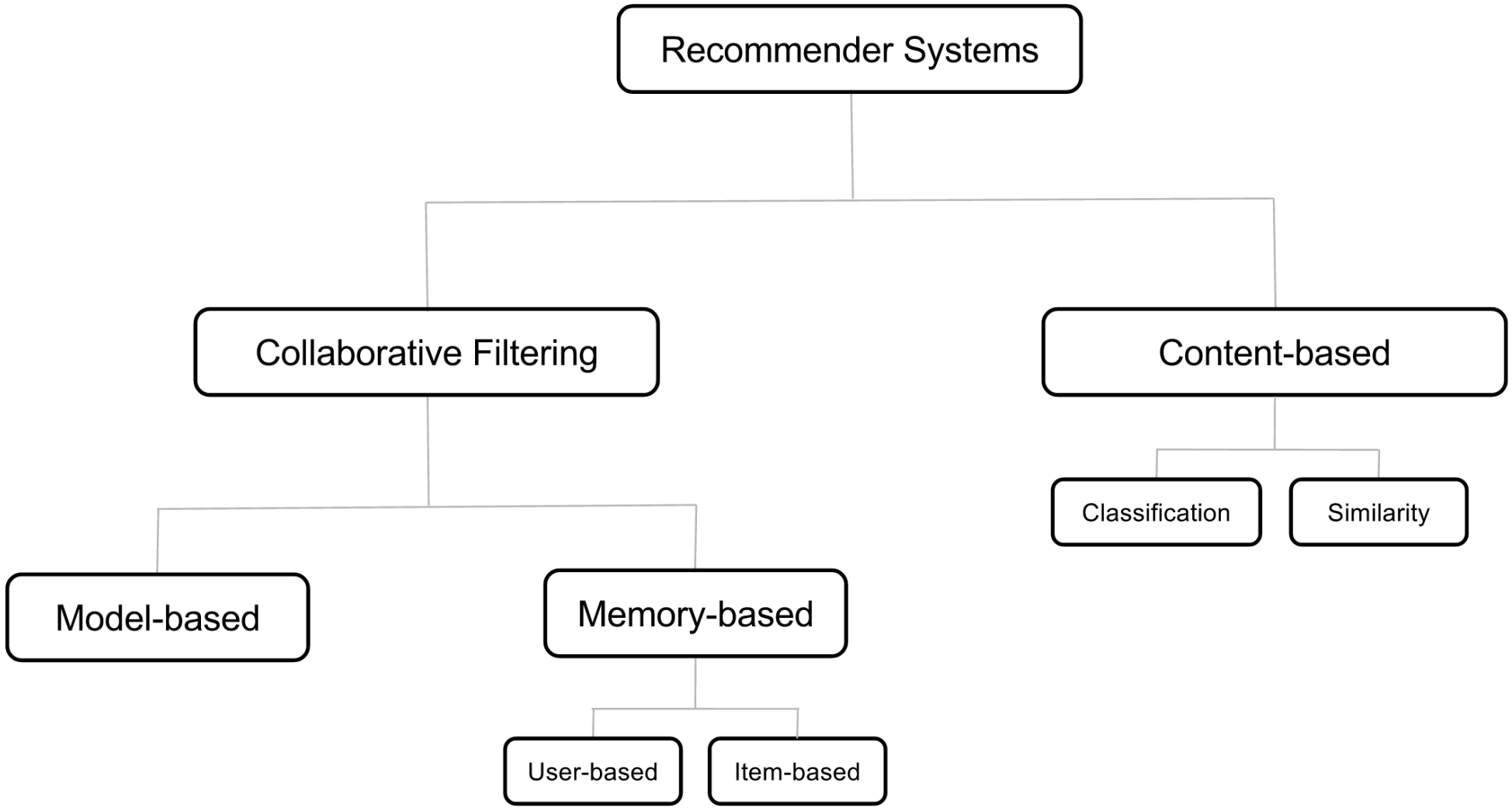
Professional Matchmaking in R

Duncan Stoddard

duncan@dsanalytics.co.uk

www.dsanalytics.co.uk

DS Analytics



DS Analytics

Fill in the gaps…

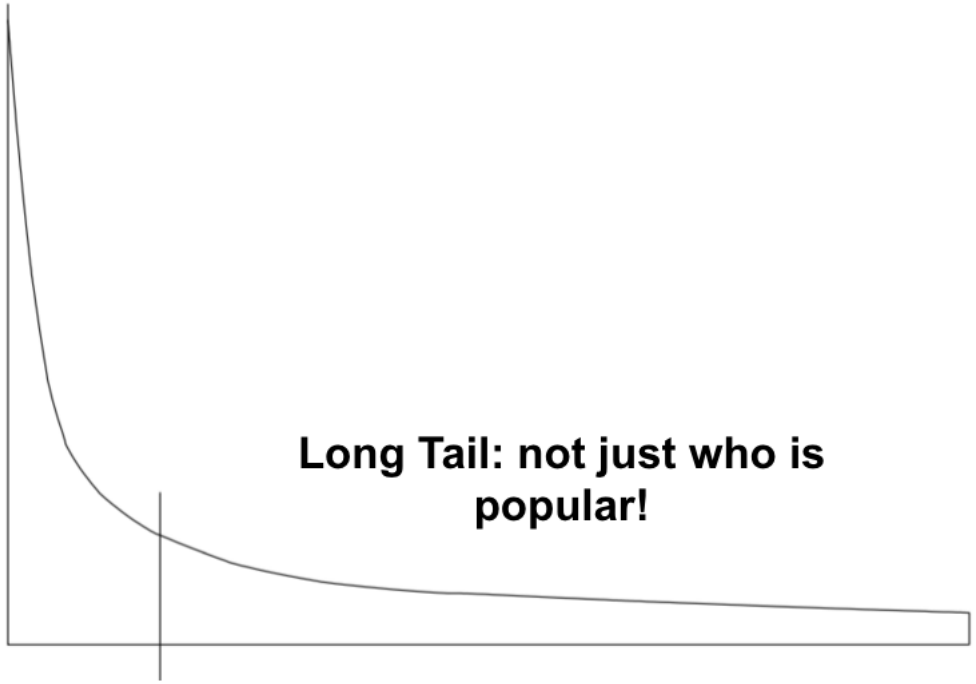
advisors

