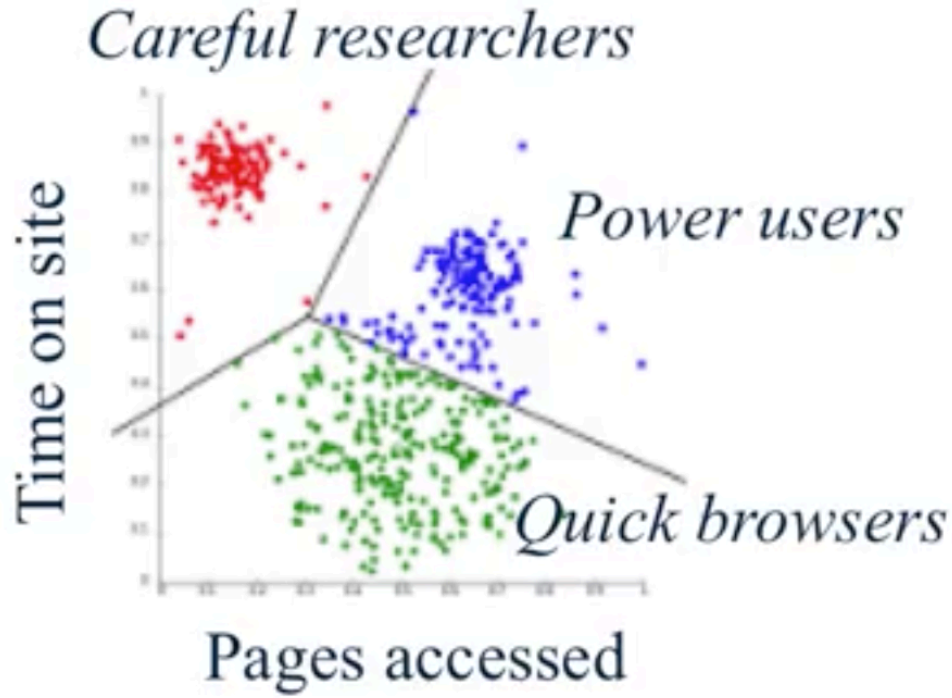
Clicking and reading the "Mackinac Island"result can be an implicit label for the search engine to learn that "Mackinac Island" especially relevant for the query [vacation in Michigan] for that user

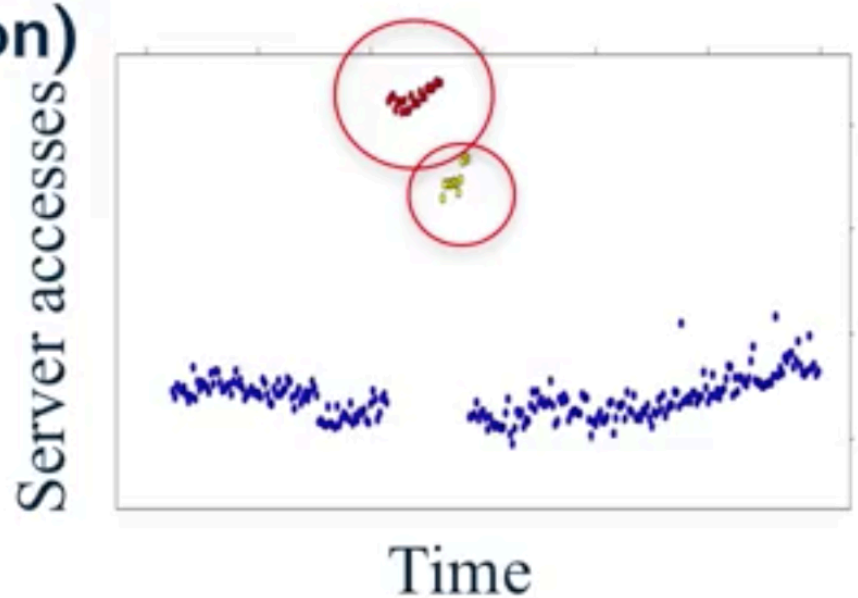
Machine Learning: Categories

- Machine Learning basically can be classified into 2 categories:
  - Supervised Machine Learning: Learn to predict target values from labelled data
    - Classification: Target values are discrete values
    - Regression: Target values are continuous values
  - Unsupervised Machine Learning: Find structure in labeled data
    - Find groups of similar instances in the data (clustering)
    - Finding unusual patterns (outlier detection)

Unsupervised learning: finding useful structure or knowledge in data when no labels are available

- Finding clusters of similar users (clustering)
- Detecting abnormal server access patterns (unsupervised outlier detection)





Quiz

- Which of the following are examples of **supervised machine learning**? Select all that apply.

a) Analyzing sales data to find groups of customers with similar buying habits

b) Making online movie recommendation based on the ”star” ratings that other users provided with their own online reviews

c) A face recognition system that trained using images with crowdsourced labels

d) A search engine that users user clicks on the results page to learn which pages are most relevant to popular queries

Quiz

- Which of the following are examples of **supervised machine learning**? Select all that apply.

a) Analyzing sales data to find groups of customers with similar buying habits

✔ b) Making online movie recommendation based on the ”star” ratings that other users provided with their own online reviews

✔ c) A face recognition system that trained using images with crowdsourced labels

✔ d) A search engine that users user clicks on the results page to learn which pages are most relevant to popular queries

A Basic Machine Learning Workflow

Choose:
- A feature representation
- Type of classifier to use

e.g. image pixels, with k-nearest neighbor classifier

Choose:
- What criterion distinguishes good vs. bad classifiers?

