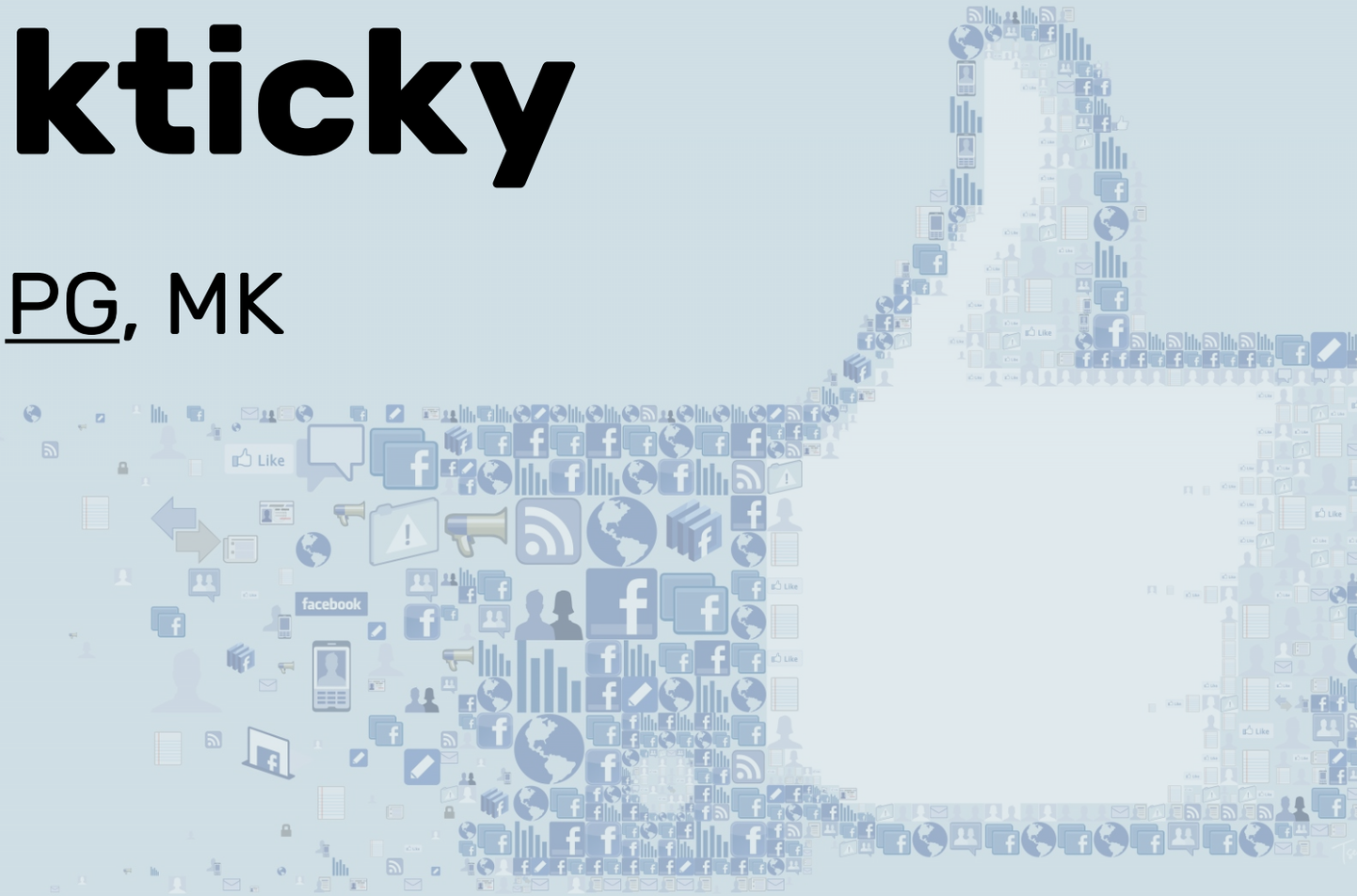
Odporúčanie prakticky

PG, MK



8.3.2018

Zdroj obrázka: www.wallpaperswide.com

datalys

Ako odporúčate?

Ako riešite úlohy ML?

Odporúčanie

Výskumníci i prax sa ním zaoberá veľa rokov.

Vzniklo viacero knižníc na odporúčanie. Prečo?

znovupoužiteľnosť,

priamočiare porovnanie kvality odporúčania,

užitočnosť pre prax.

MyMediaLite

.NET

Akademické pozadie (University of Hildesheim)

.NET (C#) - podpora aj pre Linux, OS X

Dostupné viaceré algoritmy a možnosť tvorby vlastných:

Predikcia položiek (BPRMF, ItemKNN, WRMF, BPRSLIM)

Predikcia hodnotenia (SlopeOne, BPSO, Latent-feature log linear, Matrix factorization with factor-wise learning).

