

Graduate Certificate in Intelligent Reasoning Systems

Reasoning and Knowledge Discovery from (large) Datasets

Dr. Barry Shepherd
Institute of Systems Science
National University of Singapore
Email: barryshepherd@nus.edu.sg



© 2021 NUS. The contents contained in this document may not be reproduced in any form or by any means, without the written permission of ISS, NUS, other than for the purpose for which it has been supplied.

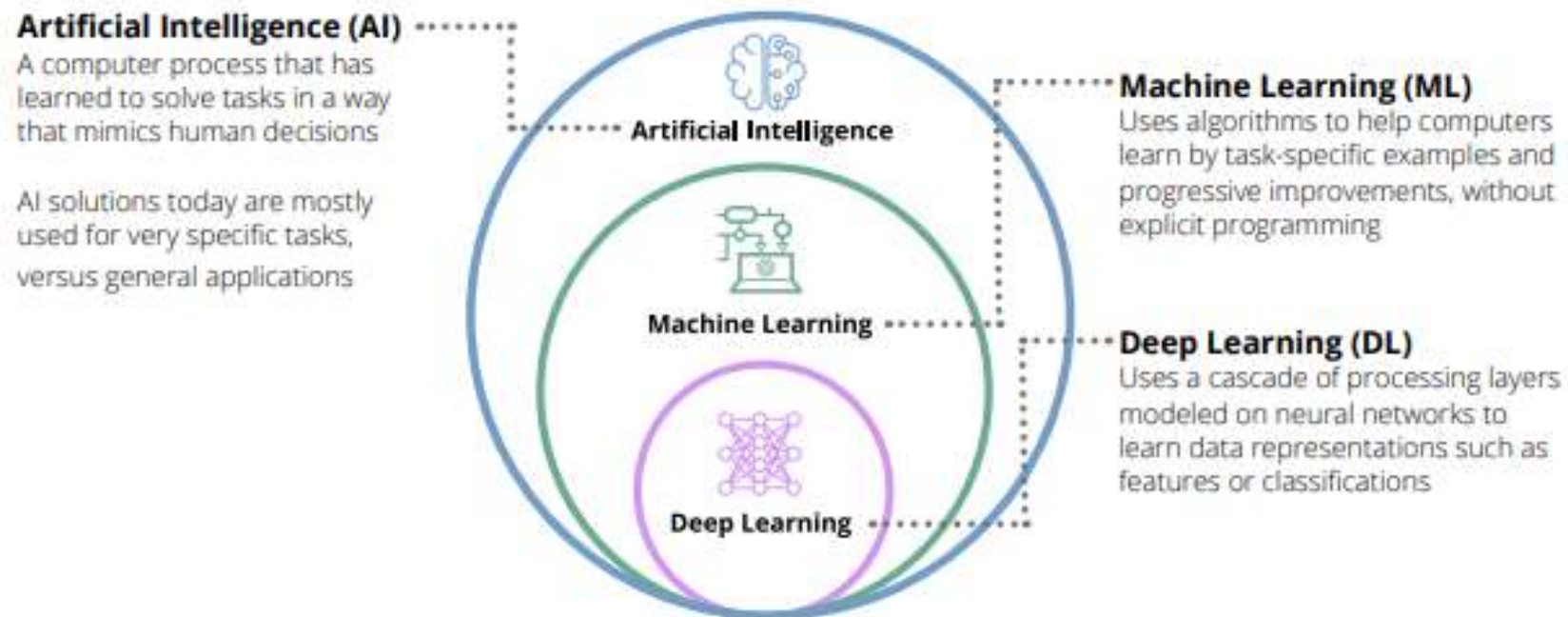
Agenda



Day	Topic
Day4 (am)	Introduction – basic concepts overview
Day4 (am)	Market Basket Analysis and Recommender Systems
Day4 (pm)	Similarity-Based Recommender Systems
Day5 (am)	Model-Based Recommender Systems
Day5 (pm)	Hybrid and Advanced Recommender Systems
Day5 (pm)	Course Assessment Quiz

Reasoning from large datasets

Increasingly, Reasoning Systems incorporate models built from large datasets, usually derived using Machine Learning (ML) and Deep Learning (DL).



In recent years ML, and in particular DL, has become synonymous with AI.

Data (lots) is crucial...

Data as the fuel of the future

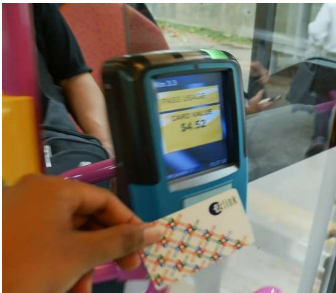
With the massive amounts of data being produced by the current "Big Data Era," we're bound to see innovations that we can't even fathom yet, and potentially as soon as in the next ten years. According to the experts, some of these will likely be deep learning applications.

Andrew Ng, the chief scientist of China's major search engine Baidu and one of the leaders of the Google Brain Project, shared a great analogy for deep learning with Wired Magazine: "I think AI is akin to building a rocket ship. You need a huge engine and a lot of fuel," he told Wired journalist Caleb Garling. "If you have a large engine and a tiny amount of fuel, you won't make it to orbit. If you have a tiny engine and a ton of fuel, you can't even lift off. To build a rocket you need a huge engine and a lot of fuel."

"The analogy to deep learning is that the rocket engine is the deep learning models and the fuel is the huge amounts of data we can feed to these algorithms."

- Andrew Ng (source: Wired)

Data Sources and Sizes



Events

Data Type	Size	Examples	Possible Applications
User-level data	Moderate	Account data, Registration data, Demographics, Interests, ...	Customer Relationship Management (CRM), Customer Retention Sale / Marketing Personalised Offers Recommendations Prediction Diagnosis Fraud detection
Events (discrete)	Huge	purchase transactions financial transactions healthcare visits bus trips web page views clicks, likes, downloads...	
Sensor Data (multi-media and often continuous)	Massive	Images, video, speech, sound, IOT...	Facial recognition Surveillance Inspection Chatbots Robots / Vehicles Smart city, Smart home

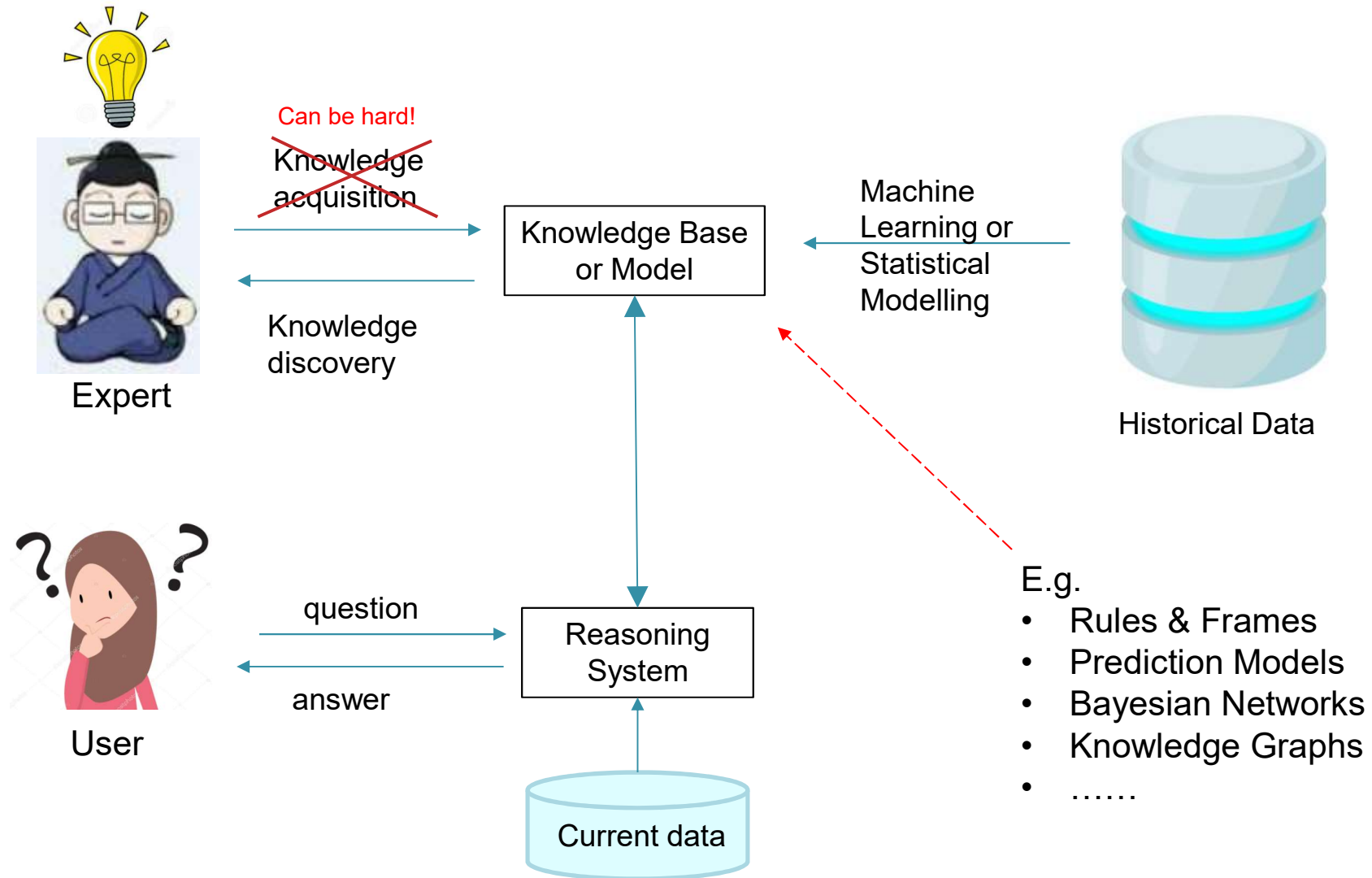


For ML, data size is a function of both length (#records) and width (#variables)



Multi-media, Sensory

Reasoning From Data versus From Expert Knowledge



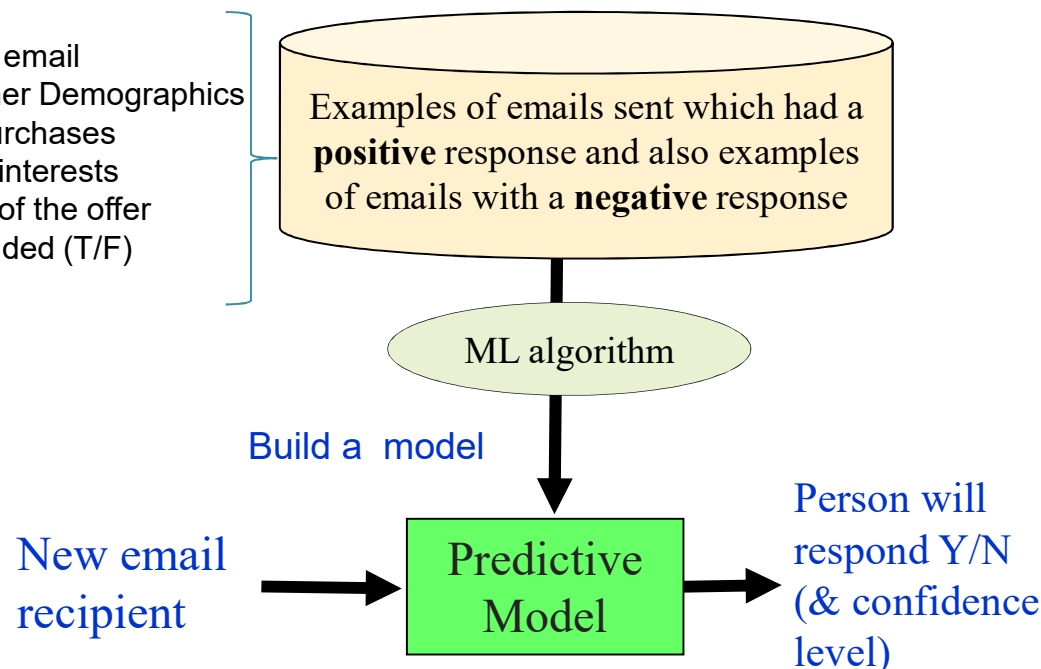
Example: Reasoning using Predictive Models

Given adequate historical data about an event we can use ML to build a model to predict if the event will occur in the future

- Will a user respond to a email or ad (e.g. buy a product) ?
- Will a customer close their account (churn)?
- Will a person respond to a drug?
- Will a patient stay long in the hospital?

E.g. Email Campaign Response Model

- Date of email
- Customer Demographics
- Past Purchases
- Known interests
- Details of the offer
- Responded (T/F)



Types of Predictive Model

- Linear regression
- Logistic regression
- Advanced Regression (e.g. GLM)
- Decision tree
- Rule sets
- Neural Network
- Deep Neural Network
- Support Vector Machine
- Naïve Bayes
- Nearest Neighbour
- Random Forests
- Gradient Boosting Machines
-

Data required for building a Predictive Model

- Historical data comprising a **relevant** set of input variables and a **suitable** output
- E.g. Predicting if Income Level is high (>\$50K) or low (<=\$50K):

<http://archive.ics.uci.edu/ml/machine-learning-databases/adult/>

age	education	marital	occupation	relationship	race	sex	hrs-per-wk	class
39	Bachelors	Never-married	Adm-clerical	Not-in-family	White	Male	40	<=50K
50	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	13	<=50K
38	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	40	<=50K
53	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	40	<=50K
28	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	40	<=50K
37	Masters	Married-civ-spouse	Exec-managerial	Wife	White	Female	40	<=50K
49	9th	Married-spouse-absent	Other-service	Not-in-family	Black	Female	16	<=50K
52	HS-grad	Married-civ-spouse	Exec-managerial	Husband	White	Male	45	>50K
31	Masters	Never-married	Prof-specialty	Not-in-family	White	Female	50	>50K
42	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	40	>50K
37	Some-college	Married-civ-spouse	Exec-managerial	Husband	Black	Male	80	>50K
30	Bachelors	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	40	>50K
23	Bachelors	Never-married	Adm-clerical	Own-child	White	Female	30	<=50K
32	Assoc-acdm	Never-married	Sales	Not-in-family	Black	Male	50	<=50K

Each record is an example of the event (or class) being predicted

The dataset should contain plenty of examples of all of the possible outputs

Input variables (aka fields)

Output variable (target)

Supervised Learning:

- Requires an output variable for training the model, this can be **categorical** or **numerical**.
- If **categorical** then the model is a **classification model** and the output is the **class value**.
- The number of class values should not be too large.
- Input variables must (usually) be numerical, one-hot coding can convert categories to numbers.

Acquire Valued Shoppers Challenge

Predict which shoppers will become repeat buyers



- You are given a minimum of a year of shopping history prior to each customer's incentive, as well as the purchase histories of many other shoppers (some of whom will have received the same offer). The transaction history contains all items purchased, not just items related to the offer.
- This challenge provides almost **350 million rows** of completely anonymised transactional data from over **300,000 shoppers**.
- How do you combine all of these records into a format suitable for modelling?
- What constitutes a training example?
- Usually, customer modelling datasets are created by aggregating customer transactions and activities and merging with account and demographic data

history

id - A unique id representing a customer

chain - An integer representing a store chain

offer - An id representing a certain offer

market - An id representing a geographical region

repeattrips - The number of times the customer made a repeat purchase

repeater - A boolean, equal to repeattrips > 0

offerdate - The date a customer received the offer

transactions

id - see above

chain - see above

dept - An aggregate grouping of the Category (e.g. water)

category - The product category (e.g. sparkling water)

company - An id of the company that sells the item

brand - An id of the brand to which the item belongs

date - The date of purchase

productsize - The amount of the product purchase (e.g. 16 oz of water)

productmeasure - The units of the product purchase (e.g. ounces)

purchasequantity - The number of units purchased

purchaseamount - The dollar amount of the purchase

offers

offer - see above

category - see above

quantity - The number of units one must purchase to get the discount

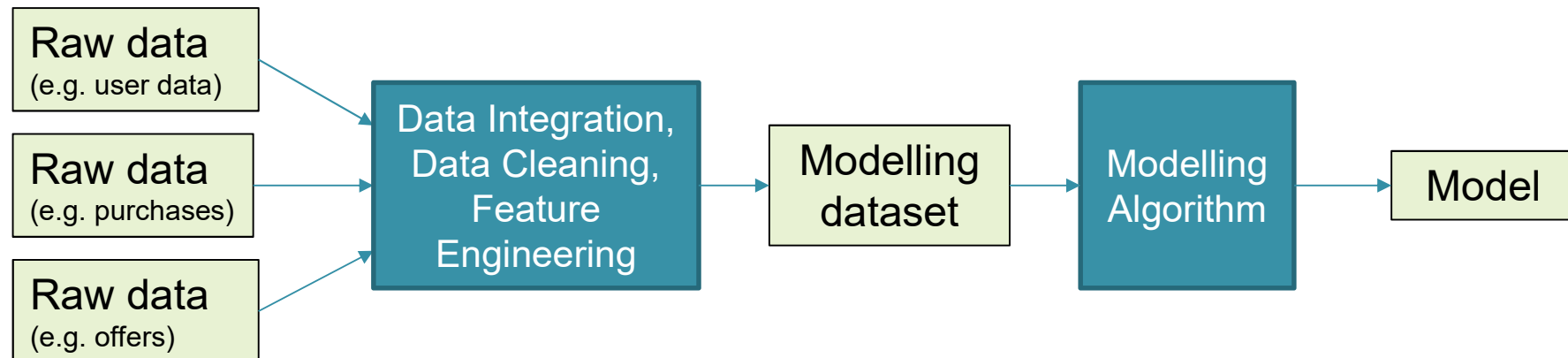
company - see above

offervalue - The dollar value of the offer

brand - see above

Creating a Modelling Dataset

- Raw data is usually not suitable for direct input into a model
- Data Preparation and Data Transformation are equally as important as the modelling algorithm (arguably more important). This process can be very time consuming.