e.g. % correct predictions on test set

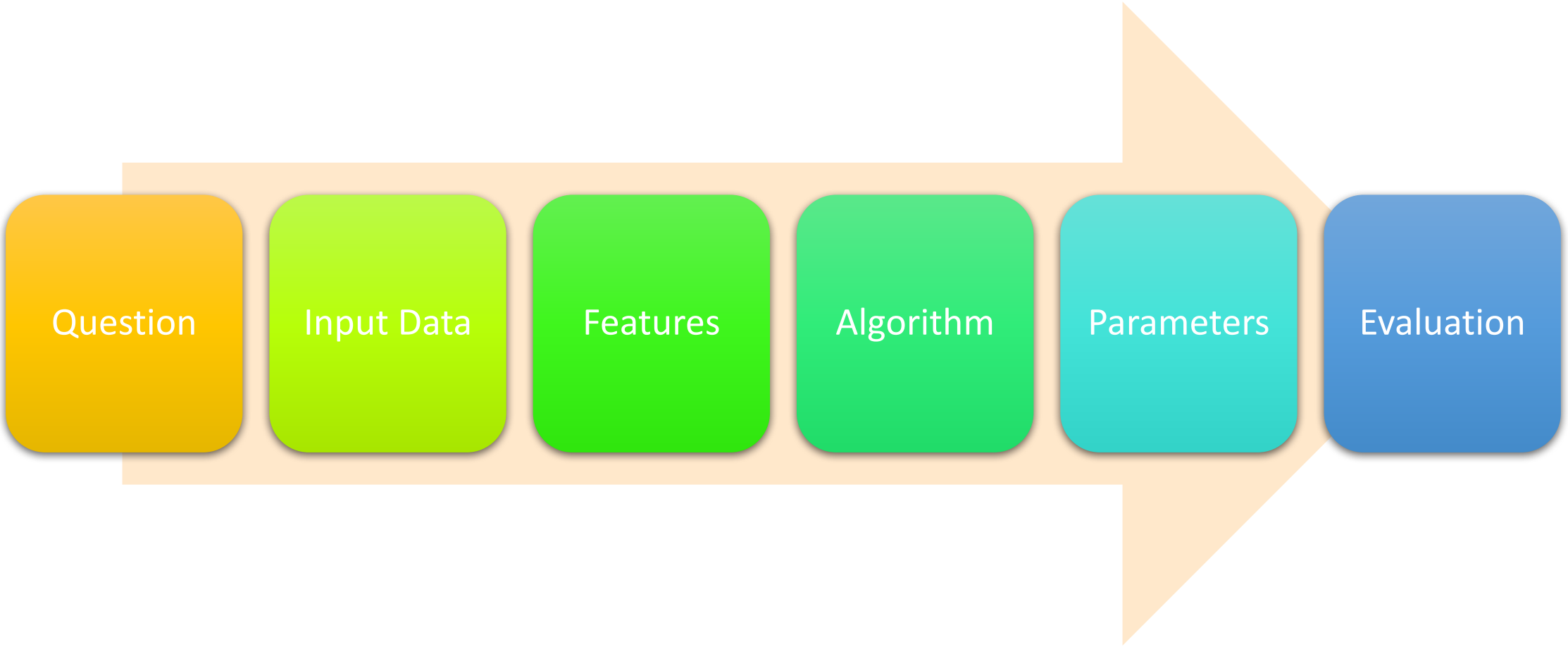
Choose:
- How to search for the settings/parameters that give the best classifier for this evaluation criterion

e.g. try a range of values for "k" parameter in k-nearest neighbor classifier

Prediction pseudo-code

1. Choose training set
2. Use characteristics of training set
3. Get prediction function
4. Evaluate how prediction function does

Components of a predictor



Feature Representations

Feature representation

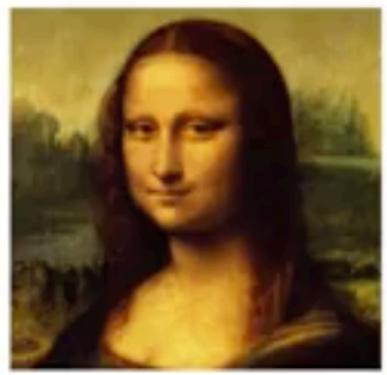
Email

```
To:  Chris Brooks
From:  Daniel Romero
Subject:  Next course offering
Hi Daniel,
Could you please send the outline for the
next course offering?  Thanks! -- Chris
```

Feature	Count
to	1
chris	2
brooks	1
from	1
daniel	2
romero	1
the	2
...	

A list of words with their frequency counts

Picture



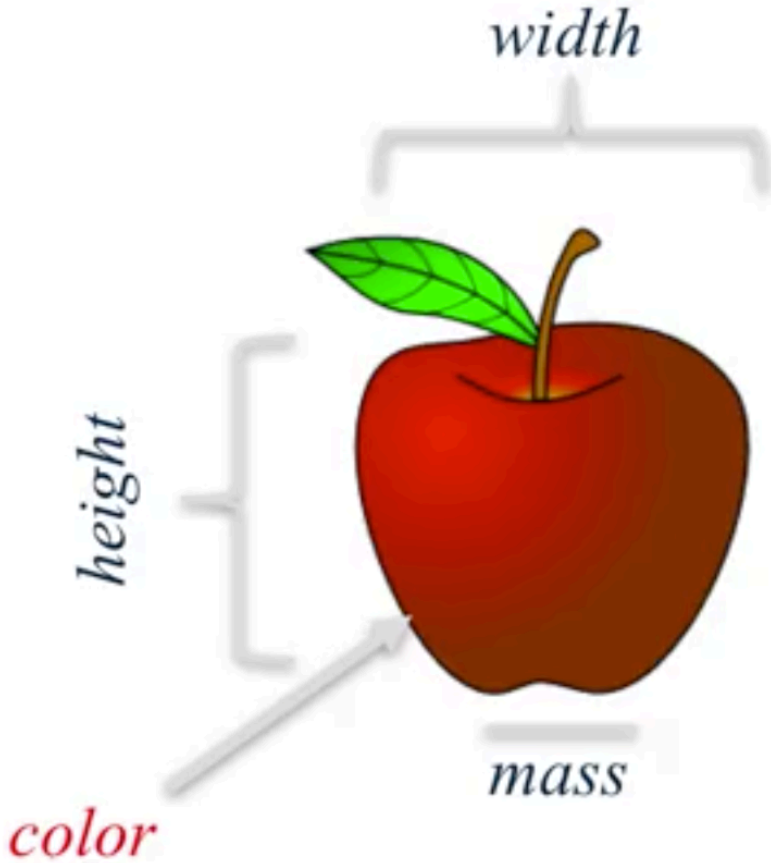
A matrix of color values (pixels)