Serializácia, inkrementálne aktualizácie modelov,

MyMediaLite

.NET

```
/// <inheritdoc/>
public override double Predict(int user_id, int item_id)
{
    if (item_id > MaxItemID)
        return global_average;

    double prediction = 0.0;
    int frequency = 0;

    foreach (RatingEvent r in Ratings.ByUser[user_id])
    {
        int f = freq_matrix[item_id, r.item_id];
        if (f != 0)
        {
            prediction += ( diff_matrix[item_id, r.item_id] + r.rating ) * f;
            frequency += f;
        }
    }

    if (frequency == 0)
        return global_average;

    return (double) prediction / frequency;
}
```

MyMediaLite

.NET

http://www.mymedialite.net/examples/index.html

Využitie MyMediaLite na KDD Cup 2011

Výsledky state-of-the-art algoritmov odporúčania

Zoznam prác, ktoré využívajú MyMediaLite

Výkonnosť

mREC

Python

Modely:

SLIM (odporúčanie pomocou lineárnej regresie)

WRMF - váhovaná maticová faktorizácia

Weighted approximately ranked pairwise ranking loss (WARP)

Umožňuje aj hybridné odporúčanie s obsahom

Predspracovanie, rôzne metriky vyhodnotenia, paralelné odporúčanie (IPython).

mREC - Ukážky

Python

Predspracovanie:

```
mrec_prepare --dataset ml-100k/u.data --outdir splits --rating_thresh 4 --
test_size 0.5 --binarize
```

Trénovanie:

```
mrec_train -n4 --input_format tsv --train "splits/u.data.train.*" --outdir models
```

Predikcia:

```
mrec_predict --input_format tsv --test_input_format tsv --train "splits/
u.data.train.*" --modeldir models --outdir recs
```

Vyhodnotenie:

```
mrr             0.6541 +/- 0.0023
prec@5          0.4082 +/- 0.0016
prec@10         0.3529 +/- 0.0010
prec@15         0.3180 +/- 0.0009
prec@20         0.2933 +/- 0.0008
```