projects / professionals

USERS	ITEMS a	b	c	d	e
1	1	0			
2		1	1		1
3	0				0
4				0	
5	1	1	1		1

DS Analytics

User-personalised recommendations



DS Analytics

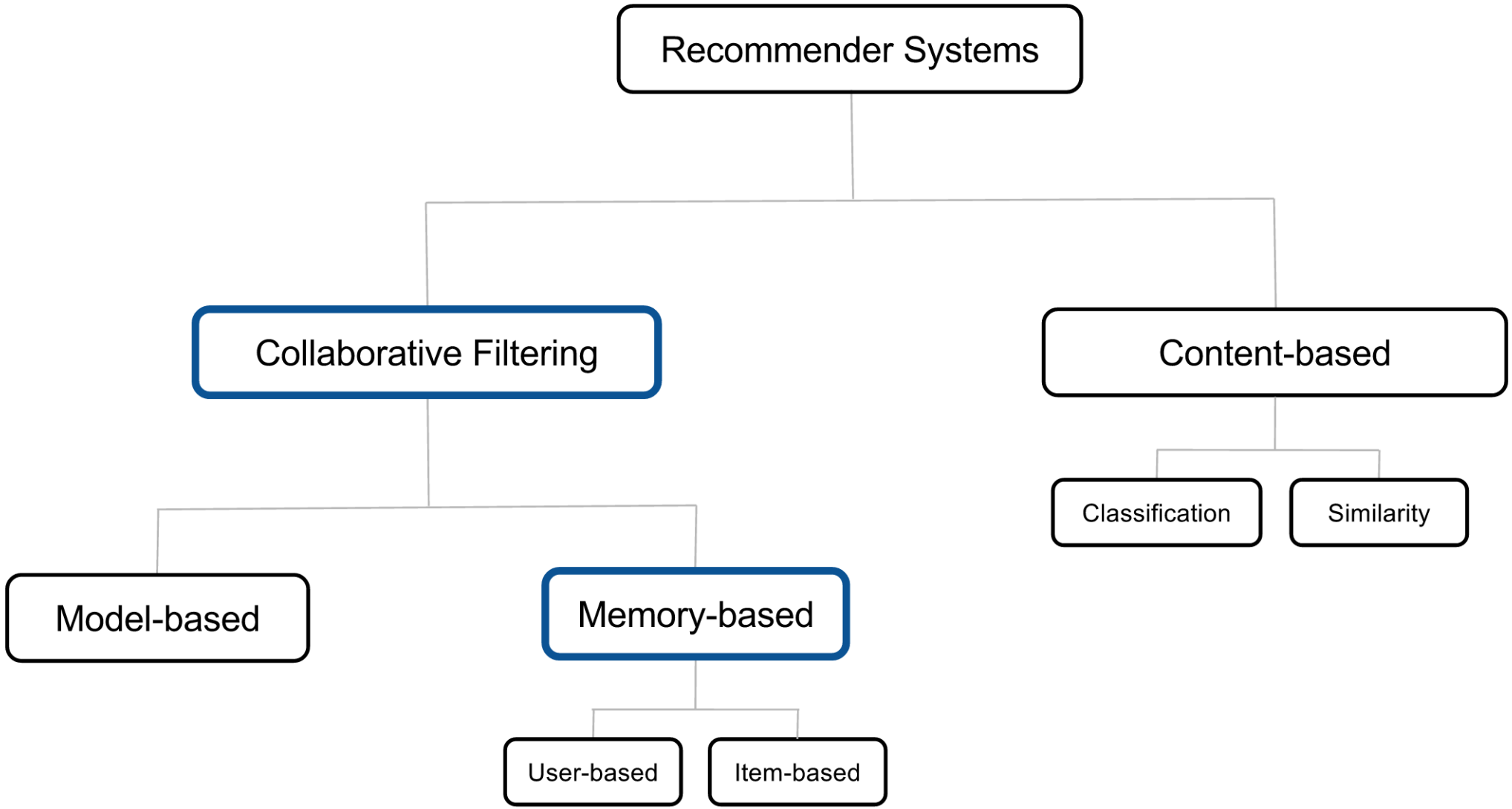
Recommender evaluation

actual / predicted	negative	positive
negative	a	b
positive	c	d

$$Precision = \frac{correctly\ recommended\ items}{total\ recommended\ items} = \frac{d}{b+d}$$

$$Recall = \frac{correctly\ recommended\ items}{total\ useful\ recommendations} = \frac{d}{c+d}$$