Sea Creatures

Feature	Value
DorsalFin	Yes
MainColor	Orange
Stripes	Yes
StripeColor1	White
StripeColor2	Black
Length	4.3 cm

A set of attribute values

Representing a piece of fruit as an array of features (plus label information)



1. Feature representation

Label information
(available in training data only)

Feature representation

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
18	1	apple	cripps_pink	162	7.5	7.1	0.83

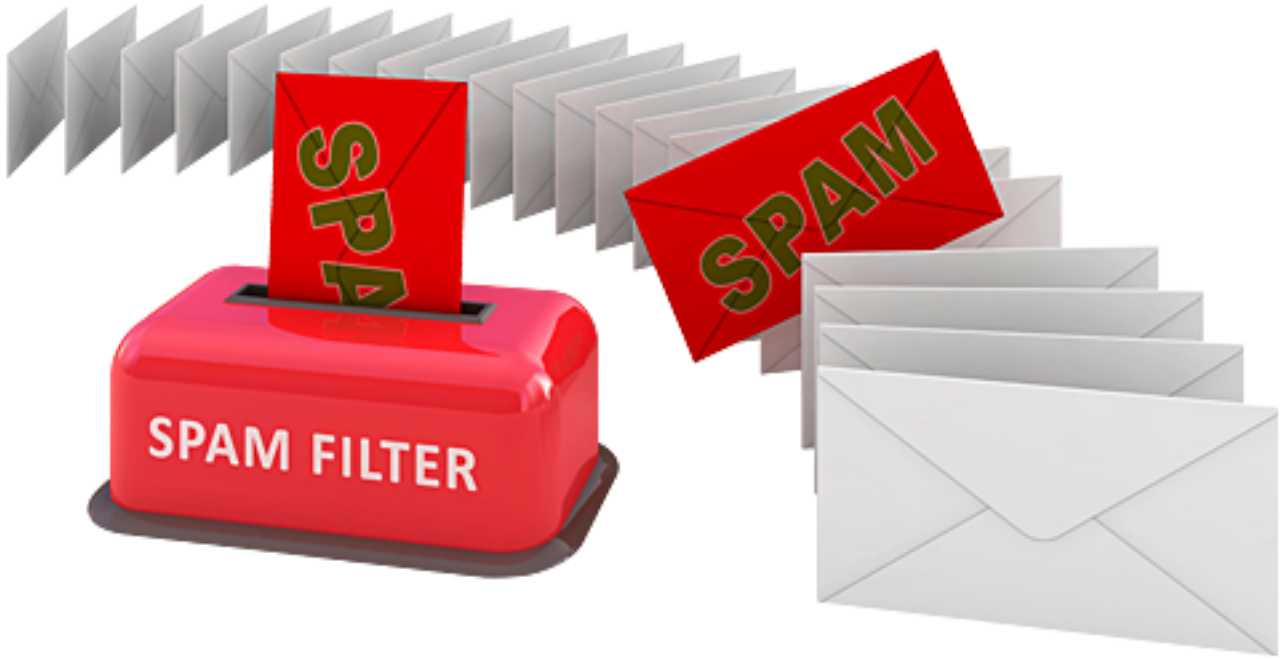
2. Learning model

Classifier

Predicted class
(apple)

Example: Classifying SPAM emails

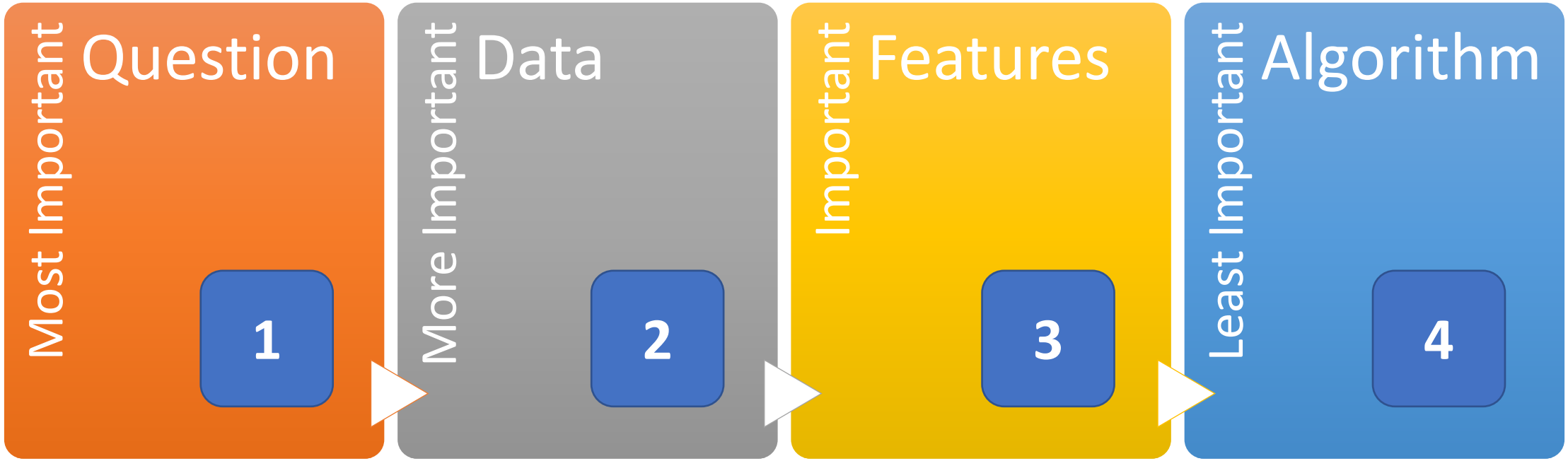
- Classic example of machine learning: SPAM emails classification