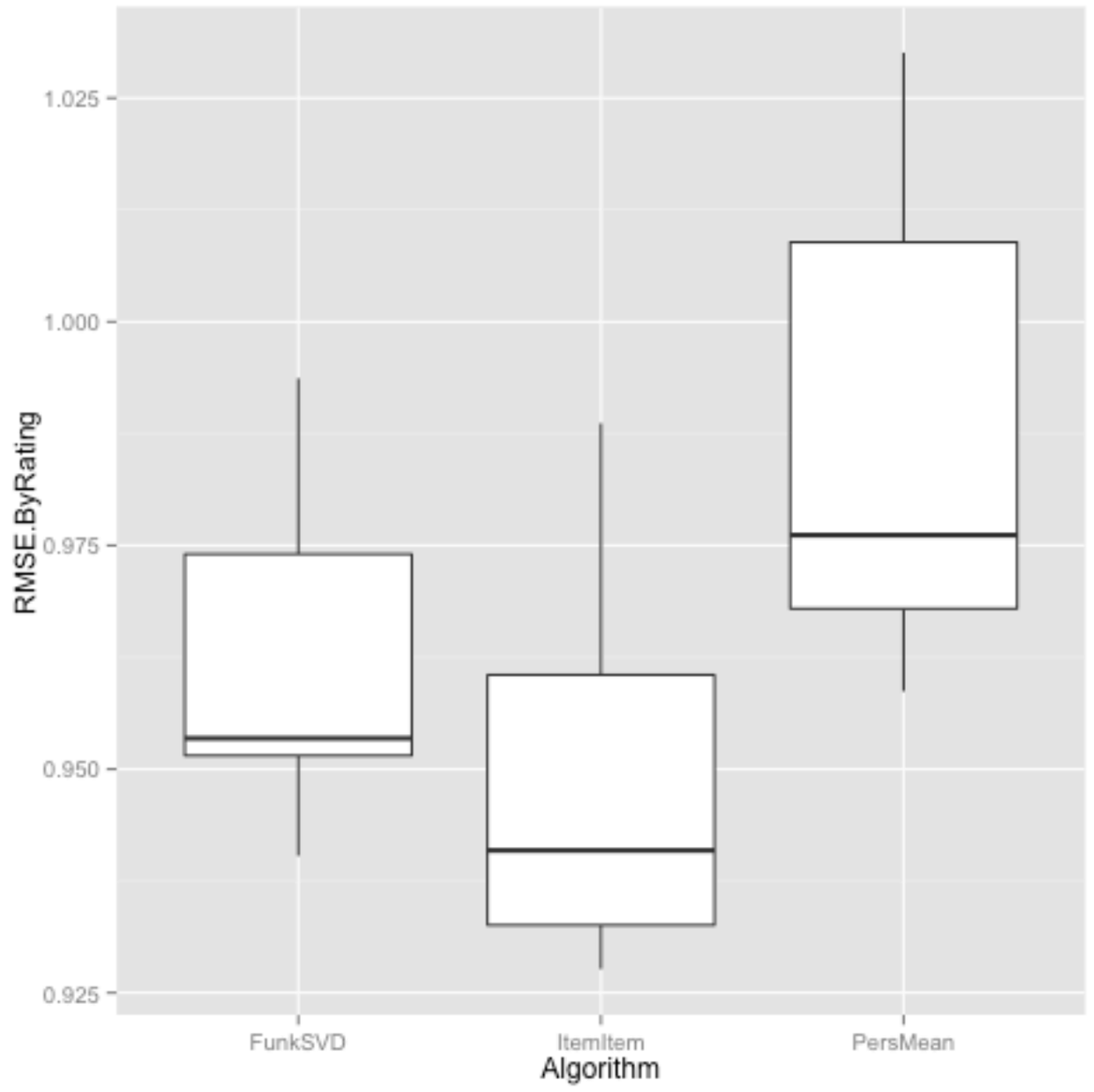
LensKit

Java

Pôvod v GroupLens, dodnes využívajú v rámci MovieLens.

Viaceré prístupy:

- item-based CB, user-based CB, matrix factorization (FunkSVD), slope-one predikcia hodnotenia.