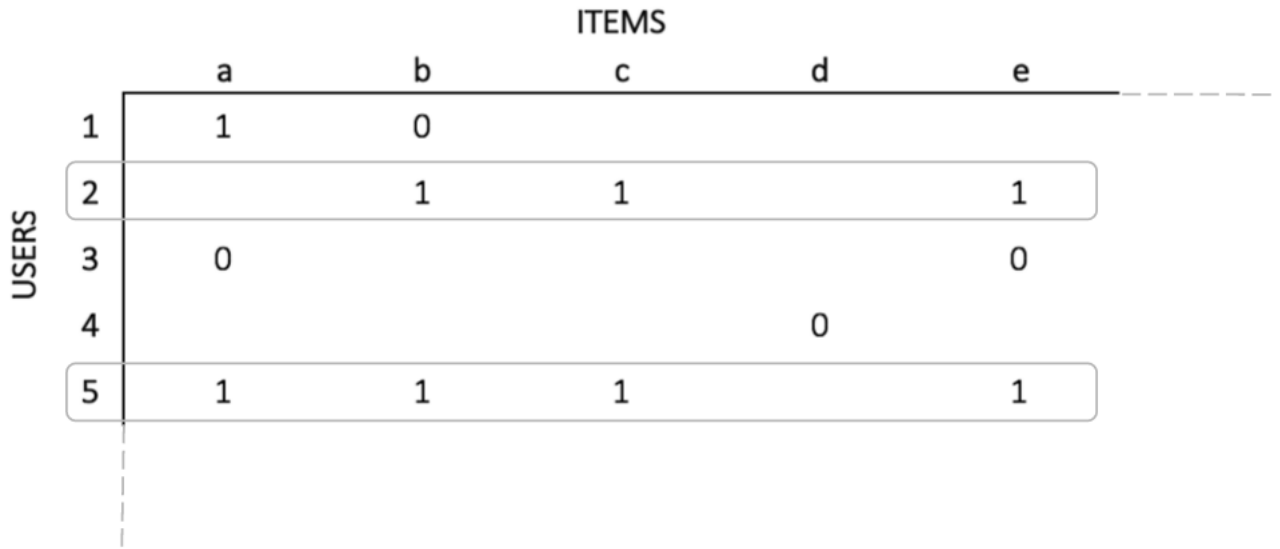
DS Analytics



DS Analytics

User-based

1. Get similarity scores



DS Analytics

User-based

1. Get similarity scores

$$\mathrm{sim}_{\mathrm{Pearson}}(\vec{x}, \vec{y}) = \frac{\sum_{i \in I} (\vec{x}_i \, \bar{\vec{x}})(\vec{y}_i \, \bar{\vec{y}})}{(|I| - 1)\,\mathrm{sd}(\vec{x})\,\mathrm{sd}(\vec{y})}$$

$$\mathrm{sim}_{\mathrm{Cosine}}(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\|\,\|\vec{y}\|},$$

DS Analytics

User-based

1. Get similarity scores
2. Predict blanks as weighted average of neighbours'

$$\hat{r}_{aj} = \frac{1}{\sum_{i \in \mathcal{N}(a)} s_{ai}} \sum_{i \in \mathcal{N}(a)} s_{ai} r_{ij}$$

s = sim
r = rating
N(a) = users in neighbourhood of active user, *a*

DS Analytics

User-based

1. Get similarity scores
2. Predict blanks as weighted average of neighbours'
3. Recommend top n items not already used by user

Problems:

- 'Cold-start': how do you predict prefs of new users?
- Computationally intensive

Item-based

1. Get similarity/distance score between each item vector of users' preferences
2. For a given *active user*, create a weighted list of items similar to their top rated items

DS Analytics

Item-based

1. Get similarity/distance score between each item vector of users' preferences
2. For a given *active user*, create a weighted list of items similar to their top rated items

E.g. Wedding Crashers = 5, Skyfall = 1

*pred_rating(marley & me) = (5 * sim(wedding, marley) + 1 * sim(sky, marley)) / sum(sims)*