Summary of the SPAM example

Components	Elaboration
Question	Can I use quantitative attributes of the email to classify the email as belonging to the SPAM or non- SPAM group?
Input Data	Is the data perfect or sufficient? If not what do you do about it?
Features	In this context, can be a word count : in this example the count of the word ‘you’ is evaluated as a feature through
Algorithm	Here, a kernel density function of the frequency of the word ‘your’ in the email is used as the prediction function
Parameter	The cut-off point of 0.5 is used to differentiate spam emails from non-spam email
Evaluation	The prediction accuracy is measured in terms of percentage of correctly predicted labels

Which components are more important?



Key take-aways concerning Questions, Data, Features and Algorithm

- For DATA, garbage in = garbage out
  - More DATA is better than BETTER prediction models
- For FEATURES, it should be based on EXPERT domain knowledge, and lead to data COMPRESSION while RETAINING relevant information
  - It’s a mistake to rely heavily on automatic feature selection, ignore data quirks and throw away information
- For ALGORITHM, it matters less, with sometimes marginal differences between prediction models

What is the “BEST” ML method then?

Prediction : All about making a trade-off

Do you wants?	Or do you want?	Example
ACCURACY	Easy to interpret	Decision Tree, Logistic Regression
	Speed	Regression
	Simplicity	Regression
	Scalability	

In Sample VS. Out of Sample Errors

- In Sample Error : the error rate you get on the same data set you used to build your predictor, aka resubstitution error
- Out of sample error: the error rate you get on a new data set, aka generalization error

This is what we should care about, due to overfitting errors (in other words, out of sample error is more important than in sample error)

In Sample Vs Out Sample Errors Discussion

- Let's assume a subset of the SPAM data is taken, and called smallSpam – this consists of 10 observations selected at random
- A set of prediction rules is build around the feature capitalAve (average length uninterrupted sequence of capital letters)
- Prediction rule 1:
  - capitalAve > 2.7 = 'spam'
  - capitalAve < 2.4 = 'nonspam'
  - between 2.4 and 2.45 = spam
  - between 2.45 and 2.7 = nonspam
- This gives perfect results!

```
> table(rule1(smallSpam$capitalAve),smallSpam$type)#accuracy using rule 1

         nonspam spam
nonspam        5    0
spam           0    5
```

In Sample Vs Out Sample Errors Discussion

- Suppose we introduce Rule 2, which states that capitalAve > 2.4 = spam, else nonspam

```
> table(rule2(smallSpam$capitalAve),smallSpam$type)#accuracy using rule 2

         nonspam spam
nonspam        5    1
spam           0    4
```

- This results in one error where a non-spam is classified as a spam
- Which Rule is better? Rule 1 or Rule 2?

In Sample Vs Out Sample Errors Discussion

- At this point we have considered Rule 1 and Rule 2 on a subset of the SPAM dataset
- In this case, for the subset that we have considered , in sample error seems acceptable
- What of Out Sample Errors?
- Test Rule 1 and Rule 2 on the entire SPAM dataset.

```
> table(rule1(spam$capitalAve), spam$type)

         nonspam spam
nonspam    2141  588
spam        647 1225
> table(rule2(spam$capitalAve), spam$type)#apply rule2 to full SPAM dataset

         nonspam spam
nonspam    2224  642
spam        564 1171
```

- Rule 1 and Rule 2's accuracy for the whole dataset is examined. Which is better?

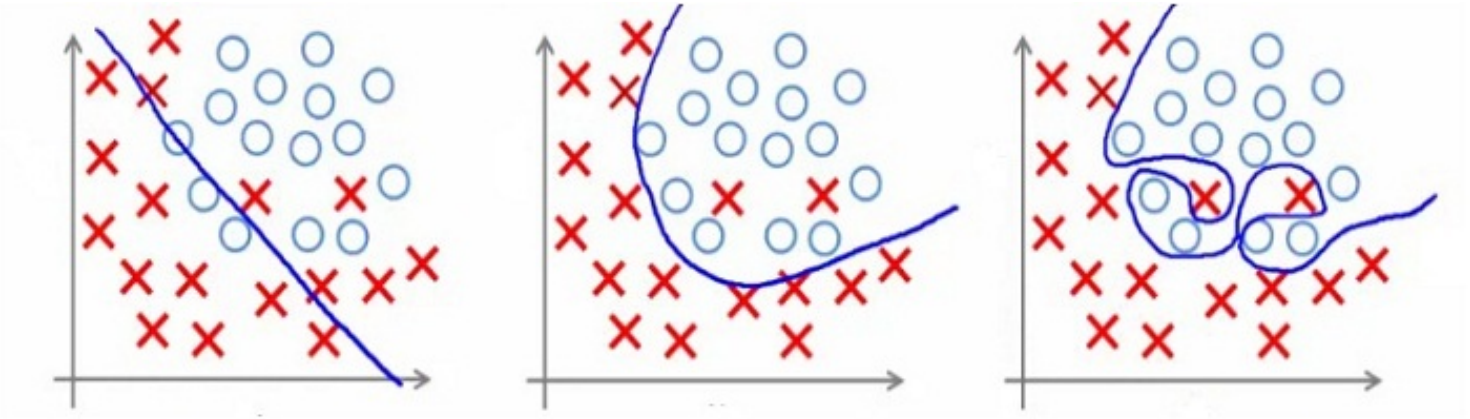
```
> sum(rule1(spam$capitalAve) == spam$type)
[1] 3366
> sum(rule2(spam$capitalAve) == spam$type)
[1] 3395
```