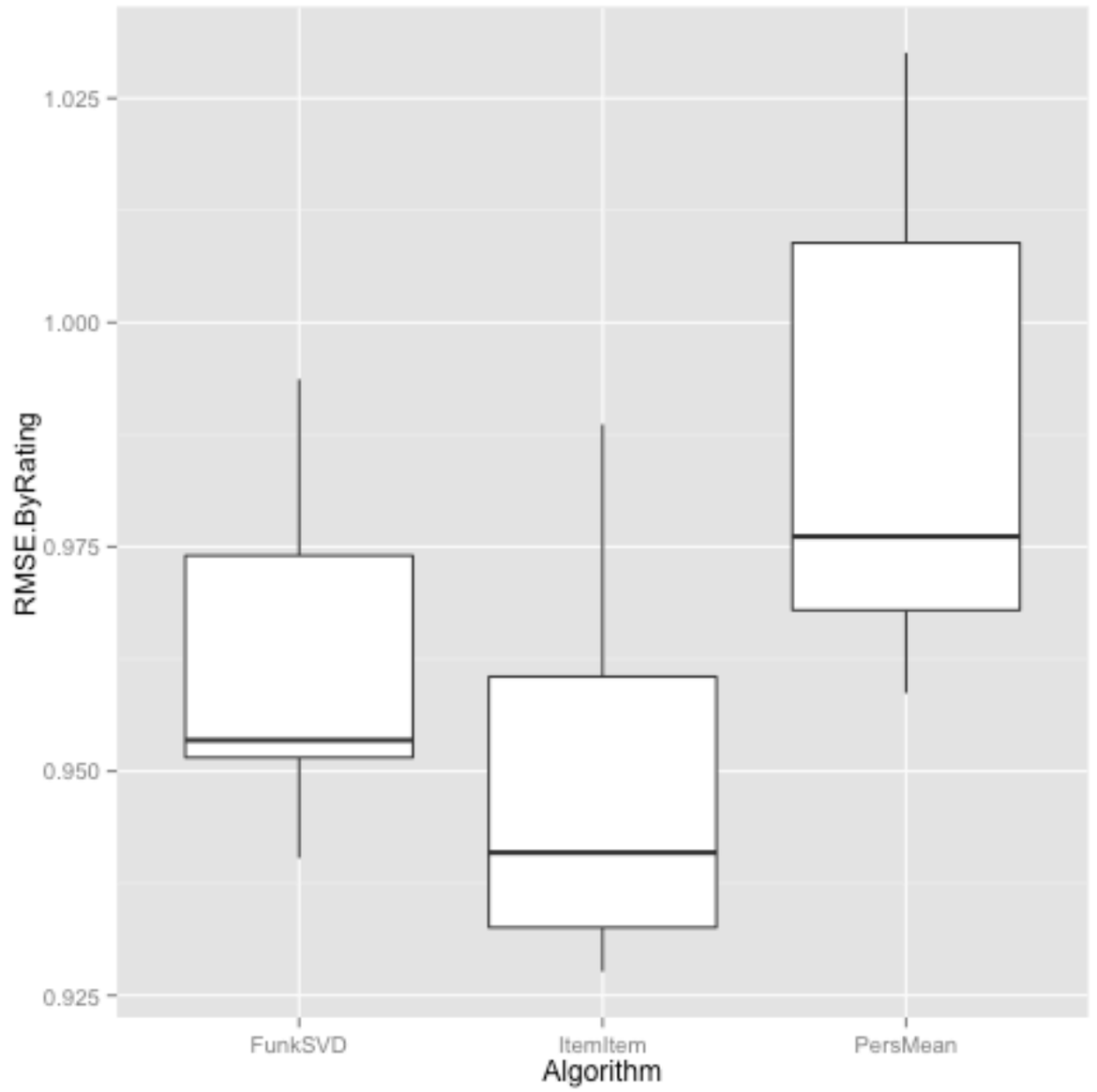
LensKit

Java

Jednoduchá vizualizácia metrík cez R.

Napriek tomu, že je to Java, málo ukecané.

```
metric topNnDCG {
    listSize 10
    candidates ItemSelectors.allItems()
    exclude ItemSelectors.trainingItems()
}
```



Surprise

Python

Základné prístupy, neighborhood methods, matrix factorization (SVD, PMF, SVD++, NMF), Slope-One, co-clustering. Rôzne metriky na vyhodnotenie (cosine, MSD, pearson…).

Nielen odporúčanie, ale aj práca s dátami (dataframy, matice, cross-validácia,..).

Jednoduchá schéma pre vytvorenie vlastného odporúčača.

Pomerne kvalitná dokumentácia.

Surprise

Python

```
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate

data = Dataset.load_builtin('ml-100k')

algo = SVD()

cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE              0.9311  0.9370  0.9320  0.9317  0.9391  0.9342  0.0032
MAE               0.7350  0.7375  0.7341  0.7342  0.7375  0.7357  0.0015
Fit time          6.53    7.11    7.23    7.15    3.99    6.40    1.23
Test time         0.26    0.26    0.25    0.15    0.13    0.21    0.06
```

Surprise - zaujímavosť

Keď zistíte, že pre používateľa nemôžete odporúčať, **vyhodíte výnimku**.

```
from surprise import PredictionImpossible
raise PredictionImpossible
```

Výnimku zachytí metóda `predict()` a pre používateľa a položku bude nastavené globálne priemerné hodnotenie.

LightFM

Python

Základ: kolaboratívny prístup (MF)

Možnosť obohatiť o obsahové dáta - používateľov aj položiek.

Implicitná (klik) aj explicitná (hodnotenie) spätná väzba.

Vektory používateľov a čŕt aj ako **vstup pri generovaní odporúčaní**.

LightFM model

Python

Viaceré stratové funkcie:

logistic

BPR: Bayesian Personalised Ranking

WARP: Weighted Approximate-Rank Pairwise (mREC)

k-OS WARP

Dokážu pracovať **s pozitívnou aj negatívnou spätnou väzbou**. Sú využívané na latentnú reprezentáciu používateľských preferencií.

LightFM

Python

BPR: pairwise loss. Maximalizuje rozdiel v predikcii pre pozitívny a náhodný negatívny príklad (vhodné aj pri positive only spätnej väzbe).

WARP. Maximalizuje poradie pozitívnych príkladov opakovaným výberom tých negatívnych, kým nenarazí na taký, ktorý “kazí” správne poradie. Potom vykoná gradient update, aby mal pozitívny príklad vyššie poradie ako negatívny.

k-OS WARP. Podobné ako WARP, využíva k-ty pozitívny príklad pre učenie v páre.

LightFM

Python

Odporúčanie (a vyhodnotenie) na pár riadkov

```
from lightfm import LightFM
from lightfm.datasets import fetch_movielens
from lightfm.evaluation import precision_at_k

# Načítanie MovieLens 100K datasetu.
data = fetch_movielens(min_rating=5.0)

# Vytvorenie a natrénovanie
model = LightFM(loss='warp')
model.fit(data['train'], epochs=30, num_threads=2)

# Vyhodnotenie natrénovaného modelu
test_precision = precision_at_k(model, data['test'], k=5).mean()
```