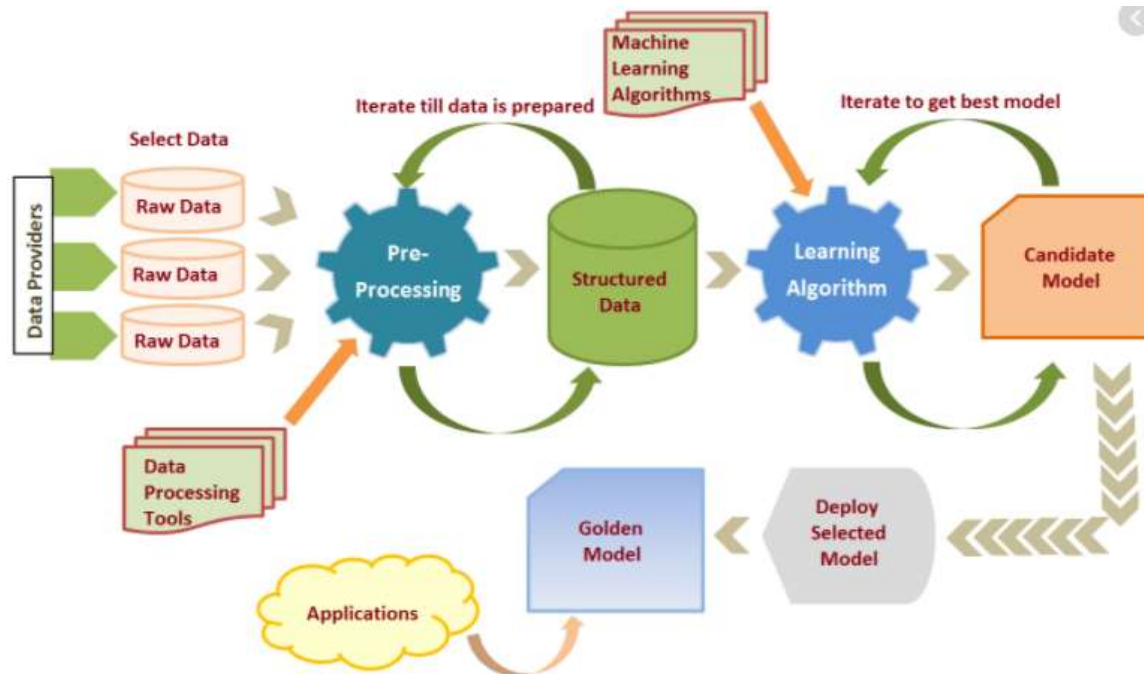
Feature Engineering typically refers to the process of transforming and combining raw variables to create new features with better predictive power, e.g:

Raw variables
 Unit price
 Number sold

→ Total cost

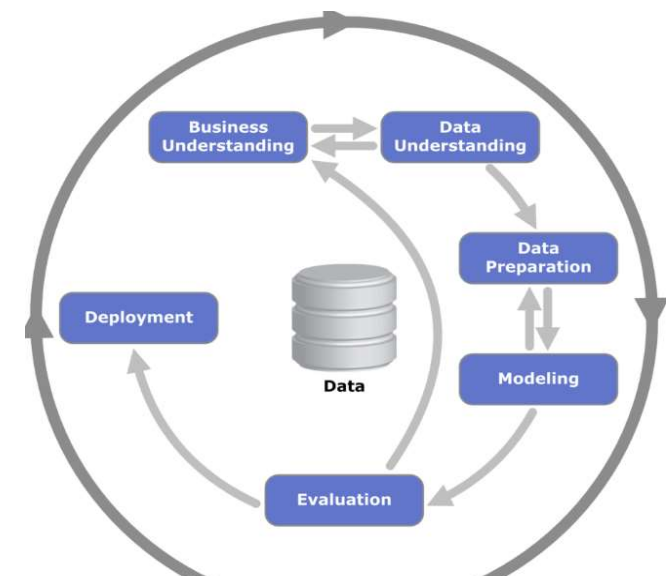
The (Complete) Model Building Process

- Also called the Data Mining Process - many methodologies exist, but all similar. Iteration is crucial – keep refining the data and testing the model until its good enough for deployment



<https://elearningindustry.com/machine-learning-process-and-scenarios>

CRISP methodology

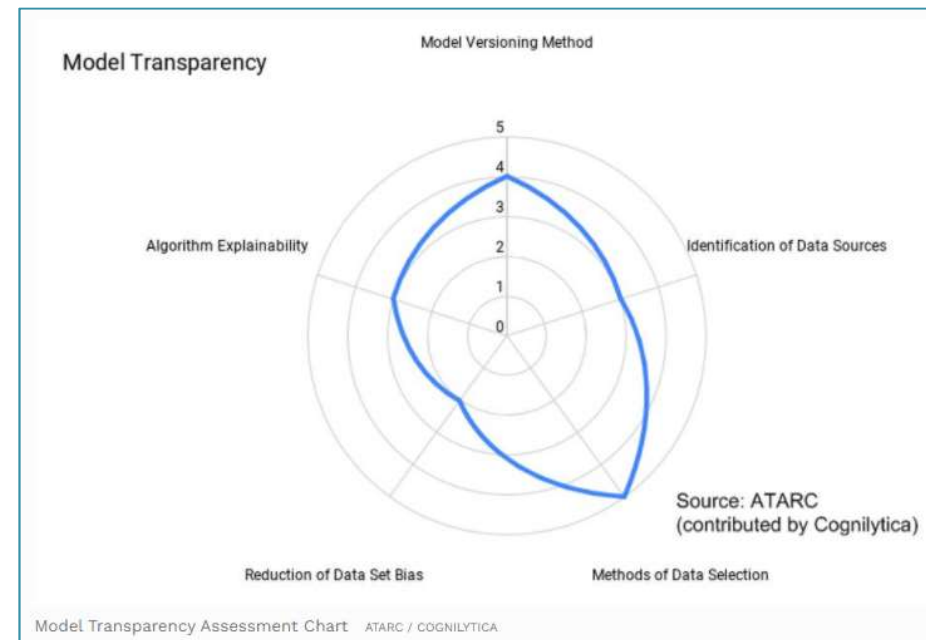


Many emphasise Business and Data Understanding as crucial for project success – obtaining good model accuracy against a test set is not enough - the model must fit the situation in which it is to be deployed (a good solution to the wrong question is of no use)

Note : the model does not need to be executable, it can be a representation of knowledge discovered without necessarily modelling a specific outcome (knowledge discovery)

Knowledge Discovery & Transparency

- Knowledge Discovery in Databases (KDD)** is the nontrivial process of identifying valid, novel, potentially useful, and ultimately *understandable* patterns or relationships within a dataset in order to *make important decisions* (Fayyad et al, 1996). The term KDD is often used interchangeably with Data Mining
- Transparency and Trust** - for some AI applications there is no need for the user to understand how a decision is made as long as the result/decision is correct. But in many applications we need transparency in the decision making process, we need to *understand*:
 - Why did you recommend that drug?
 - Why did you show me that ad?
 - How do I know the system is trustworthy, does not have a hidden bias?



Different Aspects of Model Transparency

<https://www.forbes.com/sites/cognitiveworld/2020/05/23/towards-a-more-transparent-ai>

Explainable AI

- One way to gain explainability and trust in AI systems is to use machine learning algorithms that are inherently explainable:

- Decision tree and Decision rule learning
- Association rule learning
- Similarity-based Reasoning (Nearest Neighbour algorithms)
- *Linear Regression*
- *Logistic Regression*
- *Naïve Bayes*
- *Clustering*

covered in
PRS Cert

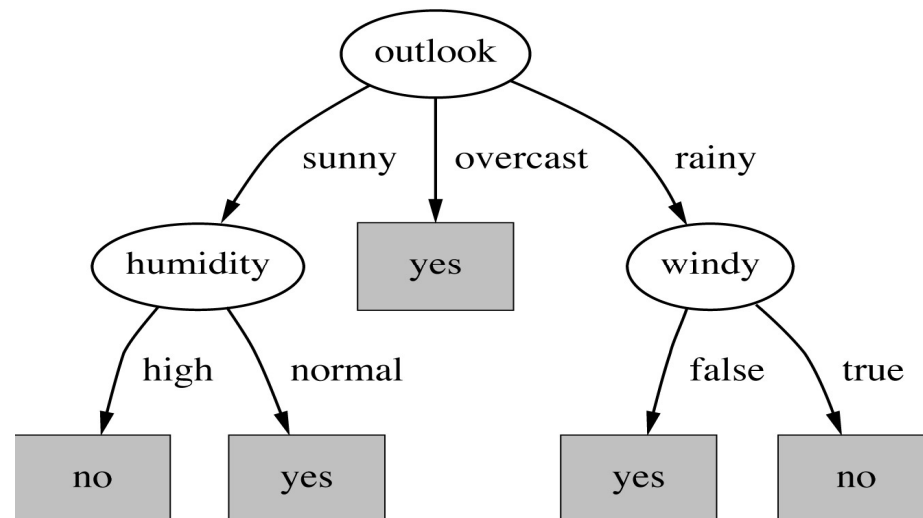
- But increasingly, many AI models are black boxes. Explainable AI is an emerging field in machine learning that aims to address how black box decisions of AI systems are made
 - e.g. LIME takes a black box model, probes it to generate a new data set and then builds an interpretable model (e.g. a decision tree) from this new data set



e.g. see <https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime-5adc035109b5>

Decision Trees & Rules

- A decision tree is a “flow-chart-like” tree structure, easily understood
- Typically used for classification tasks and making predictions
- E.g. Should I play tennis today?

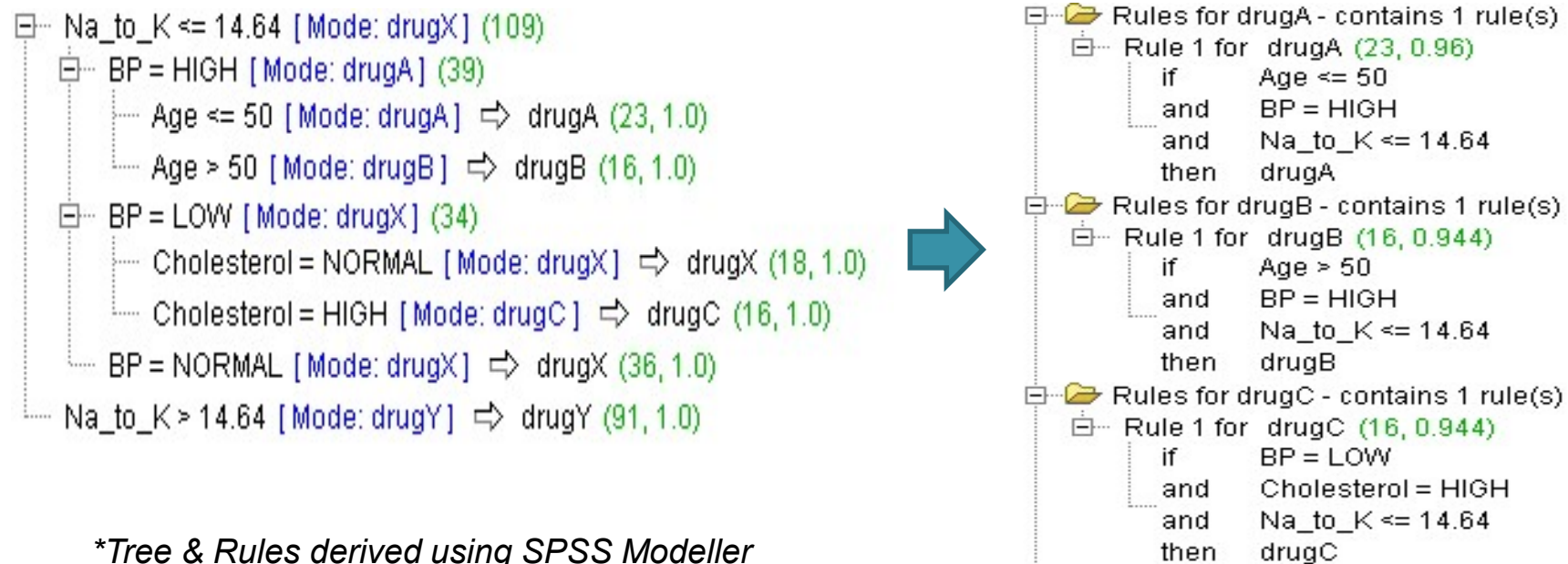


- Make a classification (prediction) based on the values of known attributes
- The **internal nodes** perform tests on these attributes. Branches represent the results of tests
- The **leaf nodes** represent class labels
- A new case is classified by following a path to a leaf node

Many algorithms exist for generating decision trees from datasets
 e.g. C5, C4.5, CART, CHAID... (we examine in detail in the PRS cert)

Decision Trees can be converted into Rules

- Potentially makes the knowledge easier to understand
- Can result in fewer rules than tree nodes (after rule simplification)



**Regardless of transparency, most applications usually select the modelling technique that performs best (most accurate) when tested on a test set*

Basic Python Code to Build Decision Tree (Income Level Prediction)

```
import pandas as pd
from sklearn import tree

# load the data
data = pd.read_csv('adult-train.csv', skipinitialspace=True)

# preprocess the data
# (convert the categorical variables into numbers using one-hot encoding)
catvars = ('workclass', 'education', 'marital', 'occupation',
           'relationship', 'race', 'sex', 'native-country')
for v in catvars:
    onehot = pd.get_dummies(users[v], drop_first=True)
    users.drop(v, inplace=True, axis = 1)
    users = users.join(onehot)

# define input and output variables
target = 'class' # the name of the target field
inputs = list(set(data.columns) - set([target]))

# build the tree
tclf = tree.DecisionTreeClassifier()
tclf.fit(data[inputs], data[target])

# view the tree as text
text_representation = tree.export_text(tclf, feature_names=inputvars)
print(text_representation)
```

This basic data prep is generic, it would be the same if another model type were used instead of a decision tree (e.g. logistic regression)