In Sample vs Out Sample Error Discussion

- What caused Rule 1 to do well on the small SPAM data set but not so on the full SPAM data set?
- OVERFITTING has occurred
- Data have two parts: signal, noise
- The goal of a predictor is to find signal
- You can always design a perfect in-sample predictor
- You capture both signal and noise when you do that
- Predictor won't perform as well on new samples

Generalization vs Overfitting vs Underfitting

- **Generalization** – algorithm's ability to give accurate predictions for new, previously unseen data
- **Overfitting** - occurs when a model is excessively complex, such as having too many parameters relative to the number of observations. Overfitting describes random error or noise instead of underlying relationship
- **Underfitting** - occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data. It is when the model or the algorithm does not fit the data well enough



Tips on minimizing In and Out of Sample Error

- Define your error rate
- Split data into: training, testing, validation (optional)
- On the training set, pick features. use cross-validation
- On the training set, pick prediction function. use cross-validation
- If no validation: apply once to test set (don't want to use test set to train the model)
- If there is validation: apply to test set and refine, and apply once to validation

Tips on minimizing In and Out of Sample Error

- Know the benchmarks
- Study design:
  - E.g. Netflix, test may be 'hold out set', split to 3: probe (labels provided), quiz, test (so you would train on training, and apply to the probe. Netflix applied model to quiz set and you could see results. Test was for ultimate test at end of competition.)
  - Kaggle: used by professionals
- Avoid small sample sizes, especially for test set

Tips on minimizing In and Out of Sample Error

- Rules of thumb for prediction study design
  - If you have a large sample size: 60% training, 20% test, 20% validation
  - Medium sample size: 60% training, 40% testing
  - Small: do cross validation, report caveat of small sample size, or even reconsider whether to do
- Some principles to remember
  - Set the test validation set aside and don't look at it
  - In general, randomly sample training and test
  - Your data sets must reflect structure of the problem - if predictions evolve with time, split train/test in time chunks (called backtesting in finance)
  - All subsets should reflect as much diversity as possible: random assignment does this; you can also try to balance by features, but this is tricky

Common error measures

- Mean Squared Error (or Root Mean Squared Error)
  - For continuous data, e.g time-series data
- Median Absolute Deviation
  - Also for continuous data, more robust
- Sensitivity (recall): if you want few missed positives
- Specificity: if you want few negatives called positives
- Accuracy: weights false positives/negatives equally
- Concordance: distance measures for multi-class data, example: kappa

Some Error Measures From Confusion Matrix

- Let's start with an example confusion matrix for a binary classifier (though it can easily be extended to the case of more than two classes): Here let’ consider prediction of diseases

N= 165	Predicted: NO	Predicted: YES
Actual: NO	TN 50	FP 10
Actual: YES	FN 5	TP 100

- Some Useful Terms
  - **True Positives (TP):** These are cases in which we predicted yes (they have the disease), and they do have the disease.
  - **True Negatives (TN):** We predicted no, and they don't have the disease.
  - **False Positives (FP):** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
  - **False Negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

Some Error Measures from a Confusion Matrix

N= 165	Predicted: NO	Predicted: YES	
Actual: NO	TN 50	FP 10	60
Actual: YES	FN 5	TP 100	105
	55	110	

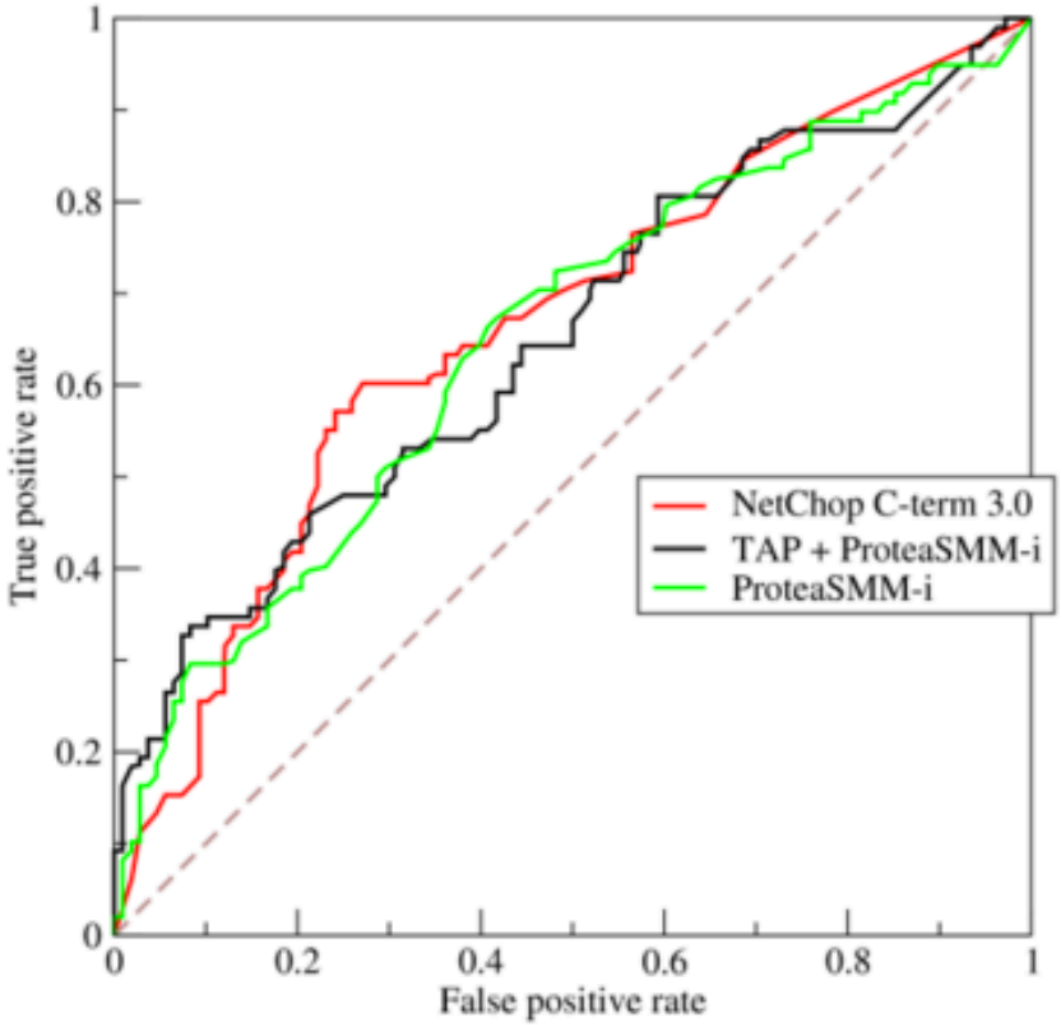
- This is a list of rates that are often computed from a confusion matrix:
- **Accuracy**: Overall, how often is the classifier correct?
  - (TP+TN)/total = (100+50)/165 = 0.91
- **Misclassification Rate:** Overall, how often is it wrong?
  - (FP+FN)/total = (10+5)/165 = 0.09
  - equivalent to 1 minus Accuracy
  - also known as "Error Rate"
- **True Positive Rate:** When it's actually yes, how often does it predict yes?
  - TP/(TP+FN) = 100/105 = 0.95
  - also known as "Sensitivity" or "Recall"
- **False Positive Rate:** When it's actually no, how often does it predict yes?
  - FP/(FP+TN)= 10/60 = 0.17
- **Specificity:** When it's actually no, how often does it predict no?
  - TN/actual no = 50/60 = 0.83
  - equivalent to 1 minus False Positive Rate
- **Precision:** When it predicts yes, how often is it correct?
  - TP/predicted yes = 100/110 = 0.91
- **Prevalence:** How often does the yes condition actually occur in our sample?
  - actual yes/total = 105/165 = 0.64