Pôvodne hodnotenia

FluRS

Python

Vhodné pre prúdové dáta

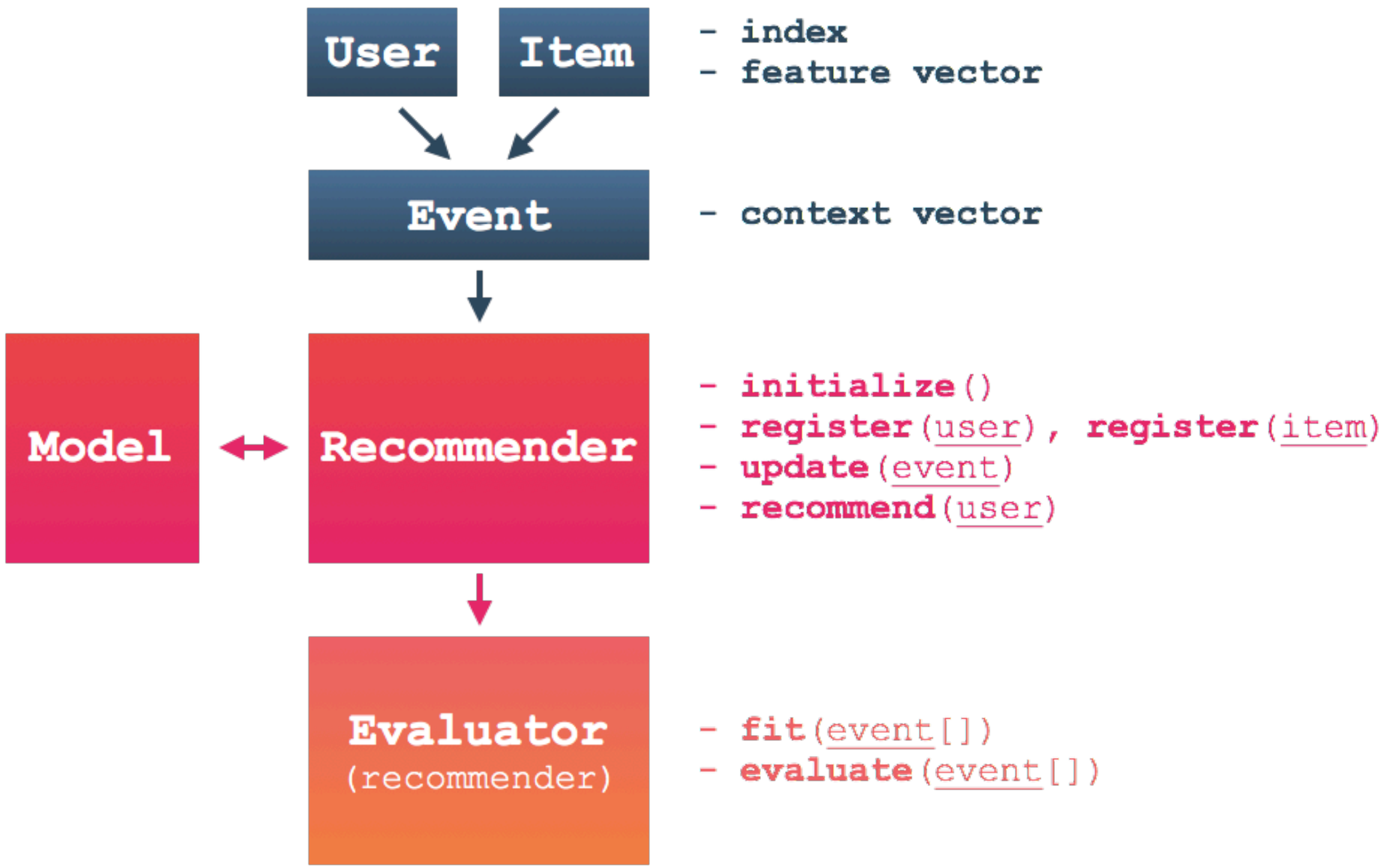
- inkrementálna aktualizácia modelu odporúčania,
- používateľ, položka, udalosť (môže mať hodnotu a kontext),
- pri odporúčaní možno zohľadniť aj **kontext**.

Jednoduchá schéma pre vytvorenie vlastného odporúčača.

Projekt je ešte pomerne v “plienkach”.