One-Hot Encoding

- Converts a categorical variable with N values into N binary (0,1) variables

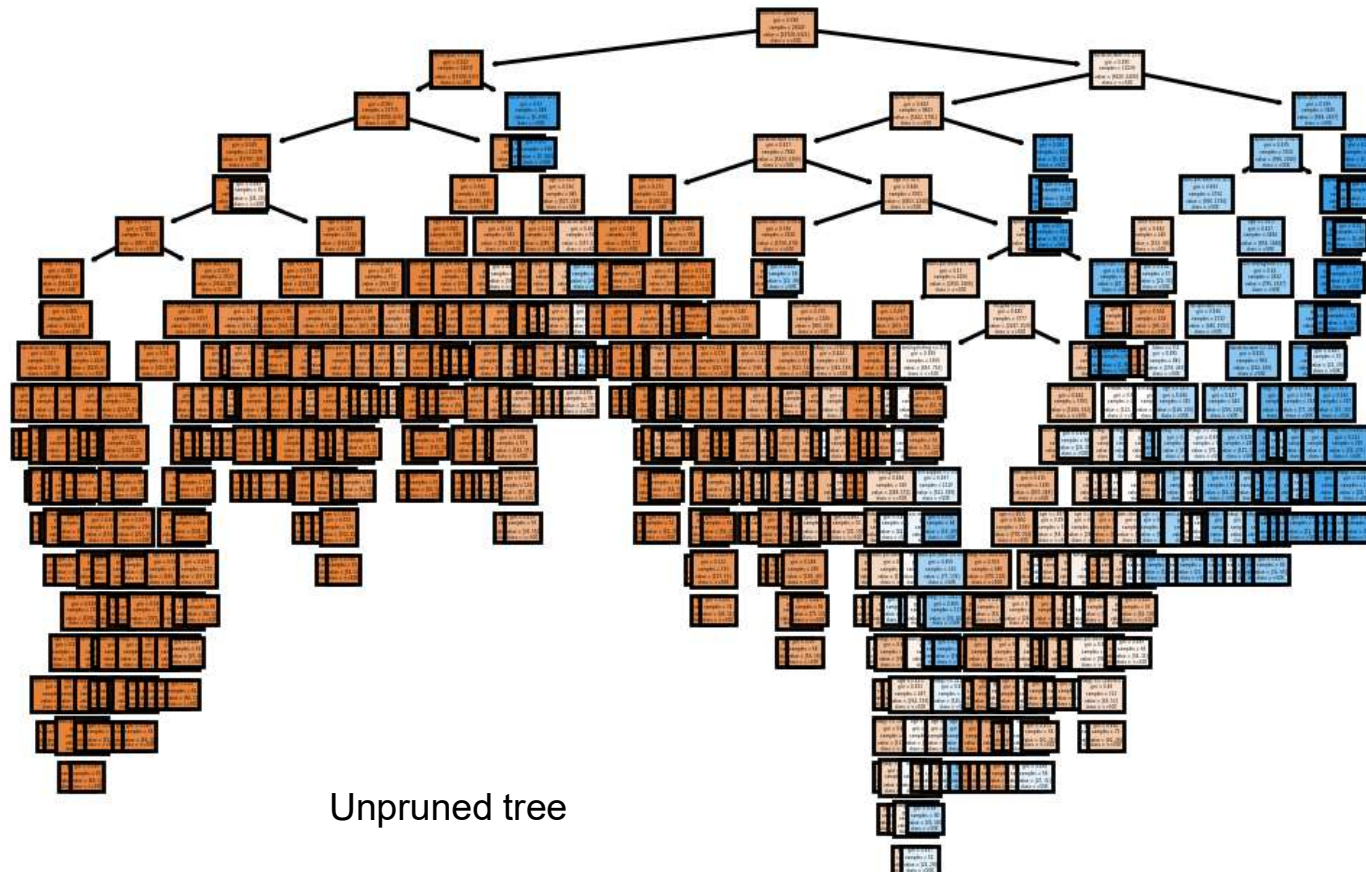
Pet	Cat	Dog	Turtle	Fish
Cat	1	0	0	0
Dog	0	1	0	0
Turtle	0	0	1	0
Fish	0	0	0	1
Cat	1	0	0	0

In practice, we can make do with N-1 binary variables, e.g. drop the Cat Column!

`onehot = pd.get_dummies(users[v], drop_first=True)`

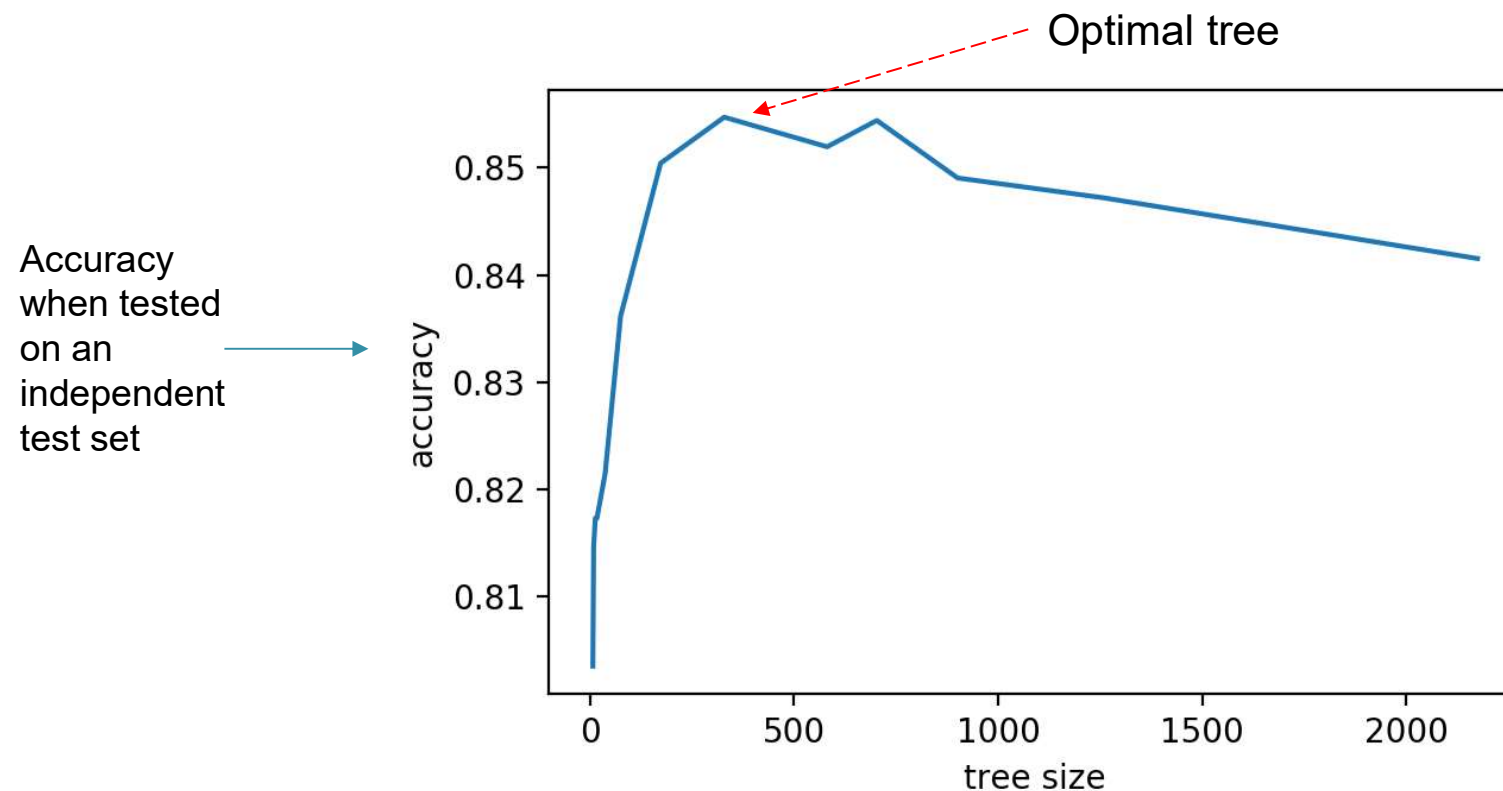
Decision Tree Pruning

- Raw trees are often too big to be understandable
- The unpruned tree for the income prediction (adult) dataset has over 7000 nodes!
- Pruning can often improve accuracy by reducing overfitting, and also make the tree easier to understand



Decision Tree Pruning

- Overfitting occurs when the tree models the training data precisely – the accuracy on the training data can be 100% but accuracy on test data may be much worse
- Accuracy often improves with some pruning, then usually worsens with too much
- E.g. results of varying pruning on the income level prediction dataset



-
- ```
graph TD
 Node0["Married-civ-spouse <= 0.5
gini = 0.368
samples = 26049
value = [19728, 6321]
class = <=50K"]
 Node1["education-num <= 11.5
gini = 0.122
samples = 14021
value = [13108, 913]
class = <=50K"]
 Node2["education-num <= 12.5
gini = 0.495
samples = 12028
value = [6620, 5408]
class = <=50K"]
 Node3["age <= 35.5
gini = 0.062
samples = 10672
value = [10332, 340]
class = <=50K"]
 Node4["gini = 0.284
samples = 3349
value = [2776, 573]
class = <=50K"]
 Node5["education-num <= 9.5
gini = 0.443
samples = 8403
value = [5622, 2781]
class = <=50K"]
 Node6["gini = 0.399
samples = 3625
value = [998, 2627]
class = >50K"]
 Node7["age <= 23.5
gini = 0.024
samples = 6577
value = [6497, 80]
class = <=50K"]
 Node8["gini = 0.119
samples = 4095
value = [3835, 260]
class = <=50K"]
 Node9["gini = 0.385
samples = 5206
value = [3850, 1356]
class = <=50K"]
 Node10["gini = 0.494
samples = 3197
value = [1772, 1425]
class = <=50K"]
 Node11["gini = 0.004
samples = 3276
value = [3269, 7]
class = <=50K"]
 Node12["gini = 0.043
samples = 3301
value = [3228, 73]
class = <=50K"]

 Node0 --> Node1
 Node0 --> Node2
 Node1 --> Node3
 Node1 --> Node4
 Node2 --> Node5
 Node2 --> Node6
 Node3 --> Node7
 Node3 --> Node8
 Node5 --> Node9
 Node5 --> Node10
 Node7 --> Node11
 Node7 --> Node12
```

Page 20



# Association Rules

- Data that represents things that happen at the same time or in the same context can be mined to find associations
- Finding association between items in a shopping baskets is a common application (aka Market Basket Analysis).
- The learned rules can be used to guide manual decisions (e.g. how to layout the supermarket shelves) – or can be used in a rule-based system (e.g. to make recommendations)
- Sort by Support and Confidence to see the most important rules

| Association Rules | Support (%) | Confidence (%) | Lift |
|-------------------|-------------|----------------|------|
| Soap → Perfume    | 30          | 75             | 1.87 |
| Perfume → Soap    | 30          | 75             | 1.87 |
| Bread → Jam       | 40          | 80             | 1.6  |
| Jam → Bread       | 40          | 80             | 1.6  |
| Chocolate → Bread | 30          | 75             | 1.5  |
| Beer → Snacks     | 30          | 100            | 2    |

# Similarity-Based Reasoning

Aka Analogical Reasoning – if two or more things are similar in some respects, they are probably also similar in some further respect

## Nearest Neighbour Systems

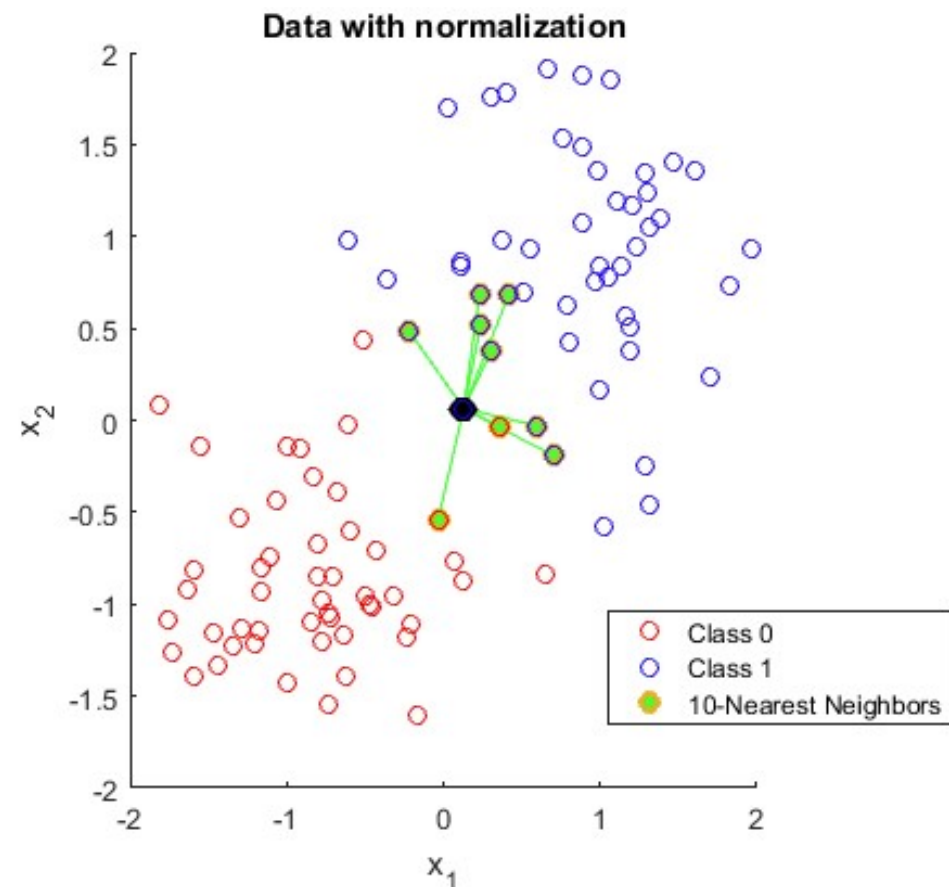
Represent situations (cases) as vectors.

To reason about a new case :

- Find the K most similar neighbours to the new case

Make a decision by either:

- taking the majority vote  
(for classification)
- averaging over the K neighbours  
(for numerical prediction)



# Agenda



| Day       | Topic                                          |
|-----------|------------------------------------------------|
| Day4 (am) | Introduction – basic concepts overview         |
| Day4 (am) | Market Basket Analysis and Recommender Systems |
| Day4 (pm) | Similarity-Based Recommender Systems           |
| Day5 (am) | Model-Based Recommender Systems                |
| Day5 (pm) | Hybrid and Advanced Recommender Systems        |
| Day5 (pm) | Course Assessment Quiz                         |

# Another Prediction Scenario...

- Best Buy Challenge:** Predict which items (skus) will get clicked when a user does a search on the BestBuy website / app

| user           | sku     | category     | query               | click_time | query_time |
|----------------|---------|--------------|---------------------|------------|------------|
| 585c9ba287c96  | 2032076 | abcat0701002 | gears of war        | 22:56.1    | 21:42.9    |
| d3626c4b56ff5c | 9854804 | abcat0701002 | Gears of war        | 35:42.2    | 35:33.2    |
| d3626c4b56ff5c | 2670133 | abcat0701002 | Gears of war        | 36:08.7    | 35:33.2    |
| d3626c4b56ff5c | 9984142 | abcat0701002 | Assassin creed      | 37:23.7    | 37:00.0    |
| 5450f7be1df336 | 2541184 | abcat0701002 | dead island         | 15:34.3    | 15:26.2    |
| ace315dbdbef5  | 3046066 | abcat0701002 | Rocksmith           | 44:36.8    | 44:22.9    |
| 329b4dc052268  | 2977637 | abcat0701002 | Nba n2k             | 10:06.5    | 09:33.5    |
| 838c635196205  | 2670133 | abcat0701002 | Call of duty        | 05:40.9    | 05:06.4    |
| 48e3e5458ed52  | 9328943 | abcat0701002 | rock band           | 02:54.2    | 02:44.2    |
| ad7a0a3d7fce6  | 1180104 | abcat0701002 | xbox                | 39:51.7    | 36:40.2    |
| ad7a0a3d7fce6  | 2598445 | abcat0701002 | xbox                | 41:16.2    | 36:40.2    |
| 25507e5e95633  | 1563461 | abcat0701002 | Crysis 2            | 47:23.6    | 46:56.0    |
| 7813088660359  | 9854804 | abcat0701002 | Gears of war 3      | 54:06.4    | 53:31.5    |
| ca198f6e7372c1 | 1012721 | abcat0701002 | Sims                | 07:58.0    | 07:39.3    |
| 1bd0fc6e1cc79  | 2173065 | abcat0701002 | batman arkham city  | 52:23.0    | 51:55.4    |
| 9ec3438e8785c  | 1228939 | abcat0701002 | rage                | 29:37.8    | 29:22.3    |
| ba2dc188542e2  | 2541184 | abcat0701002 | Deda island         | 03:57.5    | 03:38.5    |
| 54971eb300862  | 2173065 | abcat0701002 | batman arkham city  | 35:14.5    | 35:02.7    |
| f4b3047aef3a2  | 2375195 | abcat0701002 | Madden 12           | 16:19.2    | 15:17.6    |
| fd09424f35c760 | 2480232 | abcat0701002 | lice madness return | 47:43.3    | 47:32.7    |
| f6126e66308c0  | 2670133 | abcat0701002 | modern warfare 3    | 22:00.9    | 21:45.3    |



<https://www.kaggle.com/c/aim-sf-chapter-hackathon-big>

Issues:

1000's of skus

Few input variables

# Reasoning with 1000's of outcomes

## 12 million products

**Amazon sells** more than 12 million **products**.

In its quest to be all things to all people, **Amazon** has built an unbelievable catalog of more than 12 million **products**, books, media, wine, and services. If you expand this to **Amazon** Marketplace sellers, as well, the number is closer to more than 350 million **products**.