Receiver Operating Characteristics (ROC)

- ROC is a measure of the quality of the goodness of a prediction algorithm
- ROC exists as a curve because:
  - In binary classification, you are predicting one of two categories: alive/dead, ad click/not
  - But predictions are often quantitative: P(alive), prediction on a scale from 1-10
  - The cutoff you choose gives different results

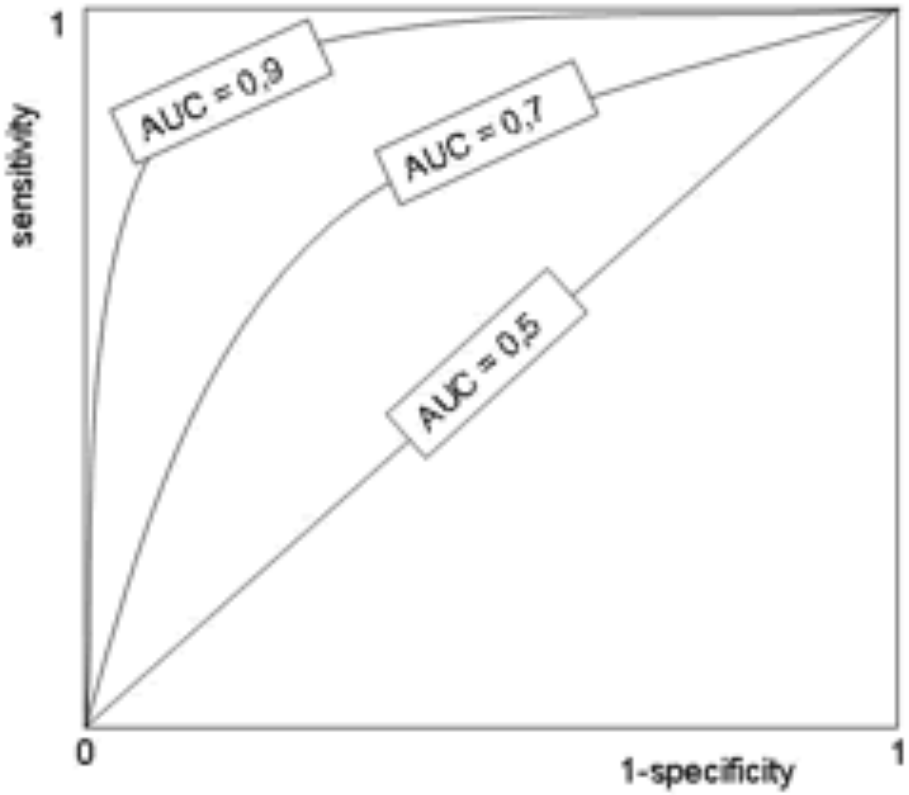
Description of a ROC curve

- x axis = 1-specificity (P(FP) = false positive rate)
- y axis = sensitivity (P(TP) = true positive rate)
- Both axes go 0 to 1, with a concave down curve



Evaluating **A**rea **U**nder ROC **C**urve (AUC)

- AUC = 0.5: random guessing (45 degree line basically)
- AUC < 0.5 means you did worse
- AUC = 1: perfect classifier
- In general, AUC above 0.8 is considered 'good'