FluRS

Python



datalys

https://takuti.github.io/flurs/

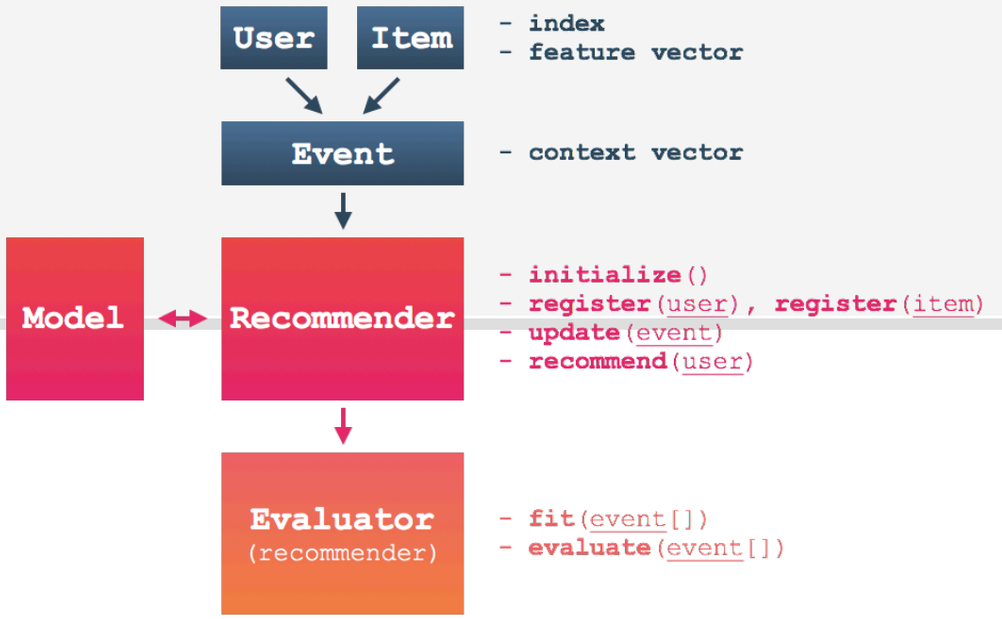
Feature-based recommender - Factorization Machine (FM)

Create prediction model from context-aware feature vectors

Feature vector **x**																					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...		
	User				Movie					Other Movies rated						Last Movie rated						
$x^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	$y^{(1)}$
$x^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$x^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$x^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$x^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$x^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$x^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

S. Rendle. Factorization Machines with libFM. ACM Transactions on Intelligent Systems and Technology, 3(3), May 2012.

FluRS



Výhody:

trénovanie z matice aj inkrementálne,

priamočiara možnosť vytvoriť vlastný odporúčač.

Nevýhody:

existujúce algoritmy zamerané na prúdové spracovanie,

stále vo vývoji (to sa prejavuje aj na množstve algoritmov, evaluácii a dokumentácii).

Recommenderlab

Snaha o vytvorenie prostredia pre vývoj a vyhodnotenie odporúčania – najmä z hľadiska výskumu.

Konzistentné a efektívne spracovanie dát.

Jednoduchá integrácia algoritmov (vlastné aj existujúce).

Príprava experimentov a ich vyhodnotenie.

Veľmi podrobná dokumentácia.

Recommenderlab

Rozdelenie na train a test cez:

Splitting (náhodne)

Bootstrapovanie (z testovacej množiny vyberáme používateľov s opakovaním. Nová testovacia množina budú všetci tí, ktorí nie sú v traine.)

K-fold cross-validáciu

Recommenderlab


```
# Binárne dáta
data(MSWeb)
MSWeb10 <- sample(MSWeb[rowCounts(MSWeb) >10,], 50)
e <- evaluationScheme(MSWeb10, method="split", train=0.9, k=1, given=3)

# User-based odporúčanie
r <- Recommender(getData(e, "train"), "UBCF")

# Predikcia
p <- predict(r, getData(e, "known"), type="topNList", n=10)

# Výpočet správnosti
calcPredictionAccuracy(p, getData(e, "unknown"), given=3)
```