<http://rejoiner.com/resources/amazon-recommendations-secret-selling-online/>

Each month **more than 197 million people** around the world get on their devices and visit Amazon.com.

That's more than the entire population of Russia.

<https://www.bigcommerce.com/blog/amazon-statistics/>



- How do we reason when there are huge numbers of outcomes?
- Recommendations Systems are a class of reasoning systems that must solve this problem



# Recommender Systems

- Very ubiquitous, an essential part of many websites

Customers who bought this item also bought

|                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  <p><b>The Hundred-Page Machine Learning Book</b><br/>         › Andriy Burkov<br/>         ★★★★★ 74<br/>         Kindle Edition<br/>         \$34.99</p> |  <p><b>Feature Engineering for Machine Learning: Principles and...</b><br/>         › Alice Zheng<br/>         ★★★★★ 8<br/>         Kindle Edition<br/>         \$29.99</p> |  <p><b>Hands-On Recommendation Systems with Python: Start building powerful and...</b><br/>         › Rounak Banik<br/>         Kindle Edition<br/>         \$14.59</p> |  <p><b>Designing Data-Intensive Applications: The Big Ideas Behind Reliable,...</b><br/>         › Martin Kleppmann<br/>         ★★★★★ 145<br/> <b>#1 Best Seller</b> in Database Management Systems<br/>         Kindle Edition<br/>         \$28.68</p> |  <p><b>Statistical Methods for Recommender Systems</b><br/>         Deepak K. Agarwal<br/>         ★★★★★ 4<br/>         Kindle Edition<br/>         \$36.49</p> |  <p><b>Hands-On Machine Learning with Scikit-Learn and TensorFlow:...</b><br/>         › Aurélien Géron<br/>         ★★★★★ 267<br/> <b>#1 Best Seller</b> in Natural Language Processing<br/>         Kindle Edition<br/>         \$29.99</p> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|





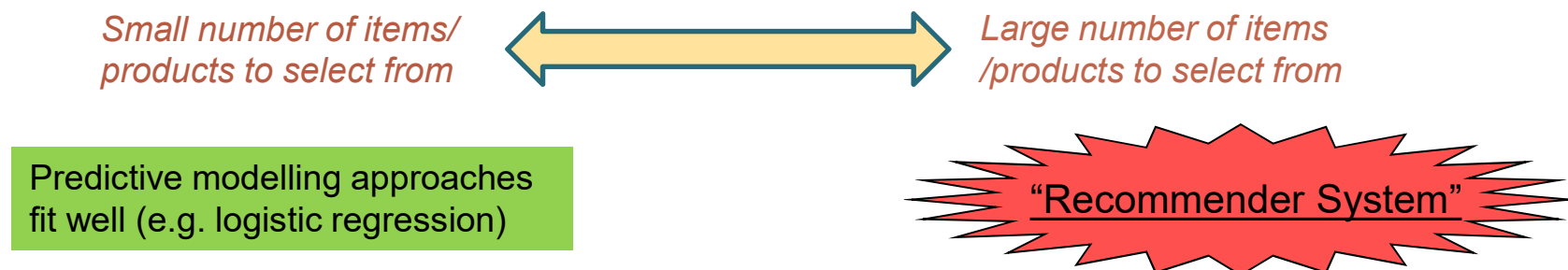
# Scoping Recommender Systems

The phrase **Recommender System** is very broad – it *could be* used to describe a very wide range of applications, including common predictive modelling applications. (Just replace the word *predicting* with the word *recommending*!)

For example:

- Recommending who to mail ads/promotions to (targeted marketing)
- Recommending the best drug or treatment for a patient
- Recommending the best business partner
- Recommending if a device/equipment needs preventative maintenance (T/F)
- Recommending which insurance claim to investigate further (possible fraud)

One distinguishing feature is the number of possible recommendations that can be made:



# Market Basket Analysis

- Much can be done with simple analysis of things that happen together:
  - Items bought together (in same shopping basket)
  - Items viewed in the same browsing session
  - Items bought by same users but on different days

- Requires a list of transactions
- E.g. transactions at a convenience store
  - Transaction1: frozen pizza, cola, milk
  - Transaction2: milk, potato chips
  - Transaction3: cola, frozen pizza
  - Transaction4: milk, peanuts
  - Transaction5: cola, peanuts
  - Transaction6: cola, potato chips, peanuts



*Items that occur together (e.g. are in the same basket) are called an item-set. If an item-set has a large count (a frequent item-set) then there is a potential association between the items in the item set*

# Cooccurrence Counting

- A simple approach to MBA is to cross-tabulate into a table to show how often each possible pair of products were bought together:
- The table (the co-occurrence matrix) is symmetrical since the items in each basket have no temporal ordering. The diagonal shows the total number of times the item was bought.

|        | Pizza | Milk | Cola | Chips | P/nuts |
|--------|-------|------|------|-------|--------|
| Pizza  | 2     | 1    | 2    | 0     | 0      |
| Milk   | 1     | 3    | 1    | 1     | 1      |
| Cola   | 2     | 1    | 4    | 1     | 2      |
| Chips  | 0     | 1    | 1    | 2     | 1      |
| P/nuts | 0     | 1    | 2    | 1     | 3      |

Pizza buyers (2) always also buy cola (2)

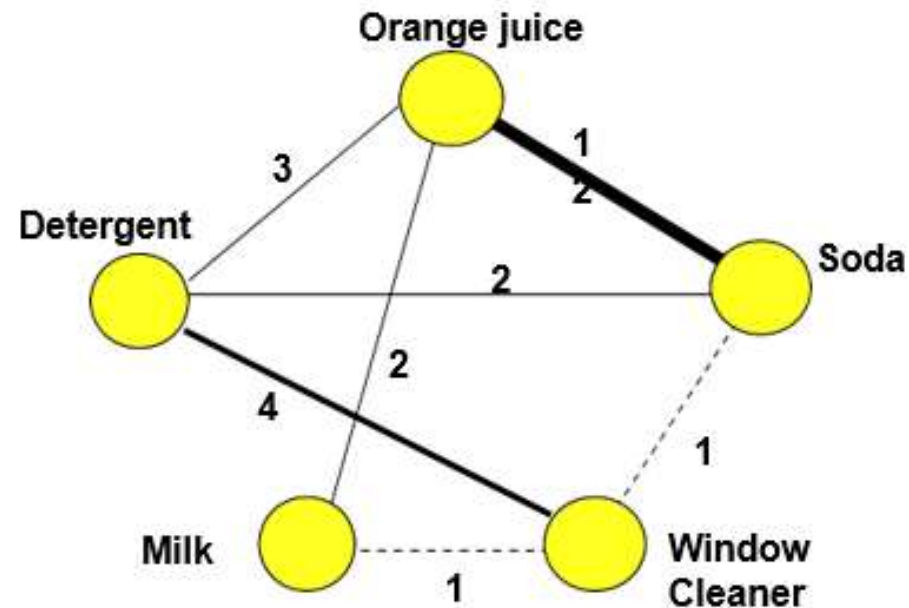
But Cola buyers (4) do not always buy pizza (2)

Milk sells well with everything

Peanut buyers (3) often buy cola (2)

# Link Analysis


- Cooccurrences can be visualized using link analysis
- Nodes represent items, thickness of joining line indicates number of times they occurred together



# Association Rule Learning

- Cooccurrence counting is not viable for large datasets and large item-sets
- Algorithms such as Apriori (Agrawal et al, '93) use heuristics to reduce the combinatorial size of the search space
  - If an item-set is frequent, then all of its subsets must also be frequent
  - The user specifies minimum rule support and rule confidence
- The found associations are expressed as rules, e.g.

*If buy pizza then also buy cola*



LHS (left-hand side)      RHS (right-hand side)

- In general, association rules can have multiple items in the LHS or RHS

*If Coffee and Milk then Sugar*

*If BBQ charcoal then Sausages and Steak*

**Note: rules indicate co-occurrence, not causality or a sequence over time**

# Association Rule Metrics

- Rule Support
  - The proportion of transactions that contain the item set (LHS + RHS items)
- Rule Confidence
  - The proportion of rule firings that are correct predictions
  - Support for combination (LHS & RHS) / Support for condition (LHS)

$$= \frac{\text{\#transactions containing all items in the rule (LHS and RHS)}}{\text{\#transactions containing all items in the rule condition (LHS)}}$$

|                                                                                  |                                      |                  |
|----------------------------------------------------------------------------------|--------------------------------------|------------------|
| e.g. consider<br>page-views<br>in a web<br>browsing<br>session as<br>the baskets | session1: home, news, sport          | home → news      |
|                                                                                  | session2: finance, news              | Support = 4/7    |
|                                                                                  | session3: fashion, home              | Confidence = 4/6 |
|                                                                                  | session4: news, finance, home        |                  |
|                                                                                  | session5: sport, home, finance       | news → home      |
|                                                                                  | session6: fashion, home, news        | Support = 4/7    |
|                                                                                  | session7: home, finance, news, sport | Confidence = 4/5 |

Note: news→home does not have same *confidence* as home→news

To get intuition consider: whisky->coke (often), but coke->whisky (less)



# Association Rule Metrics

$$\text{Rule lift} = \frac{\text{Support (LHS \& RHS)}}{\text{Support (LHS) * Support (RHS)}} = \text{the ratio of the observed support to that expected if LHS and RHS were independent}$$

If lift = 1 then this implies that the probability of occurrence of X and Y are independent (no association)  
 If the lift is > 1, then LHS and RHS are dependent on each other (one makes the other more likely)  
 If the lift is < 1, then one (LHS or RHS) has a negative effect on presence of other

session1: home, news, sport  
 session2: finance, news  
 session3: fashion, home  
 session4: news, finance, home  
 session5: sport, home, finance  
 session6: fashion, home, news  
 session7: home, finance, news, sport

home → news or news → home

$$\text{Lift} = (4/7) / ((6/7)*(5/7)) \\ = 0.57 / 0.61 = 0.93$$

[https://en.wikipedia.org/wiki/Association\\_rule\\_learning#Lift](https://en.wikipedia.org/wiki/Association_rule_learning#Lift)

# Association Mining Applications

- Items purchased on a credit card (e.g. rental cars, hotel rooms) give insight into the next product the customer may buy
- Optional services bought by telecom customers (call waiting, forwarding, auto-roam etc) show how best to bundle these services
- Banking services used by retail customers (investment services, car loans, home loans, money market accounts etc) show possible cross-sells
- Unusual combinations of insurance claims may indicate fraud
- May find associations between certain combinations of medical treatments and complications in medical patients

# Issues with Association Rules

- Can generate a huge number of rules, often trivial and with repetition:  
*If coffee and milk then sugar*  
*If milk and sugar then coffee*  
*If sugar and coffee then milk*
- Define minimum support and minimum confidence for rule pruning/filtering to get “strong” rules
- Analyst must make decisions regarding validity & importance of rules to be accepted (subjective)

| Consequent    | Antecedent              | Support % | Confidence % |
|---------------|-------------------------|-----------|--------------|
| frozenmeal    | cannedveg               | 30.300    | 57.100       |
| beer          | cannedveg               | 30.300    | 55.120       |
| cannedveg     | frozenmeal              | 30.200    | 57.280       |
| beer          | frozenmeal              | 30.200    | 56.290       |
| frozenmeal    | beer                    | 29.300    | 58.020       |
| confectionery | wine                    | 28.700    | 50.170       |
| wine          | confectionery           | 27.600    | 52.170       |
| beer          | cannedveg<br>frozenmeal | 17.300    | 84.390       |
| cannedveg     | frozenmeal<br>beer      | 17.000    | 85.880       |
| frozenmeal    | cannedveg<br>beer       | 16.700    | 87.430       |

*e.g. rules sorted by support*

# Example: MSNBC Website Analysis

- Data from the Pagview log of MSNBC.com (a news site) on 28 Sep'99
  - Raw data ~ Datetime, URL, cookieID, IPaddress, UserAgent(browser) etc...

- Preprocessing
  - Page view categorization** – the pages viewed (URLs) were first converted into topic categories
  - Grouping by unique user** - each data row shows the sequence of pageview categories for one user on that day

Codes for the msnbc.com page categories

| category  | code | category | code | category  | code |
|-----------|------|----------|------|-----------|------|
| frontpage | 1    | misc     | 7    | summary   | 13   |
| news      | 2    | weather  | 8    | bbs       | 14   |
| tech      | 3    | health   | 9    | travel    | 15   |
| local     | 4    | living   | 10   | msn-news  | 16   |
| opinion   | 5    | business | 11   | msn-sport | 17   |
| On-air    | 6    | sports   | 12   |           |      |

% Sequences:

```

6
1 1
6
6 7 7 7 6 6 8 8 8 8
6 9 4 4 4 10 3 10 5 10 4 4 4
1 1 1 11 1 1 1
12 12

```

Number of users: 989,818  
 Average number of visits per user: 5.7  
 No. of URLs per category: 10 to 5000

# Example MSNBC Rules\*

- Association Rule Examples

|                                   |               |        |
|-----------------------------------|---------------|--------|
| on-air & business & sports & bbs  | --> frontpage | 86.22% |
| news & tech & misc & bbs          | --> frontpage | 86.18% |
| on-air & misc & business & sports | --> frontpage | 86.16% |
| tech & misc & travel              | --> on-air    | 86.09% |
| tech & living & business & sports | --> frontpage | 86.08% |
| news & living & sports & bbs      | --> frontpage | 85.99% |
| misc & business & sports          | --> frontpage | 85.79% |

- Sequence Rules Examples

|                                  |       |
|----------------------------------|-------|
| on-air → news                    | 1.51% |
| news → frontpage → news          | 1.49% |
| local → news                     | 1.46% |
| frontpage → frontpage → business | 1.35% |
| news → sports                    | 1.33% |
| news → bbs                       | 1.23% |
| health → local                   | 1.16% |

Are these rules  
likely to be  
useful?