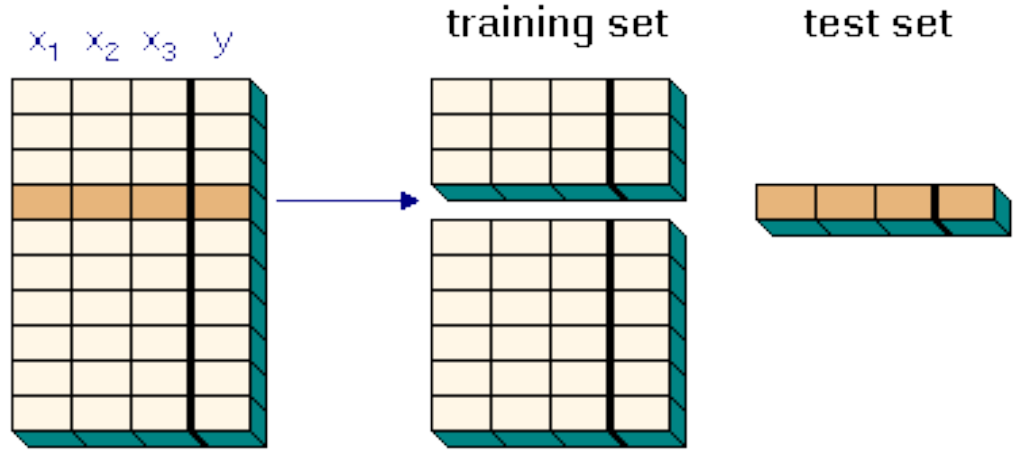
Cross-Validation

Key ideas:

- Accuracy on the training set (re-substitution accuracy) is optimistic
- A better estimate comes from an independent set (test set accuracy)
- However we can't use the test set when building the model or it becomes part of the training set
- Therefore we estimate the test set accuracy with the training set

Cross Validation Approach

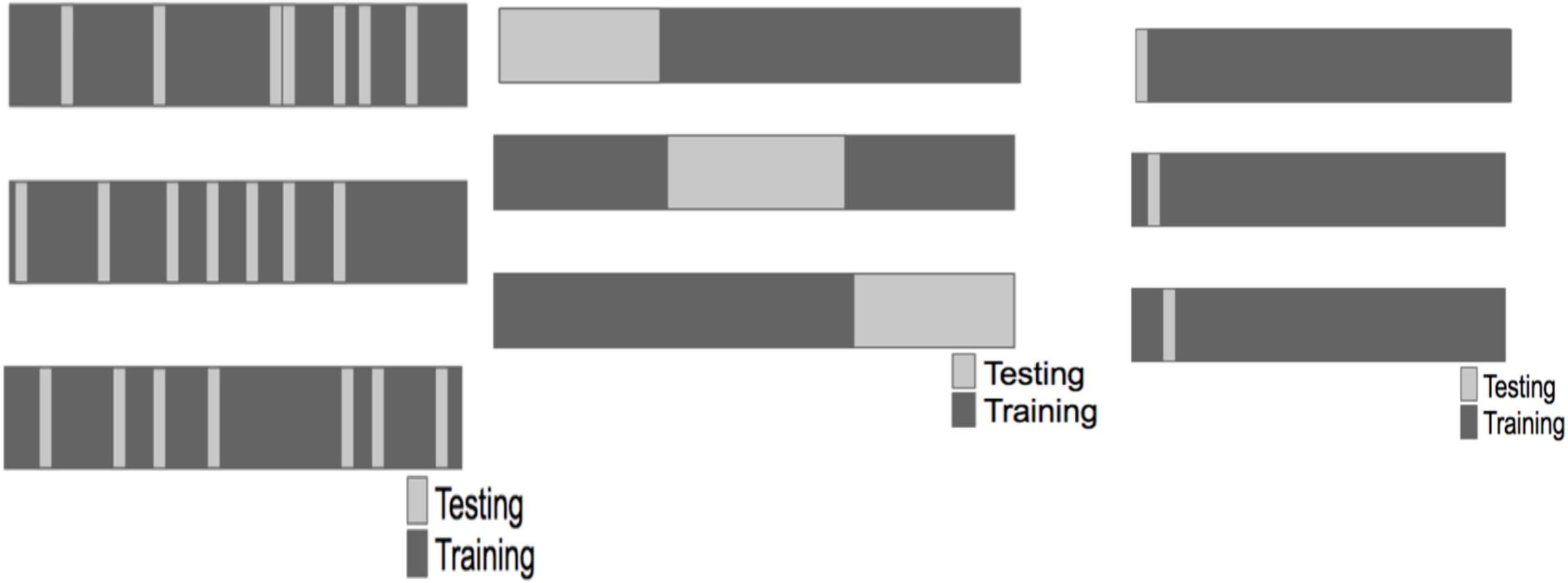
- Approach:
  - Use the training set
  - Split this training into training/test sets
  - Build a model on the training subset
  - Evaluate on the test subset
  - Repeat and average the estimated errors



Cross Validation is used for…

- Picking variables to include in a model
- Picking the type of prediction function to use
- Picking the parameters in the prediction function
- Comparing different predictors

Subsampling can be done in various ways..