DS Analytics

Item-based

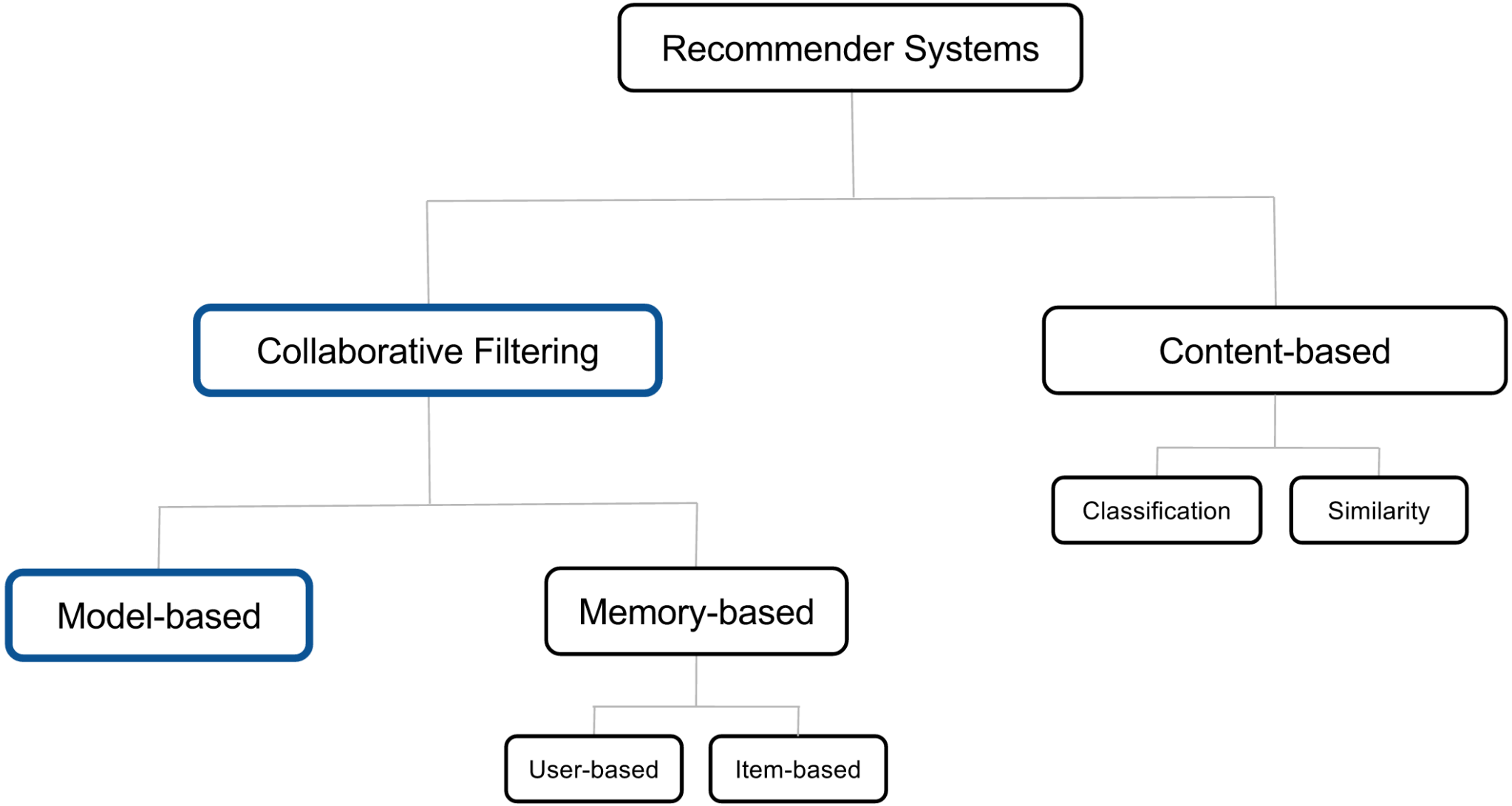
Pros:

1. Better when items available are less varying than users
2. Can pre-calculate similarities matrix
3. Don’t need attributes

Problems:

- ‘Cold-start’: how do you predict prefs of new users?