(\*Frequent Pattern Mining in Web Log Data, Ivancsy, Vajk, 2006.  
Using their own association and sequence finding algorithms )



# What Level of Detail to Recommend?

- At what level of detail should we make the recommendations?
- E.g. Should we look for associations between:

*frozen pizza, chips, cola, milk*

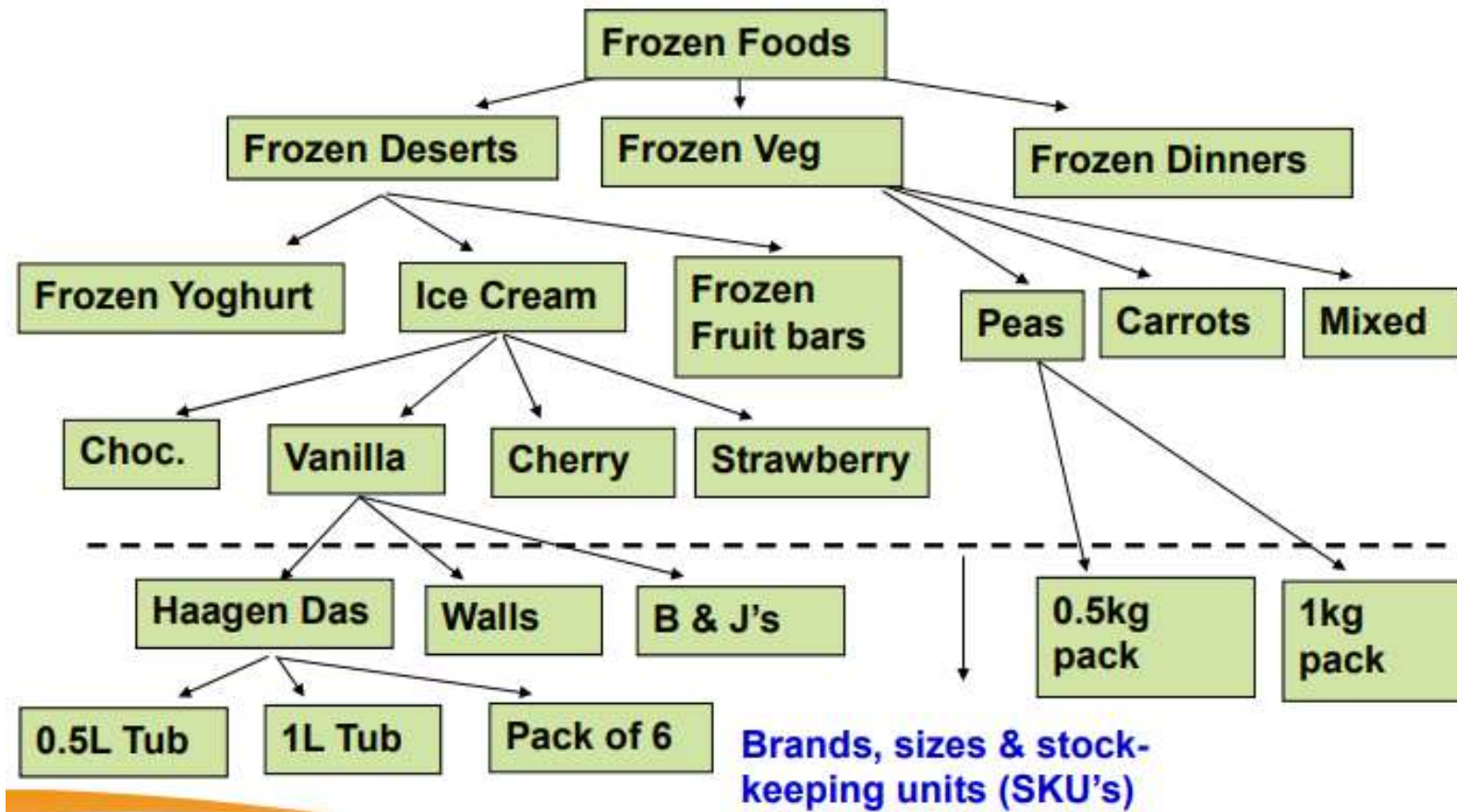
Or:

*cheese pizza, tuna pizza, salami pizza, vegetable pizza, chips, cola, milk*

Too much detail can  
generate an overload  
of associations

Too little detail may  
yield obvious and non-  
actionable results

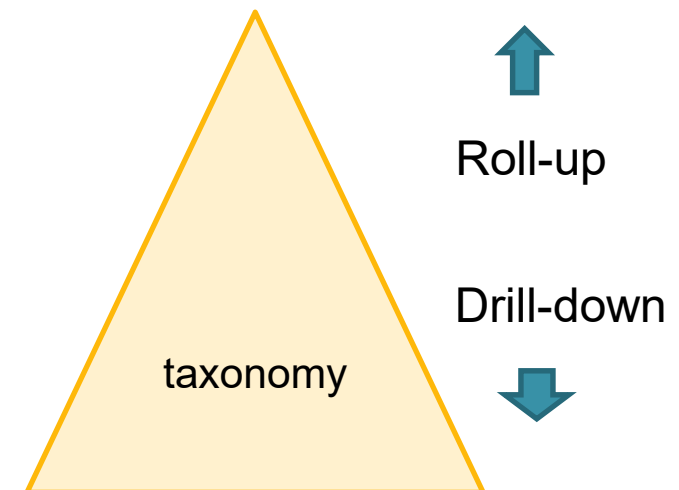
# Products often form Taxonomies



- Note that the taxonomy is often not unique

# Recommending with Taxonomies

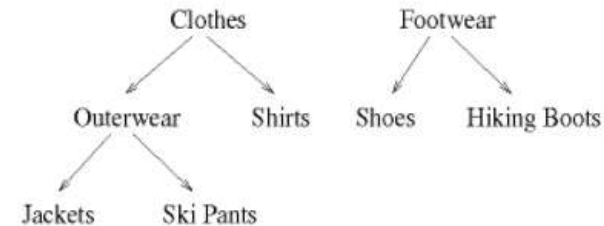
- No need to have all items at the same level
- Level should reflect importance
  - Is brand more important than flavour? Is pack size important?
- Items should occur in approximately the same frequency, otherwise rules are dominated by the common items
  - E.g. rice is bought very frequently, durian cakes very infrequently, hence the rule: *if durian cake then rice* is likely to be found, but is not useful
- Hence .....
  - Roll-up rare items to higher levels so they are more frequent
    - e.g. salami pizza, tuna pizza, veg.pizza etc -> pizza
  - Break-down (drill-down into) very frequent items
    - e.g. pizza -> salami pizza, tuna pizza, veg.pizza etc



# Generalised Association Mining

- Given a large database of transactions, where each transaction consists of a set of items, and a taxonomy (is-a hierarchy) on the items. Find associations between items at any level of the taxonomy.

| Transaction | Items Bought            |
|-------------|-------------------------|
| 100         | Shirt                   |
| 200         | Jacket, Hiking Boots    |
| 300         | Ski Pants, Hiking Boots |
| 400         | Shoes                   |
| 500         | Shoes                   |
| 600         | Jacket                  |



**Frequent Itemsets**

| Itemset                     | Support |
|-----------------------------|---------|
| { Jacket }                  | 2       |
| { Outerwear }               | 3       |
| { Clothes }                 | 4       |
| { Shoes }                   | 2       |
| { Hiking Boots }            | 2       |
| { Footwear }                | 4       |
| { Outerwear, Hiking Boots } | 2       |
| { Clothes, Hiking Boots }   | 2       |
| { Outerwear, Footwear }     | 2       |
| { Clothes, Footwear }       | 2       |

**Rules**

| Rule                                 | Support | Conf. |
|--------------------------------------|---------|-------|
| Outerwear $\Rightarrow$ Hiking Boots | 33%     | 66.6% |
| Outerwear $\Rightarrow$ Footwear     | 33%     | 66.6% |
| Hiking Boots $\Rightarrow$ Outerwear | 33%     | 100%  |
| Hiking Boots $\Rightarrow$ Clothes   | 33%     | 100%  |

- Many algorithms proposed, e.g. see:

<https://www.semanticscholar.org/paper/Mining-generalized-association-rules-Srikant-Agrawal/66a5d6c2b8111c8523e6e9bd43024d54d501ec84>

# Virtual Items

- Expand the scope of association mining from items / products to any categorical variable of interest e.g.
  - Type of promotion
  - Store location (urban, suburban, rural)
  - Season or month, time of day (am, lunch, pm, evening)
  - Payment mode (cash, cheque, credit card)
  - Gender of customer (male, female)
- Can add any relevant information as an ‘item’ into the basket
- E.g. To find associations between purchased items and new customers
  - Enter transaction as: {sweater, jacket, new customer}
  - Possible Association Rule: *If new customer and jacket then sweater*
- Can include numerical concepts into the baskets by first binning them
  - E.g. Age -> young-age, middle-age, senior-age
  - Possible Association Rule: *If outdoor jacket and middle-age then hiking boots*

# Sequence Mining

- Similar to association mining, but where the order in which items are placed into the basket is important
  - E.g. A sequence of events over time, DNA sequences, word sequences
  - Many algorithms based on extended apriori , e.g. SPADE
- Useful for session-based recommender systems
  - Many web-site browsing sessions are made by anonymous users, the only ID is the cookie ID. Hence storing historical data for the user is usually not possible. Making recommendations must use only the browsing behaviour in the current session

*In general:*

| sequences       |
|-----------------|
| a(ab)c(ac)d(cf) |
| (ad)c(bc)(ae)   |
| a(ef)(ab)(df)cb |
| eg(af)cbc       |

Frequent sub-sequence: a(ab)c



*Items in brackets are unordered (e.g. occur at the same time)*

*Clickstream Mining:*

```

session1: home, news, sport, finance
session2: finance, news
session3: fashion, home, sport
session4: news, finance, home
session5: sport, home, finance
session6: fashion, home, news, news, sport
session7: home, finance, news, finance, sport

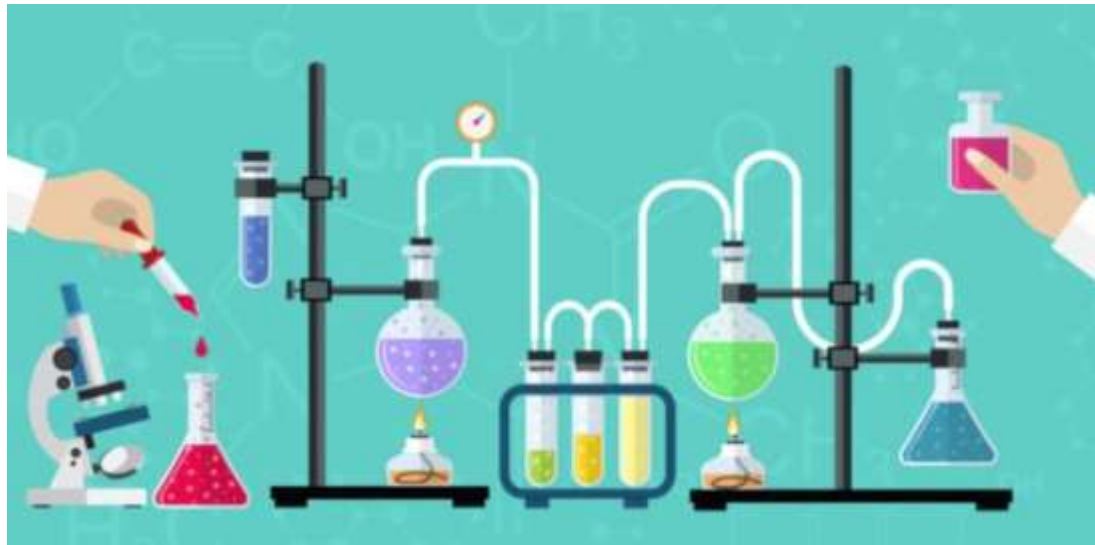
```

Frequent sub-sequence: home-> news -> sport



# Workshop1: Association Mining

- Building association rules
- Rule execution and testing
- Comparison with predictive modelling



# Agenda



| Day       | Topic                                          |
|-----------|------------------------------------------------|
| Day4 (am) | Introduction – basic concepts overview         |
| Day4 (am) | Market Basket Analysis and Recommender Systems |
| Day4 (pm) | Similarity-Based Recommender Systems           |
| Day5 (am) | Model-Based Recommender Systems                |
| Day5 (pm) | Hybrid and Advanced Recommender Systems        |
| Day5 (pm) | Course Assessment Quiz                         |

# Issues with MBA for making Recommendations

- Association rules focus mostly on top-selling items, items of likely interest to most people
- How do we recommend items in the log tail?
  - E-retailers can hold huge inventories without needing big (expensive) shops to display them. The result is a large number of products that only sell a small amount each – the long tail!

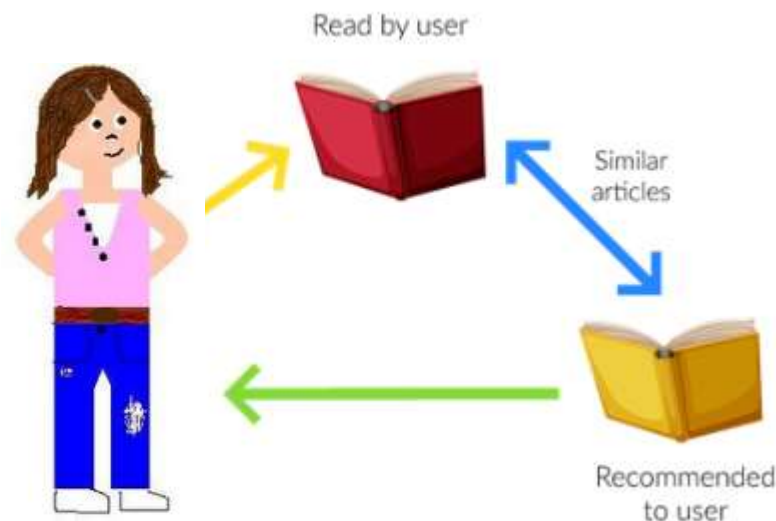


Read the full article at: <http://www.wired.com/2004/10/tail>

# Personalised Recommendation Approaches

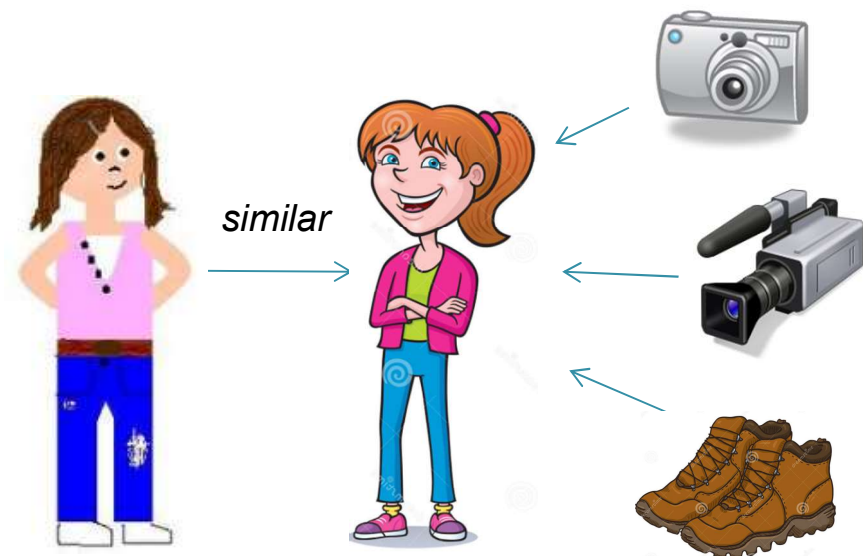
- Content-Based Recommendation

- Recommend based on what you have bought or viewed in the past
- Match users directly to products & content
- Commonly used for document recommendation: webpages, news articles, blogs etc.



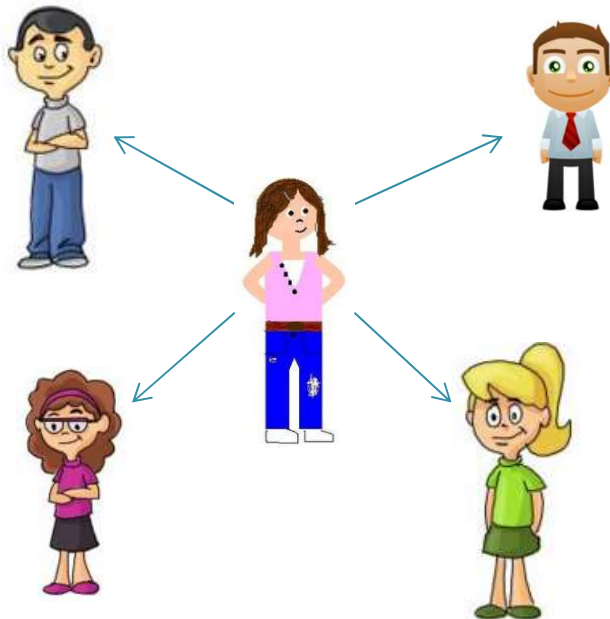
- Collaborative Filtering

- Match users to people with similar tastes
- recommend what they like
- Commonly used in e-retail
- Avoids the issue of users only being recommended more of what they already like (allows serendipity)



# Personalised Recommender System Approaches

- Community-Based/Social
  - Who are your friends and what do they like?
  - Users trust their friends opinions



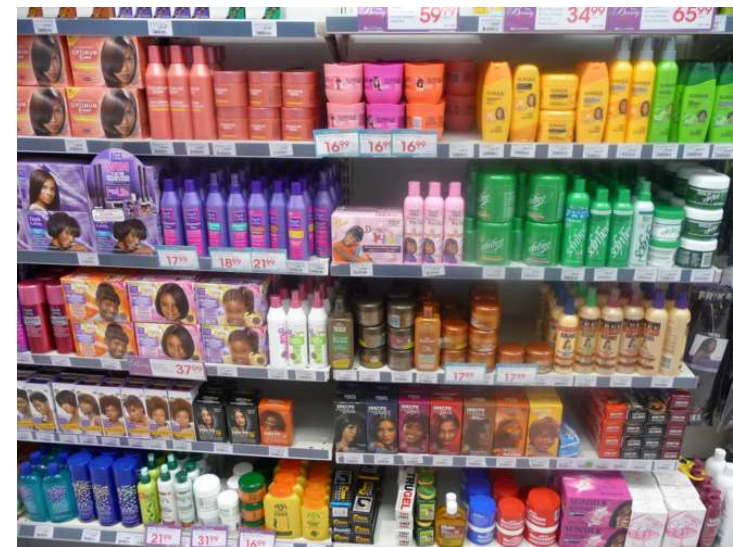
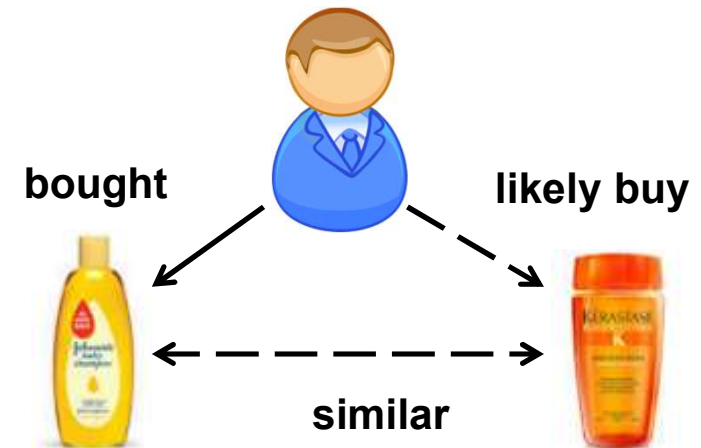
- Knowledge-Based
  - Often more appropriate for recommending products where many **user specific requirements and constraints (often conflicting)** must be taken into account
    - E.g. cars, houses, computers, financial services
  - Example Methods:
    - Constraint-based Reasoning /Search
    - Multi-Criteria Decision Making (MCDM)

# Content-Based Recommenders

Match the attributes of a user with those of an item.