Random Subsampling F-fold Leave One Out

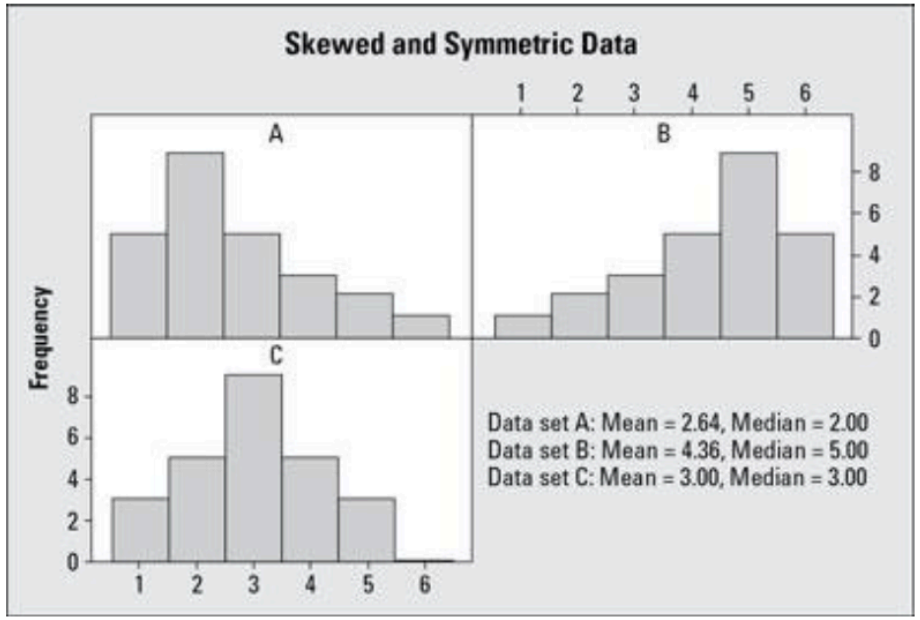
Some considerations

Considerations:

- For time series data, data must be used in 'chunks': blocks of contiguous time, can't do randomly
- For k-fold cross validation, larger k = less bias, more variance; smaller k = more bias, less variance
- For random sampling must be done without replacement
- Random sampling with replacement is the bootstrap - underestimates of the error; can be corrected, but it is complicated (use 0.632 Bootstrap rule)
- If you cross-validate to pick predictors estimate, you must estimate errors on independent data

Preprocessing

- Why do we need to preprocess?
  - Strange distributions, skewed vars in model-based predictors
  - Histogram & mean (mean not in centre shows skew)/standard deviation(watch out for highly inflated values!) can show skew in data



Preprocessing (standardization)

- Generally, we can standardize by subtracting the mean of a predictor (column in your data set) and then dividing by the standard deviation
- This will give a mean of zero and a standard deviation of one
- Generally, preprocessing should be performed for both the training and the testing dataset
- Preprocessing can also be incorporated in the model fitting stage, to be passed as an argument

How to preprocess continuous data into normal data

- If the dataset comprises continuous data (data measurable on a scale), it can be transformed into normal data (normally distributed data) using the Box-Cox transform.

Handle missing values

- Some algorithms cannot handle missing data
- Missing data can be imputed
- Subsequently, once the missing data has been handled, remember to standardize!
- REMINDER: Replace first, then standardize

On pre-processing

- Training and test must be processed in the same way
- Test transformations will likely be imperfect, especially if the test / training sets were collected at different times
- Be careful when transforming factor/categorical variables!

Covariate creation

- Covariates aka predictors, features – are variables you will use in your model to get at what outcome you care about
- 2 levels of covariate creation:
- Level 1: from raw data to covariate: eg raw email text, get the avg capital letters, # times 'you' appears, etc. - features that describe the raw email
- Level 2: transforming tidy covariates; ie no additional feature extraction steps needed

Level 1

- Level 1, Raw data to covariates
- Depends heavily on application
- The balancing act is summarization vs info loss
- Examples:
  - * text files: freq of words, phrases (Google ngrams), capital letters
  - * images: edges, corners, blobs, ridges (computer vision feature detection)
  - * webpages: number and type of images, position of elements, colors, videos (A/B testing, aka randomized trials in statistics)
  - * people: height, weight, hair color, sex, country of origin
- The more knowledge of the system you have, the better the job you will do
- When in doubt, err on the side of more features
- It can be automated, but use caution! May be important for training but won't generalize well for test

Level 2

- Level 2, tidy covariates to new covariates (functions on covariates)
- More necessary for some methods (regression, svms) than for others (classification trees)
- Only do on training set!
- Best approach through exploratory analysis (plotting / tables)
- New covariates should be added to data frames

Additional steps for covariates

- Common covariates to add, dummy variables
  - Basic idea: convert factor variables to indicator variables
  - E.g var jobclass = industrial, information; change so 2 vars, industrial, info, with values 0 or 1
- Removing zero covariates
  - Some variables have no variability at all - eg here, sex always male

Level 1 and Level 2 Covariates discussion

- Level 1 feature creation (raw to covariates):
  - Science is key; Google 'feature extraction for [data type]' eg images, voice, etc.
  - Err on overcreation of features
  - In some applications (images, voices), automated feature creation is possible / necessary
- Level 2 feature creation (covariates to new cov)
  - Create new covariates if you think they will improve fit
  - Use exploratory analysis on the training set for creating them
  - Be careful about overfitting!

Preprocessing with PCA (principal components analysis)

- In a dataset, we might have multiple quantitative variables, some highly correlated with others
- With PCA, the basic ideas are:
  - We might not need every predictor (like here, num415 and num857, do not need both)
  - A weighted combination of predictors might be better
  - We should pick this combo to capture the 'most info' possible
  - Benefits: reduced # of predictors, reduced noise due to averaging, reduce computation time

What of multivariate variables?

- Related problems:
- Multivariate variables X_1, ... X_n so $X_1 = (X_{11}, \ldots X_{1m})$
- Find a new set of multivariate vars that are uncorrelated and explain as much variance as possible (in example, use X and throw out Y and original vars)
- If you put all the vars together in one matrix, find the best matrix created with fewer variables (lower rank) that explains the original data
- The first goal is statistical, and the 2nd is data compression

PCA/SVD

- Related solutions - PCA / SVD
- SVD: if X is a matrix with each var in a column and each obs in a row, then the SVD is a 'matrix decomposition', X = UDV^T, (ie 3 matrices), where the columns of U are orthogonal (left singular vectors), the columns of V are orthogonal (right singular vectors) and D is a diagonal matrix (singular values)
- PCA: the principal components are equal to the right singular values if you first scale (subtract mean, divide by sd) the vars)

Summary

- There are many machine learning algorithms and techniques
- Machine learning models can range from a simple model to a complex model
- Always pick model that is easy to interpret