Prediction IO

Scala

Open-source server pre strojové učenie

Apache Spark, MLlib, HBase, Spray a Elasticsearch

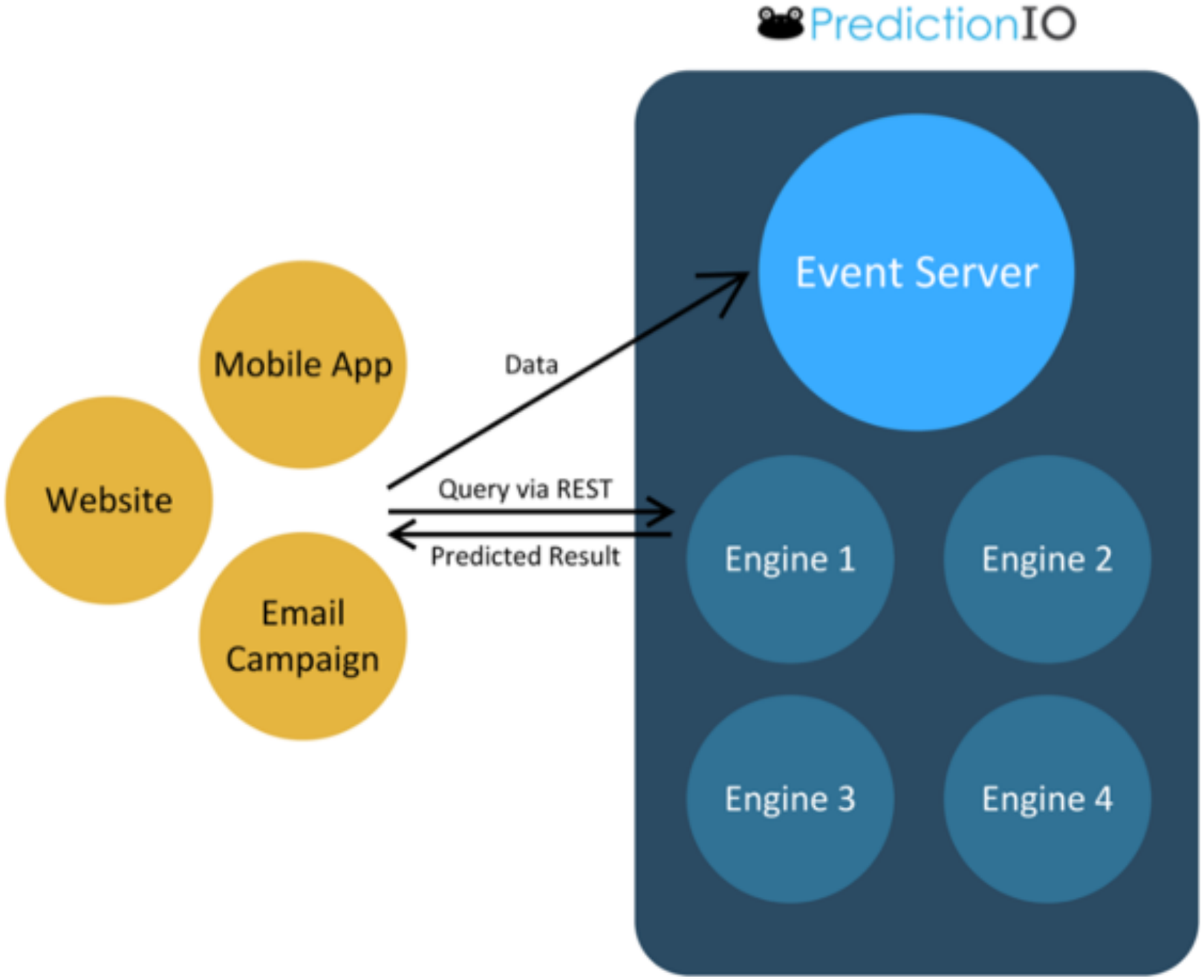
Nasadenie servera na produkciu ako celok.

Podpora viacerých knižníc na strojové učenie (napr. Spark MLLib, OpenNLP).

SDK pre Javu, Python, Ruby, PHP

Prediction IO

Scala



Prediction IO

Scala

DASE.