Main components are:

- **Content Analyser** ~ extracts structured product descriptions from a diversity of products
  - E.g. for movie: director, length, year, genre etc.
  - Data may already be available – e.g. structured data provided by the manufacturer
  - **OR** extract using text mining -parse product descriptions (e.g. product webpages) and extract features using keyword extraction
- **Profile Learner** ~ collects data about the user preferences and generalises into a user profile
- **Filtering** ~ matches the user profile against the product descriptions. Returns the products with the closest match





# Content-Based Filtering Example

|        | Length | studio | Director  | Lead Actor1 | Genre  | Date |
|--------|--------|--------|-----------|-------------|--------|------|
| movie1 | 90     | Disney | Spielberg | Tom Hanks   | Sci-fi | 2014 |
| movie2 | 110    | MGM    | Petersen  | Brad Pitt   | action | 2012 |
| movie3 | 120    | Disney | Nolan     | C. Bale     | action | 2008 |
| .....  |        |        |           |             |        |      |
| movieN |        |        |           |             |        |      |



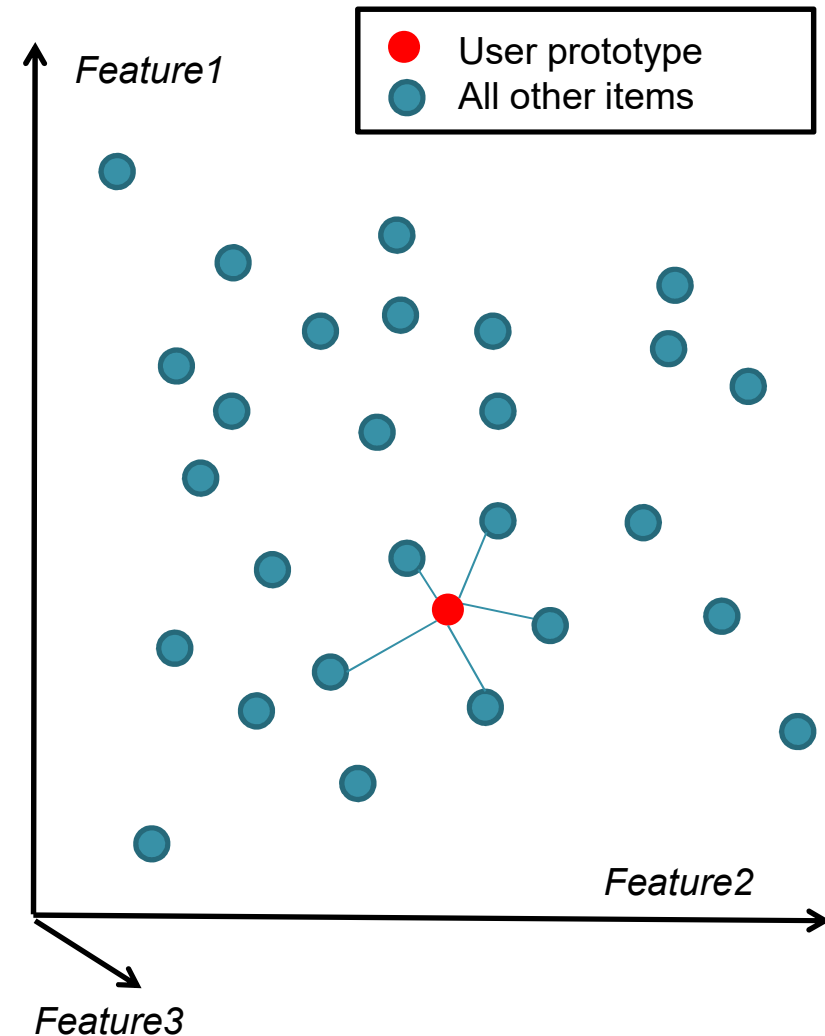
Which Movie(s) might our user like?

# Similarity-Based Reasoning\*

- Each item is represented by a set of features (a *feature vector*)
- The user is represented by the same feature set, often obtained by:
  - The user selects preferences for the various features using pull-down menus
  - Computing an “average” vector from items already bought/liked by the user. This will represent a “typical” item for the user (often called a “prototype vector”)
- Then use a distance (or similarity) measure to find the nearest items to the user
- The nearest items are then recommended – ranked by their distance

*\*Also known as:*

- *Vector-Space approach*
- *Nearest Neighbour approach*
- *Memory-Based Reasoning*

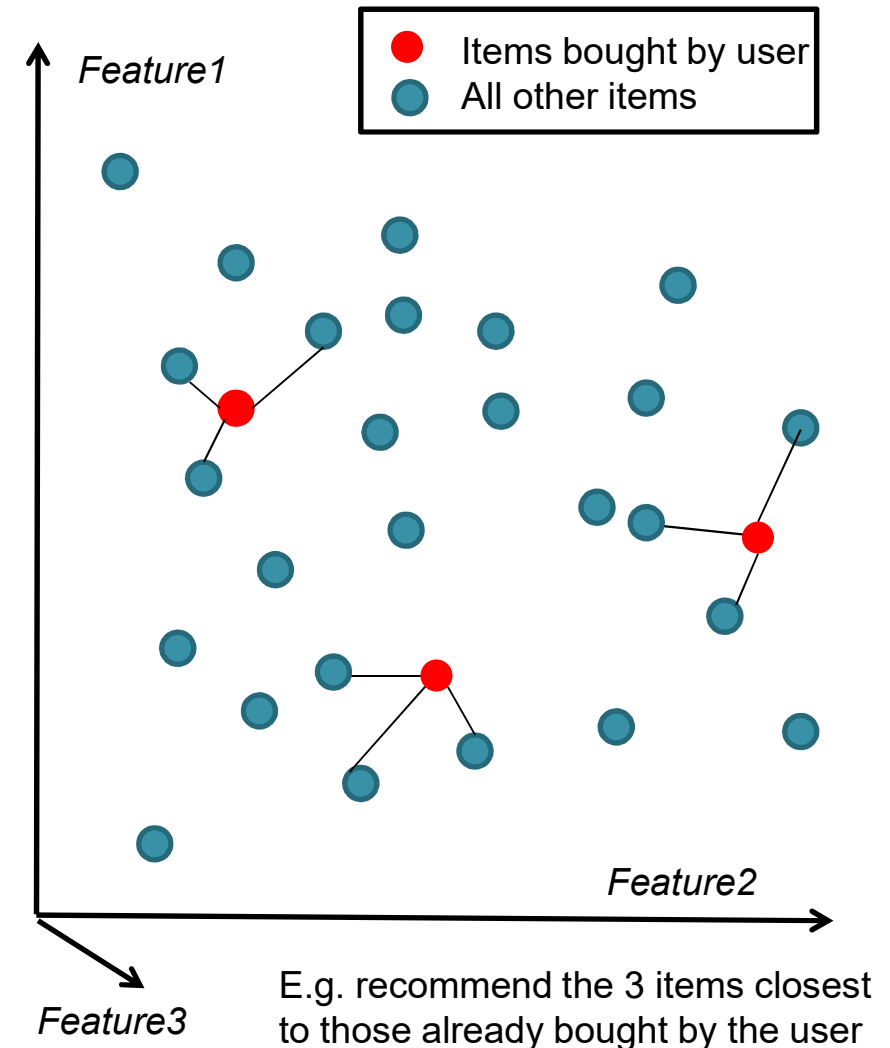


# Similarity-Based Reasoning

- If the retailer is selling an diverse product range there is probably no single “typical” item for a user
- Instead we can recommend items that are close to the items that the user has already bought (or viewed)

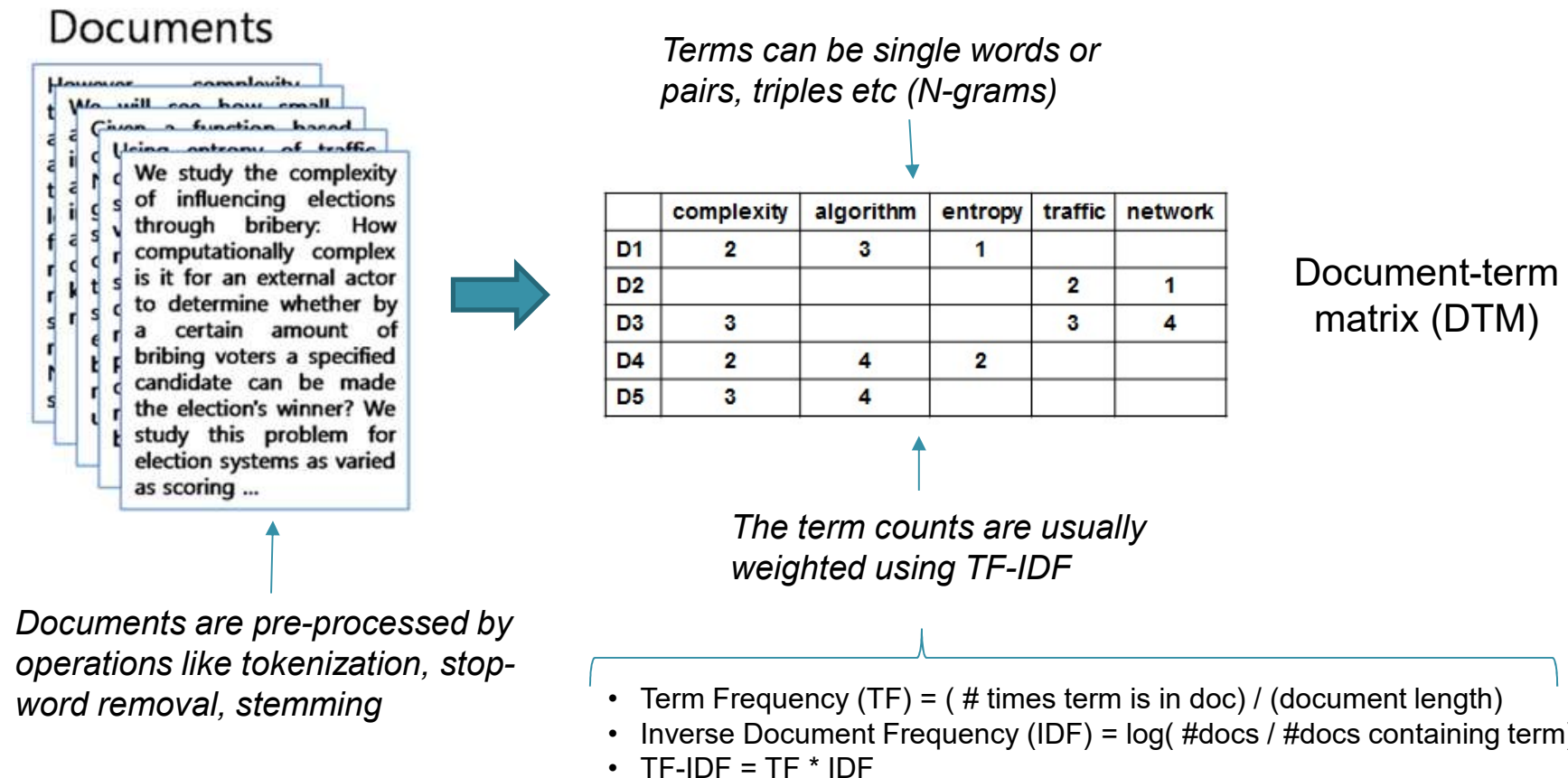
## Transparency

Can explain a recommendation by showing the factors that trigger the recommendation(s)



# Text-Based Features

- Instead of explicitly defining product features we can extract text-based features from product documentation, product reviews, product webpages etc.
- E.g. The Bag-of-Words approach (BOW) represents a document by features that count the number of times each word (or term) occurs in the document

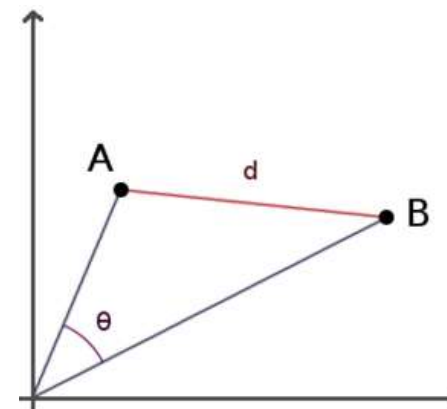


# Similarity & Distance Measures

- Content-Based Recommender Systems typically use the cosine distance to measure the similarity between user and items
- In information retrieval systems\* the similarity between two documents (A and B) is given by:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

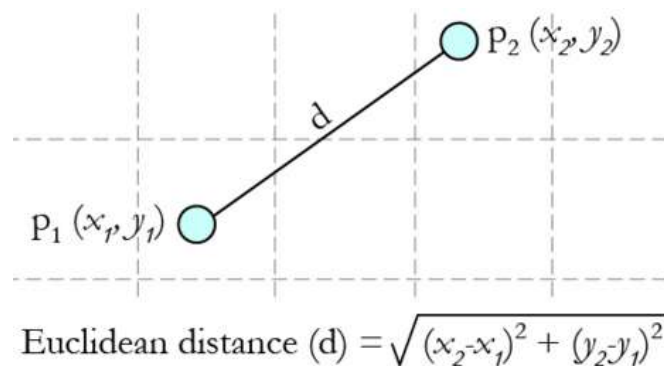
*\*Content-based recommender systems have their roots in Information Retrieval (IR). The goal of IR is to identify relevant documents in response to a user query*



When the documents are similar the angle between their vectors is small and the cosine similarity approaches +1

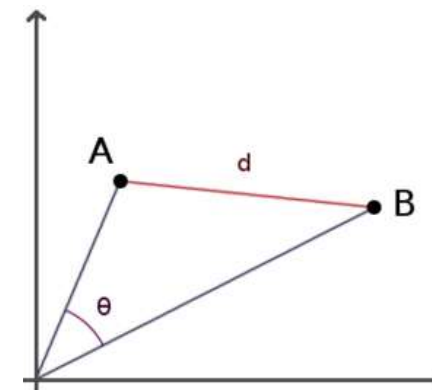
# Why Cosine Similarity?

- For general classification problems, the distance metric used in k-NN systems is commonly the Euclidean distance



Can convert to “similarity” using (for example):  
similarity =  $1/(1 + \text{distance})$

If the vectors are documents then Cosine Similarity is often better because it ignores the magnitude of the vectors (only considers angle)



E.g. if one doc has word “disease” 10 times and another has it 100 times then using cosine they will still be similar but they will be different using Euclidean distance

*E.g. Two docs with similar word profiles*

|       | w1 | w2  | w3  | w4       | w5       | w6     |
|-------|----|-----|-----|----------|----------|--------|
| Item1 | 0  | 10  | 20  | 100      | 200      | 400    |
| Item2 | 0  | 100 | 200 | 1000     | 2000     | 4000   |
|       |    |     |     | Eucl Dst | Eucl Sim | Cosine |
|       |    |     |     | 4129.2   | 0.0002   | 1.0000 |



# Content-Based Example - Pandora

- Pandora is an online streaming music company,
- A team of musicologists annotates songs based on genre, rhythm, and progression. This data is transformed into a vector for comparing song similarity.
- Most songs are based on a feature vector of approximately 450 features, which were derived in a very long and arduous process
- Once this was done binary classification methods using more traditional machine learning techniques can be used to
  - Output a probability for a certain user to like a specific song based on a training set of their song listening history.
  - Recommend the songs with the greatest probability of being liked.
- This approach helps to promote the presentation of long tail music from unknown artists that might be a good fit for a particular user.