DS Analytics



DS Analytics

recommenderlab

```
r <- as(m, "realRatingMatrix”)
```

DS Analytics

recommenderlab

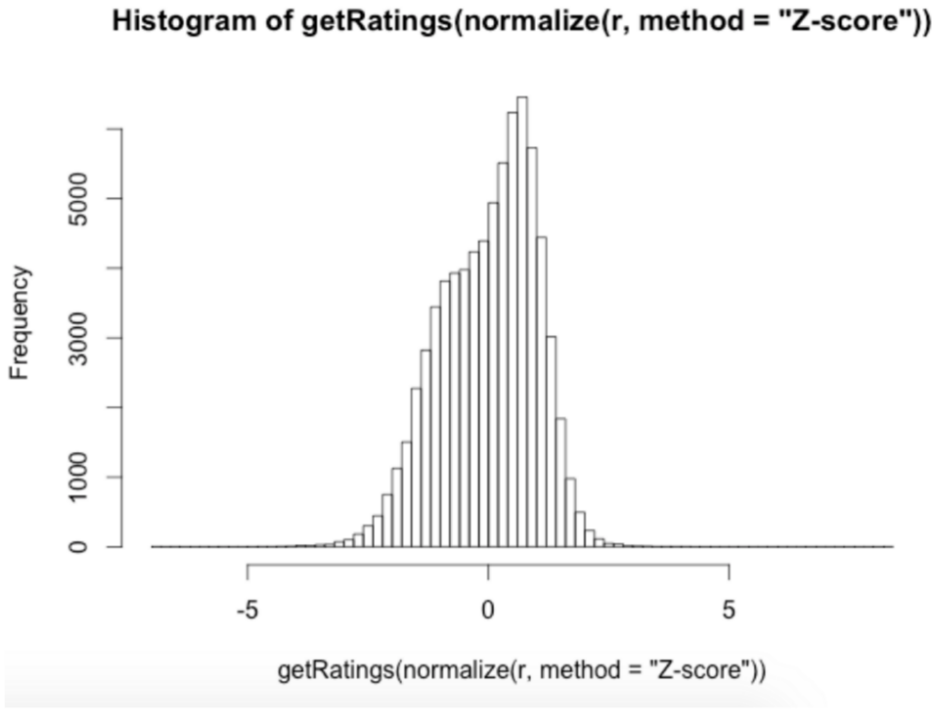
```
r <- as(m, "realRatingMatrix”)
r_m <- normalize(r)
```