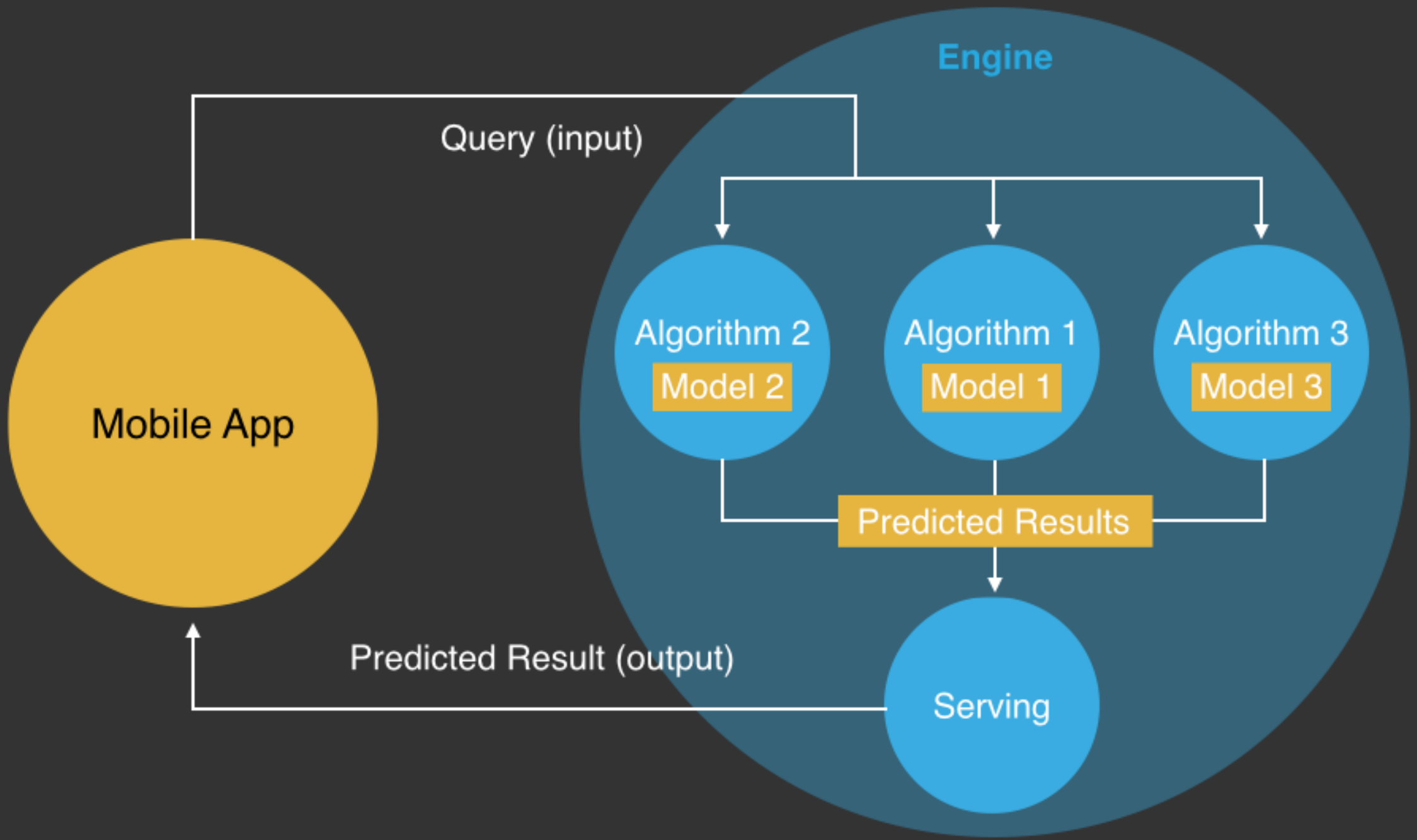
[D] Data Source and Data Preparator

[A] Algorithm

[S] Serving

[E] Evaluation Metrics

Engine B. Respond to dynamic query



Záver

Prečo ich používať?

Viete si jednoducho vytvoriť **vlastný odporúčač**.

Ak máte dataset, ktorý komunita nepozná, viete ho **overiť a porovnať s vaším prístupom**.

Mnohé veci implementujú efektívnejšie, ako by ste ich implementovali vy :)

Na čo si treba dať pozor

Rozdelenie na trénovaciu a testovaciu množinu (na validačnú sa veľakrát zabúda).

Metriky sú pri odporúčaní stále problém (častokrát si vyhodnotenie musíte naprogramovať sami).

Predikcia hodnotenia vs. predikcia poradia.

Implicitná vs. explicitná spätná väzba.

Sumarizácia

Dnes už to zďaleka nie je len o kolaboratívnom odporúčaní.

Existuje viacero prístupov kombinujúcich **kolaboratívne** s **obsahovým**, kde navyše už vystupuje aj **kontext**.

Knižnice na odporúčanie nám poskytujú mnohé existujúce prístupy a umožňujú vytvoriť aj vlastné.

Prehľad

Knižnica	Jazyk	Predikcia poradia	Predikcia hodnotenia	Hybridné odporúčanie
MyMediaLite	C#	✓	✓	✓
mREC	Python (2)	✓	✗	✓
LensKit	Java	✓	✓	✓
Surprise	Python	✗	✓	✗ *
FluRS	Python	✓	✗	✗ *
LightFM	Python	✓	✗	✓
Recommenderlab	R	✓	✓	✓

* Iba vlastná implementácia.

Datasety

Grouplens (https://grouplens.org/datasets/)

YELP (https://www.yelp.com/dataset/challenge)

Yahoo! Music (https://webscope.sandbox.yahoo.com/catalog.php?datatype=r)

Spotify 1M playlist (https://recsys-challenge.spotify.com)

Zdroje

Kompan, M., Gaspar, P., and Bielikova, M. (2018). Hybrid collaborative recommendations (prijaté). In Berkovsky, S., Cantador, I., and Tikk, D., editors, *Collaborative Recommendations: Algorithms, Practical Challenges and Applications.*