# BestBuy Revisited



Goal: Recommend the best product (sku category) for a user's search query

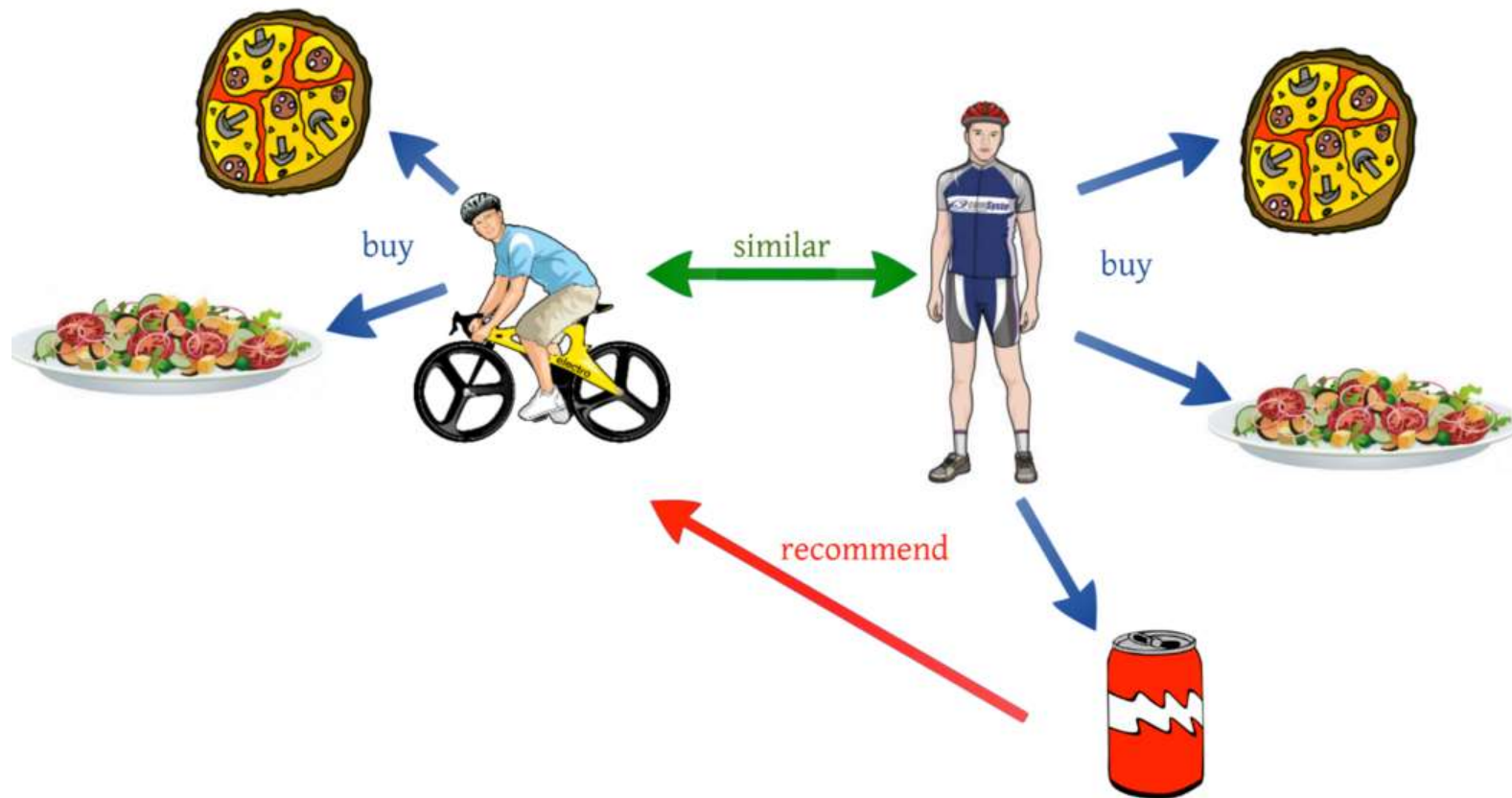
| user          | sku     | category     | query        | click_time | query_time |
|---------------|---------|--------------|--------------|------------|------------|
| 585c9ba287c96 | 2032076 | abcat0701002 | gears of war | 22:56.1    | 21:42.9    |
| d3626c4b56ff5 | 9854804 | abcat0701002 | Gears of war | 35:42.2    | 35:33.2    |

- Simplest Solution ~ frequency counting
  - Count the *query*->*product* instances
  - Build a dictionary:  
key=query, values = (sku, clickcnt) pairs
  - Recommend the sku with most clicks
  - Recommend most popular sku's in each product category  
if no past query->sku mapping exists
- Better Approach ~ deriving more features
  - Use TF-IDF to extract features from the query and from the product descriptions
  - Use similarity approach to map query (the user profile) to the closest product

'forzasteeringwheel': {'2078113':1},  
'finalfantasy13': {'9461183':3, '3519923':2},  
'guitarps3': {'2633103':1}

# Collaborative Filtering (CF)

Match the user to similar users and recommend what they like



# Measuring User Similarity

- How do we find similar users? What user data can we use?
- Most common data used is the ***user-ratings matrix***, a matrix of the user likes or interests in the various products/items on offer



|       |   |   |   |   |
|-------|---|---|---|---|
| John  | 5 | 1 | 3 | 5 |
| Tom   | ? | ? | ? | 2 |
| Alice | 4 | ? | 3 | ? |

Entries are numbers,  
e.g. 1->5, where 1  
means do not like, 5  
means like a lot

Ratings can be explicit (directly given) or  
implicit (inferred from page-views, purchases etc.)

# Finding Similar Users

- How to compute distances/similarities between entries in the ratings matrix?
- Sparsity (many missing values) is a problem
- E.g. what is the similarity between John and Tom ?

|                                                                                           |  |  |  |  |
|-------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| John     | 5                                                                                 | 1                                                                                 | 3                                                                                   | 5                                                                                   |
| Tom     | ?                                                                                 | ?                                                                                 | ?                                                                                   | 2                                                                                   |
| Alice  | 4                                                                                 | ?                                                                                 | 3                                                                                   | ?                                                                                   |

Common Metrics:

- Euclidean Distance
- Cosine Similarity
- Pearson Correlation

# Finding Similar Users

- Users may use different ratings to quantify the same level of appreciation for an item. This can create bias. E.g. one user may rate a movie as 5 and the other 4, yet they both like it equally well
- Pearson Correlation Coefficient** is similar to cosine similarity but eliminates this bias by subtracting each users mean rating from their individual ratings. Value ranges from -1 to +1

$$\text{Sim}(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i \in C} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in C} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in C} (r_{v,i} - \bar{r}_v)^2}}$$

$r_{u,i}$  = rating of user  $u$  on item  $i$   
 $r_{v,i}$  = rating of user  $v$  on item  $i$   
 $\bar{r}_u, \bar{r}_v$  = mean ratings of user's  $u, v$   
 $C$  is the set of all co-rated items

| User   | movie1 | movie2 | movie3 | movie4 | movie5 | movie6 | Mean? |
|--------|--------|--------|--------|--------|--------|--------|-------|
| target | 1      | 2      |        |        | 3      | 2      | 2     |
| 1      | 2      | 5      | 4      |        | 5      | 1      | 3.4   |
| 2      | 2      | 3      | 5      | 4      |        | 2      | 3.2   |
| 3      |        |        | 5      | 3      |        | 4      | 4     |

```
> users
 m1 m2 m3 m4 m5 m6
target 1 2 NA NA 3 2
u1 2 5 4 NA 5 1
u2 2 3 5 4 NA 2
u3 NA NA 5 3 NA 4
```

```
> cor(t(users), use="pairwise.complete.obs")
 target u1 u2 u3
target 1.000000 0.5940885 0.5000000 NA
u1 0.5940885 1.0000000 0.6454972 1.0000000
u2 0.5000000 0.6454972 1.0000000 0.3273268
u3 NA 1.0000000 0.3273268 1.0000000
```



# Making a Recommendation

- Once we have the similarities between the target user and the other users then we compute a recommendation based on the preferences of other users. One method is to sum the preferences of these other users weighted by their similarity to the target (weighted average)

$$\text{Predicted rating } (u,i) = \bar{r}_u + \frac{\sum_{v \in V} \text{sim}(u,v) * (r_{v,i} - \bar{r}_v)}{\sum_{v \in V} |\text{sim}(u,v)|} \quad \left\{ \begin{array}{l} V \text{ is the set of} \\ \text{other users} \end{array} \right.$$

| User             | similarity | m1   | m2   | m3                                           | m4                          | m5  | m6   | Mean |
|------------------|------------|------|------|----------------------------------------------|-----------------------------|-----|------|------|
| target           | 1          | -1   | 0    | ?                                            | ?                           | 1   | 0    | 2    |
| user1            | 0.59       | -1.4 | 1.6  | 0.6                                          |                             | 1.6 | -2.4 | 3.4  |
| user2            | 0.5        | -1.2 | -0.2 | 1.8                                          | 0.8                         |     | -1.2 | 3.2  |
| user3            | NA         |      |      | 1                                            | -1                          |     | 0    | 4    |
| weighted average |            |      |      | $\frac{0.59*0.6 + 0.5*1.8}{0.59+0.5} = 1.15$ | $\frac{0.5*0.8}{0.5} = 0.8$ |     |      |      |
| predicted rating |            |      |      | $2 + 1.15 = 3.15$                            | $2 + 0.8 = 2.8$             |     |      |      |

# Other Normalisation Methods

## Z-score normalization

- Divide the user *mean-centered* rating by the user's rating standard deviation. Hence the normalised rating for user  $u$  on movie  $i$  is:

$$\frac{r_{ui} - \bar{r}_u}{\sigma_u}$$

- The predicted rating for an unrated movie  $i$  for user  $u$  is then:

$$\hat{r}_{ui} = \bar{r}_u + \sigma_u \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} (r_{vi} - \bar{r}_v) / \sigma_v}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|}$$

- Z-score normalisation is useful if the rating scale has a ***wide range of discrete values or is continuous***.
- But, because the ratings are divided and multiplied by possibly very different standard deviation values, Z-score can be more sensitive than mean-centering and can often predict ratings that are outside the rating scale.

# A Simple Example\*

The raw ratings....

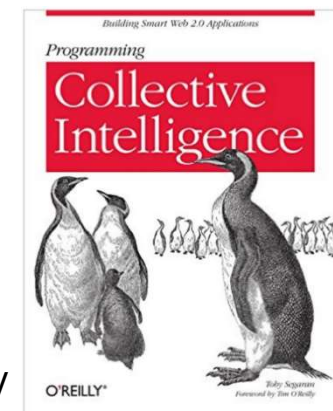
| User     | Lady In Water | Snakes On A Plane | Just My Luck | Superman | You Me And Dupree | The Night Listener |
|----------|---------------|-------------------|--------------|----------|-------------------|--------------------|
| Rose     | 2.5           | 3.5               | 3            | 3.5      | 2.5               | 3                  |
| Seymour  | 3             | 3.5               | 1.5          | 5        | 3.5               | 3                  |
| Philips  | 2.5           | 3                 |              | 3.5      |                   | 4                  |
| Puig     |               | 3.5               | 3            | 4        | 2.5               | 4.5                |
| LaSalle  | 3             | 4                 | 2            | 3        | 2                 | 3                  |
| Matthews | 3             | 4                 |              | 5        | 3.5               | 3                  |
| Toby     |               | 4.5               |              | 4        | 1                 |                    |

The goal is to make movie recommendations for Toby.

Do this by predicting his likely rating for his unseen movies:

***Lady in Water, Just My Luck, The Night Listener***

\*Example from “Programming Competitive Intelligence”, O’Reilly



# (1) Calculate Toby's Similarity to Others

| User     | Lady In Water | Snakes On A Plane | Just My Luck | Superman | You Me And Dupree | The Night Listener |
|----------|---------------|-------------------|--------------|----------|-------------------|--------------------|
| Rose     | 2.5           | 3.5               | 3            | 3.5      | 2.5               | 3                  |
| Seymour  | 3             | 3.5               | 1.5          | 5        | 3.5               | 3                  |
| Philips  | 2.5           | 3                 |              | 3.5      |                   | 4                  |
| Puig     |               | 3.5               | 3            | 4        | 2.5               | 4.5                |
| LaSalle  | 3             | 4                 | 2            | 3        | 2                 | 3                  |
| Matthews | 3             | 4                 |              | 5        | 3.5               | 3                  |
| Toby     |               | 4.5               |              | 4        | 1                 |                    |

The Pearson implementation in the book and also in R (the `cor()` fn) works with sparse data by only considering the “pairwise complete” items.

Hence:

- $\text{Mean}(\text{Rose}) = (3.5+3.5+2.5)/3 = 3.16$ ,  $\text{Mean}(\text{Toby}) = (4.5+4+1)/3 = 3.16$
- $\text{Sim}(\text{Toby}, \text{Rose}) = \frac{(3.5-3.16)*(4.5-3.16) + (3.5-3.16)*(4-3.16) + (2.5-3.16)*(1-3.16)}{\sqrt{((3.5-3.16)^2+(3.5-3.16)^2+(2.5-3.16)^2) * ((4.5-3.16)^2+(4-3.16)^2+(1-3.16)^2)}} = 0.99$

## (2) Compute Weighted Avg of Neighbour Ratings

| Critic         | Similarity | Night | S.xNight | Lady | S.xLady | Luck | S.xLuck |
|----------------|------------|-------|----------|------|---------|------|---------|
| Rose           | 0.99       | 3.0   | 2.97     | 2.5  | 2.48    | 3.0  | 2.97    |
| Seymour        | 0.38       | 3.0   | 1.14     | 3.0  | 1.14    | 1.5  | 0.57    |
| Puig           | 0.89       | 4.5   | 4.02     |      |         | 3.0  | 2.68    |
| LaSalle        | 0.92       | 3.0   | 2.77     | 3.0  | 2.77    | 2.0  | 1.85    |
| Matthews       | 0.66       | 3.0   | 1.99     | 3.0  | 1.99    |      |         |
| Total          |            |       | 12.89    |      | 8.38    |      | 8.07    |
| Sim. Sum       |            |       | 3.84     |      | 2.95    |      | 3.18    |
| Total/Sim. Sum |            |       | 3.35     |      | 2.83    |      | 2.53    |

The similarity of the other users to Toby (this was calculated using the Pearson Coefficient)

All neighbours with similarity > 0 are included in the weighted average (avoid having to select a value for K)

The weighted average of the other users ratings for each unseen movie.

Note that, for simplicity, only the raw ratings are used here

# User-Based CF: Scalability Issues

- User-Based Collaborative Filtering does not scale well
  - User likes/interests may change often hence similarities between users are best computed at the time of the recommendation – this becomes computationally prohibitive for large numbers of users
- Computational load
  - Worse case  $\sim O(M \cdot N)$  where  $M = \text{\#users}$ ,  $N = \text{\#items}$
  - In practice  $\sim O(M + N)$  since most users have rated/purchased few items
- Possible solutions
  - Dimensionality Reduction
  - Item-Item Collaborative Filtering

# Another way to Scale Collaborative Filtering

## Industry Report



# Amazon.com Recommendations

## *Item-to-Item Collaborative Filtering*

Greg Linden, Brent Smith, and Jeremy York • Amazon.com

<http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf>



# Item-Based Collaborative Filtering

- Transpose the user-rating matrix, compute distances between items not users
- Faster than User-Based – there are usually far fewer items than users AND the similarities between items change much less frequently than between users hence all item-item distances can be pre-computed
- A well known application of this method is Amazon's recommendation engine

| item   | user1 | user2 | user3 | user4 | .... | userN |
|--------|-------|-------|-------|-------|------|-------|
| movie1 | 2     | 5     | 4     |       |      |       |
| movie2 |       | 3     |       |       | 1    |       |
| .....  |       |       |       |       |      |       |

*Transposition of the user rating matrix*

# Example Item-Item Similarity Matrix

|       | item1 | item2 | item3 | item4 | item5 | item6 | ... |
|-------|-------|-------|-------|-------|-------|-------|-----|
| item1 | 1     | 0.90  | 0.61  | 0.13  | 0.68  | 0.75  | ... |
| item2 |       | 1     | 0.58  | 0.78  | 0.12  | 0.29  | ... |
| item3 |       |       | 1     | 0.55  | 0.32  | 0.69  | ... |
| item4 |       |       |       | 1     | 0.11  | 0.98  | ... |
| item5 |       |       |       |       | 1     | 0.75  | ... |
| item6 |       |       |       |       |       | 1     | ... |
| ....  |       |       |       |       |       |       | ... |

The matrix is symmetrical. Similarity is measured from 0 to 1 or -1 to 1, diagonals hence take the value 1

Typically precomputed, the frequency of updating depends on type of item (e.g. fast or slow seller, speed of new releases) – weekly or monthly updates may suffice.