DS Analytics

recommenderlab

r <- as(m, "realRatingMatrix”)

r_m <- normalize(r)



DS Analytics

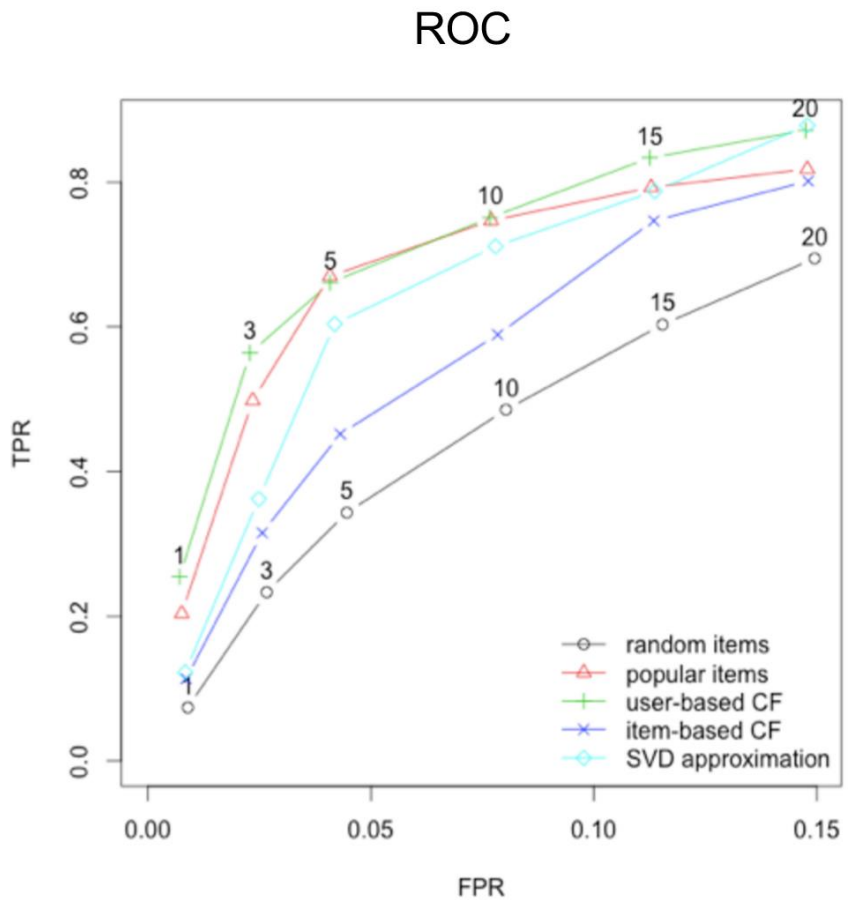
recommenderlab

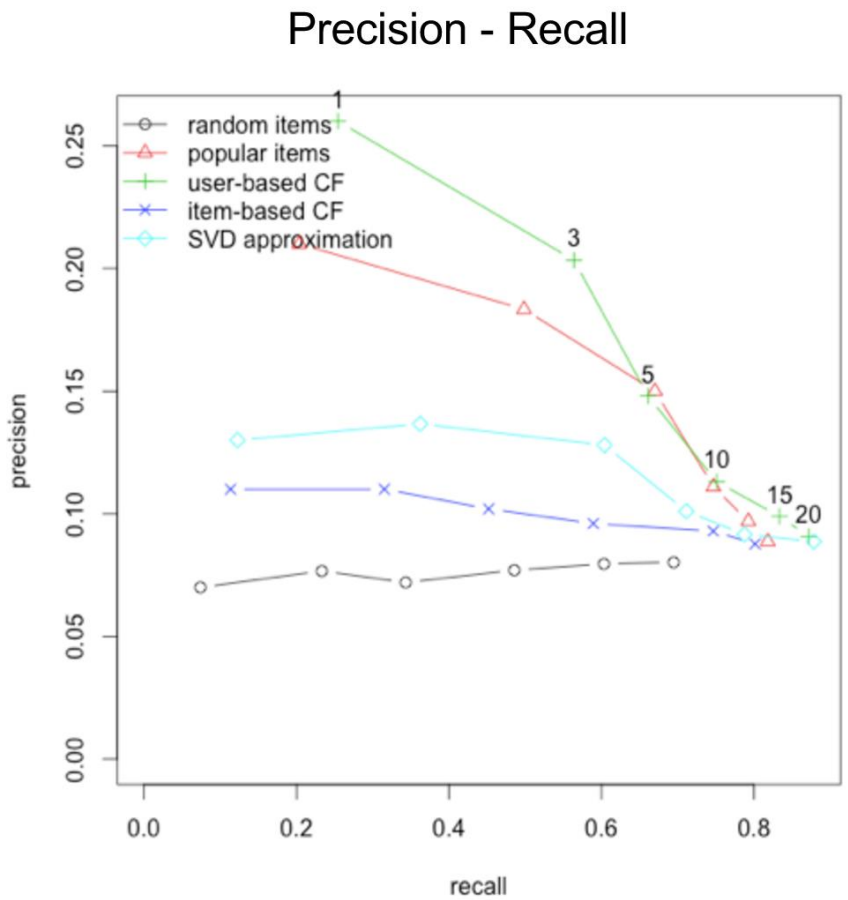
```
e <- evaluationScheme(data, method="split", train=0.8,
                      given=10, goodRating=5)
```