# Item-Based Collaborative Filtering

- Compute the distance between items  $i$  and  $j$  using the adjusted cosine similarity\*:

$$\text{sim}(\mathbf{i}, \mathbf{j}) = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_u)^2}}$$

{

$r_{u,i}$  = rating of user  $u$  on item  $i$   
 $r_{u,j}$  = rating of user  $u$  on item  $j$

$\bar{r}_u$  = mean rating of user  $u$

$U$  = set of all users

- Predict user  $u$ 's rating for an item  $i$  using:

$$\text{Predicted rating } (u,i) = \frac{\sum_{j \in J} \text{sim}(i,j) * r_{u,j}}{\sum_{j \in J} |\text{sim}(i,j)|}$$

{

$J$  is the set of  $K$  similar items to  $i$  (among items that  $u$  has rated)

\* Other similarity measures can be used (e.g. Euclidean Similarity)

Example <https://towardsdatascience.com/comprehensive-guide-on-item-based-recommendation-systems-d67e40e2b75d>  
 R code: <https://towardsdatascience.com/comprehensive-guide-on-item-based-recommendation-systems-d67e40e2b75d>

# Item-Based Recommendations for Toby

↓ Movies seen by Toby      → Movies not seen by Toby

| Movie      | Rating | Night | R.xNight | Lady  | R.xLady | Luck  | R.xLuck |
|------------|--------|-------|----------|-------|---------|-------|---------|
| Snakes     | 4.5    | 0.182 | 0.818    | 0.222 | 0.999   | 0.105 | 0.474   |
| Superman   | 4.0    | 0.103 | 0.412    | 0.091 | 0.363   | 0.065 | 0.258   |
| Dupree     | 1.0    | 0.148 | 0.148    | 0.4   | 0.4     | 0.182 | 0.182   |
| Total      |        | 0.433 | 1.378    | 0.713 | 1.764   | 0.352 | 0.914   |
| Normalized |        |       | 3.183    |       | 2.473   |       | 2.598   |

Toby's ratings for the movies he has seen

The similarities between **Night Listener** and the movies seen by Toby – these are derived from the pre-computed item-item similarity matrix (using Euclidean similarity)

$$(= 1.378/0.433)$$

Toby's estimated rating for **Night Listener** is obtained by computing his average rating for all of his seen movies – weighted by their distance to **Night Listener**

# Which is Best: User-Based or Item-Based?

- In general...
- If  $\#users \gg \#items$  (e.g. Amazon) then Item-Item may be better
- If  $\#users \ll \#items$  (e.g. recommending web content) then User-User may be better
- Workshop2
  - Compare User-based versus Item-based CF
  - Compare different similarity measurements
  - Compare normalised versus non-normalised data
  - Explore handling large datasets using data reduction

# Evaluating Recommender Systems

- How can we test a Recommender System before going live?
- How accurate does a Recommendation System Need to be?
- How can we estimate / measure the success of a Recommender System?
- What makes one Recommender System better than another?



# Evaluating the accuracy of rating predictions

- For recommender systems that predict ratings
- Hold-out testing:
  - Hide, in turn, each rating\* made by each test user and then predict the rating for the hidden item and compute:  
 $\text{error} = \text{actual rating} - \text{predicted rating}$

*\*or we can hide (hold-out) more than one rating for each test*

| User      | movie1 | movie2 | movie3 | movie4 | movie5 | movie6 | etc.... |
|-----------|--------|--------|--------|--------|--------|--------|---------|
| Test user | 2      | 5      | 4      |        | 3      | 1      |         |

For each movie rated by the test user:

- Set the movie rating to blank (NA) – but keep a copy
- Make a prediction for that movie using the training data
- Compare the prediction with actual rating:  
 $\text{error} = \text{abs}(\text{predicted rating} - \text{actual rating})$
- Increment a count of the number of tests
- Restore the blank movie rating

Do this for all test users

At the end compute the MAE (Mean Absolute Error)

$$\text{MAE} = \frac{\sum \text{abs(errors)}}{\# \text{ tests}}$$

What does a MAE of (say) 1.18 mean in practice? Is the recommendation good or bad?



# Evaluating the accuracy of predicting “like”

- Assume the user likes an item if they rate it highly , e.g. rating  $\geq 4$  (on scale of 1 to 5)
- This enables us to compute a confusion matrix:

| Actual     | Recommendation |           |
|------------|----------------|-----------|
|            | Won't Like     | Will Like |
| Rated Poor | TN             | FP        |
| Rated High | FN             | TP        |

*TN = true negative, FP = false positive*  
*FN = false negative, TP = true positive*

Precision =  $\frac{TP}{TP+FP}$       % of predicted likes that were good

Recall =  $\frac{TP}{TP+FN}$       % of likes that we predicted (aka sensitivity)

- Procedure

Same as before - for each test user and each test item:

- increment TP when predicted rating is  $\geq T$  AND actual rating is  $\geq T^*$
- increment FP when predicted rating is  $\geq T$  AND actual rating is  $< T$
- increment TN when predicted rating is  $< T$  AND actual rating is  $< T$
- increment FN when predicted rating is  $< T$  AND actual rating is  $\geq T$

BUT this only evaluates the accuracy when predicting “like”. It is not the **overall effectiveness** of the set of recommendations made

*\*T is the “like” threshold, e.g. 4*

# Evaluating Recommender Systems

- More than one item is usually recommended, what is the combined accuracy? Overall effectiveness?
- What if there is no rating prediction (e.g. as in content-based recommendation) so cannot compute *error*?
- We really need to know how **good** the **recommendations** are (not how accurate the rating or “like” predictions are)

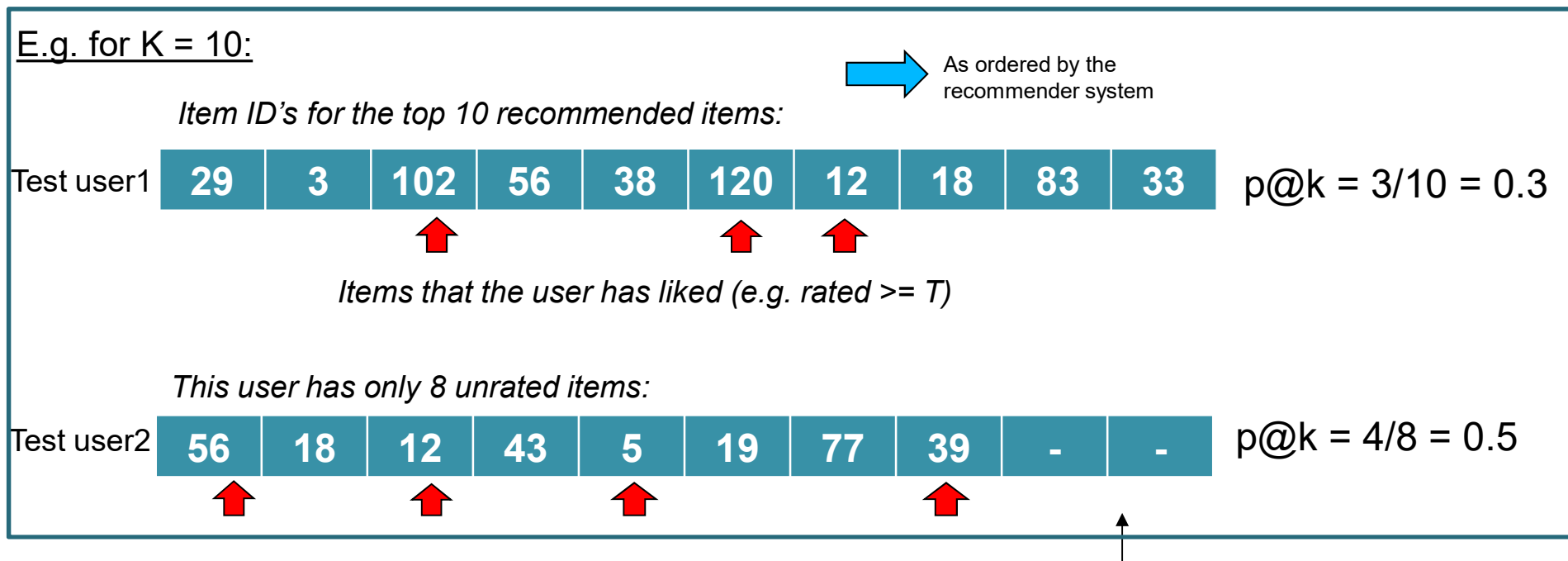


How many recommendations are made? What % are good?

E.g. if you get shown K items how many did you like?

# Precision at K ( $P@K$ )

- $P@K$  is the proportion of recommended items in the top-K set that are relevant
- An item is relevant if the user is known to like it (e.g. actual rating  $\geq 4$ )



\*When computing  $p@k$  for a user from a test set, we remove from the recommended item list all items not in the test set for that user (these are the items that we do not know the actual rating for). For users with few ratings this may cause issues.

[https://medium.com/@m\\_n\\_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54](https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54)

# Mean Average Precision at K (MAP@K)

## P@K

How many relevant items are present in the top-k recommendations of your system

For example, to calculate  $P@3$ : take the top 3 recommendations for a given user and check how many of them are good ones. That number divided by 3 gives you the  $P@3$

## AP@K

The mean of  $P@i$  for  $i=1, \dots, K$ .

For example, to calculate  $AP@3$ : sum  $P@1$ ,  $P@2$  and  $P@3$  and divide that value by 3

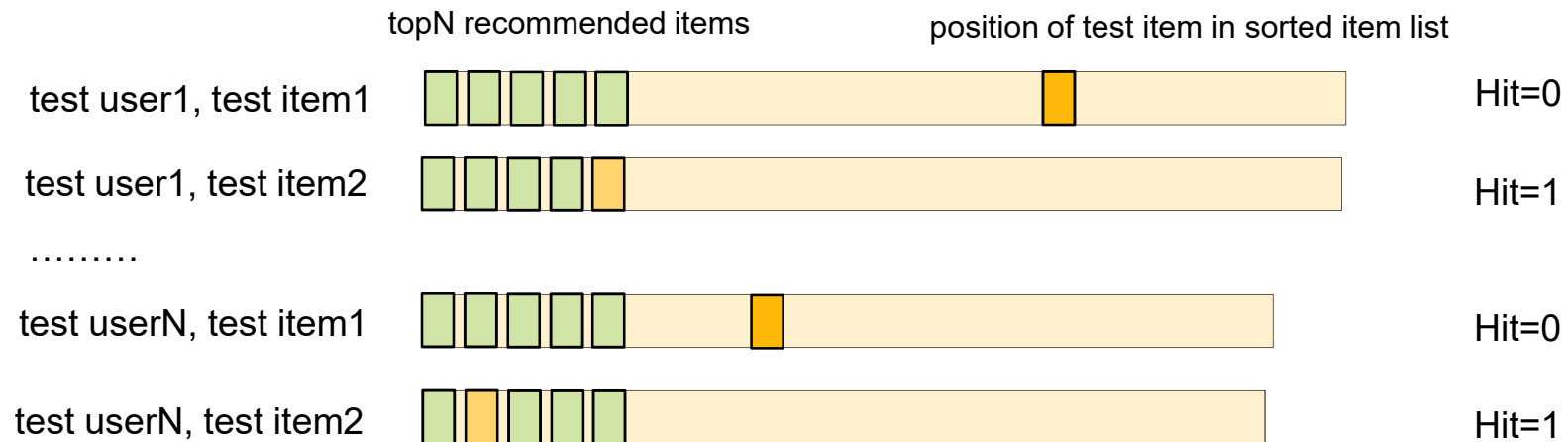
$AP@K$  is typically calculated for one user.

## MAP@K

The mean of the  $AP@K$  for all the users.

# Estimating Lift over Random

- Make recommendations for each test user by selecting the top-K items in the sorted list.
- Count how many times a test item appeared in the top-K (call this the #hits).
- Compare with how many would be expected if random recommendations are made
- E.g. if top-K = 5:



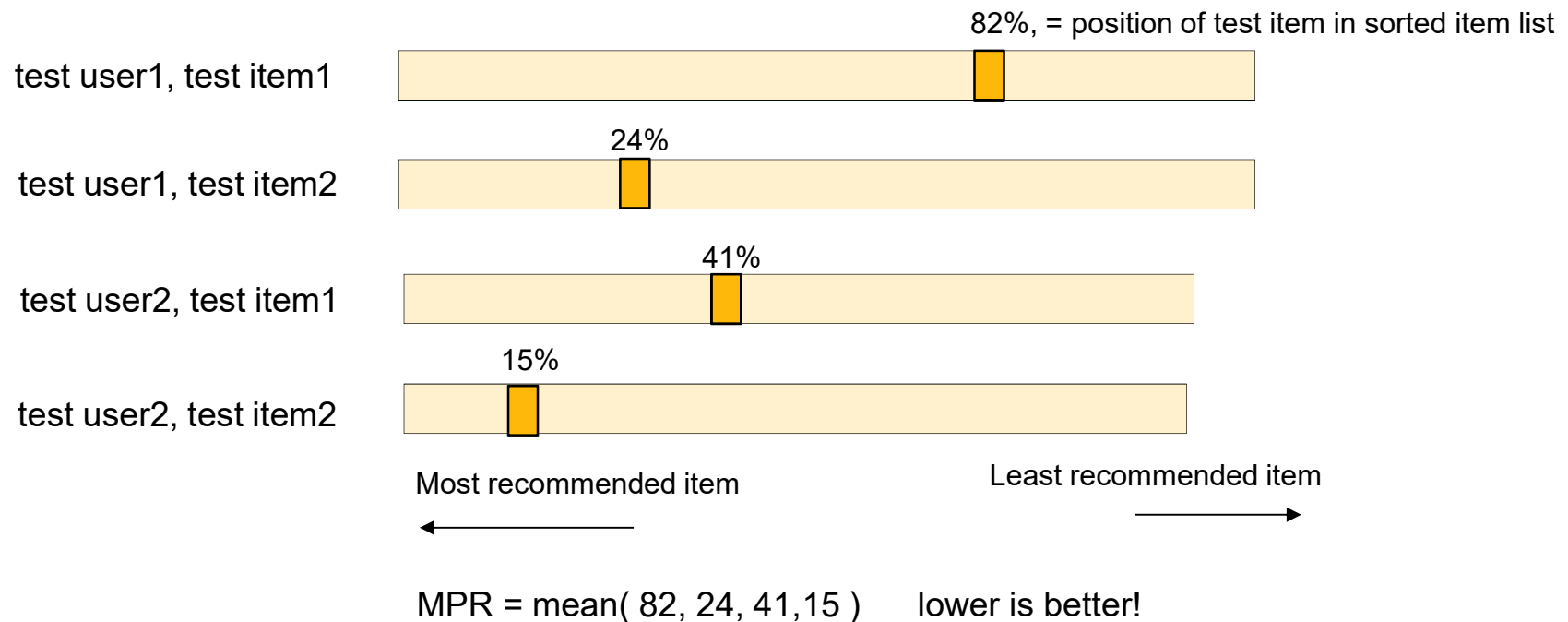
$$\text{Hit\%} = (\text{\#hits}) / (\text{\#recommendations made})$$

$$\text{Lift} = (\text{hit\% using model}) / (\text{hit\% using random})$$

*\*\*For large item sets you may need large test sets (or large K) to get significant results!*

# Mean Percentage Ranking

- As before, but only consider items that are *liked*
- For each test item, predict its position (as a %) in the ranked list of all items (*ranked by their decreasing likeability by the user, e.g. their predicted rating*)
- Since we know the item is liked we hope its predicted rank is low (to the left), *e.g. Rank=1 => top of list => top recommendation*
- $MPR < 50\%$  indicates test results are better than random guessing



# Workshop2 – Collaborative Filtering