DS Analytics

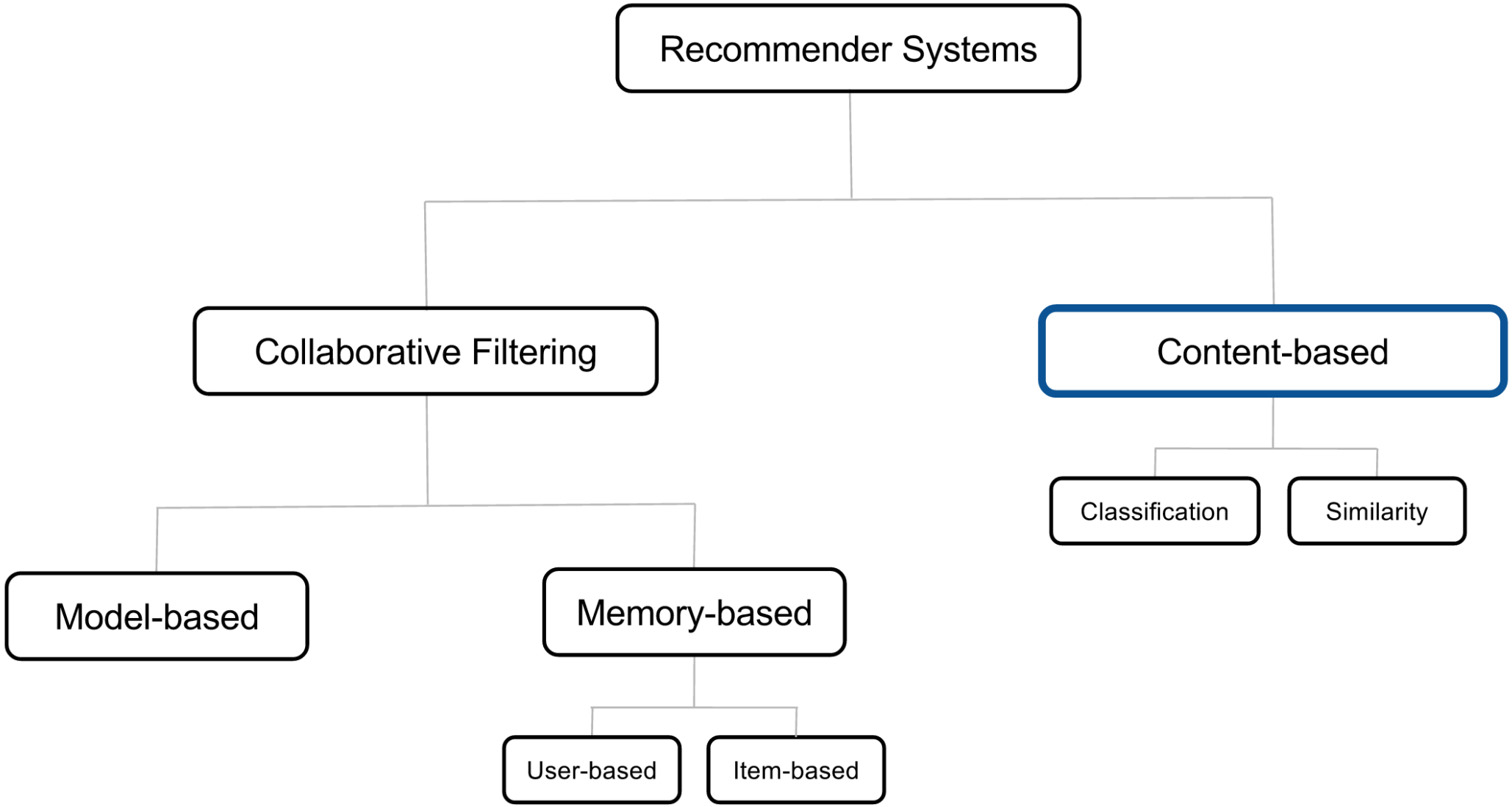
recommenderlab

evaluate(scheme, algorithms, ...)





DS Analytics



DS Analytics

Content-based

Classification method

- Train a model with ratings / behaviour and item features e.g. decision tree
- Problem: need one per user. Only useful when problem is small

DS Analytics

Content-based

Similarity method

- Item *profile* vector, with elements representing presence of feature (e.g. tf-idf for top words)

DS Analytics

Content-based

Similarity method

- Item *profile* vector, with elements representing presence of feature (e.g. tf-idf for top words)

wedding_crashers = { ‘owen_wilson’ : 1, ‘christopher_walken’ : 1, …}

DS Analytics

Content-based

Similarity method

- Item *profile* vector, with elements representing presence of feature (e.g. tf-idf for top words)
- User *profile* vector obtained from weighted item scores, where weights = values in utility matrix

Content-based

Similarity method

- Item *profile* vector, with elements representing presence of feature (e.g. tf-idf for top words)
- User *profile* vector obtained from weighted item scores, where weights = values in utility matrix

christopher_walken = (5 * crashers_profile('walken') + 1 * skyfall_profile('walken')) / (5 + 1)

DS Analytics

Possible approaches for AlphaSights

- **Collaborative filtering** after clustering projects based on project text
  - Problem: will it just propose popular advisors for cluster?
- **Content-based** using project text to create user profile vectors, then similarities between users and items
- **Content-based** using project text to create user profile vectors, then similarities between active project and others, then recommend items as weighted average of similar projects' items?
- **Content-based** classification model. Cluster projects by text, use as levels in predictive model
- Hierarchical cluster - most popular advisors

DS Analytics

Thanks

duncan@dsanalytics.co.uk

DS Analytics