# Johannes Braun


**Final project, 30/04/15**

braun.johannes@gmail.com
linkedin.com/in/johannesbraun1
johannesbraun.github.io
github.com/johannesbraun


**Sven Marquardt, Bouncer Berghain**

# Project:

## Personal local electronic music radio

Scope :

- **Local**: Berlin
- **Radio**: Autoplay of generated playlists
- **Personal**: Responsive to feedback 
- **Music**: Focus on electronic music

Motivation :

- Scoop into Berghain

# Agenda

1. Motivation
2. General approach & Data Engineering
3. Data Science:
    - Generating Recommendations
    - Measuring success
4. Launching the radio service
5. Demo

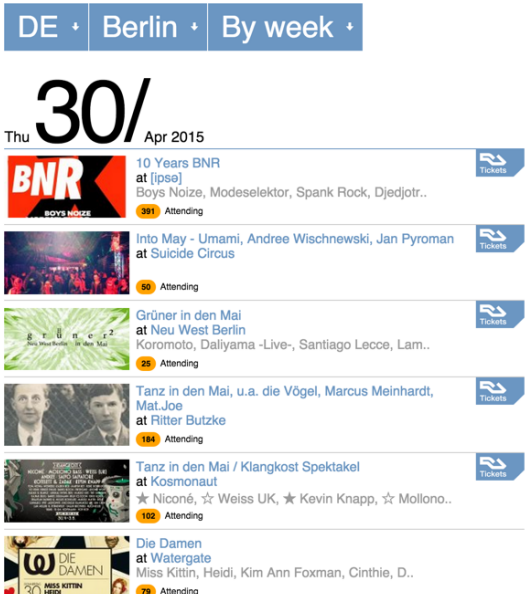
## 2. Approach & Data Engineering:

# Find out about local events

**Resident Advisor (RA)** is an online music magazine and community platform that's dedicated to showcasing electronic music, artists and events across the globe.

Top Berlin clubs

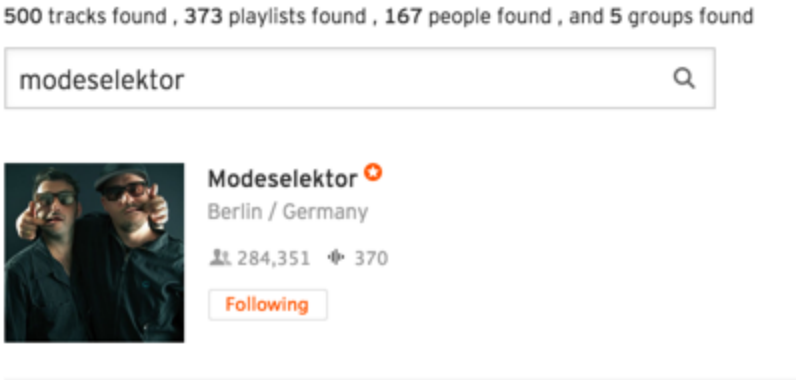
Berghain/Panorama Bar /
Watergate /
Club der Visionaere /
Tresor /
Weekend /
://about blank /
Salon Zur Wilden Renate /
Golden Gate /
Ritter Butzke /
Arena Club /
Suicide Circus /
Stattbad /




## 2. Approach & Data Engineering:

# Search for lineup on soundcloud

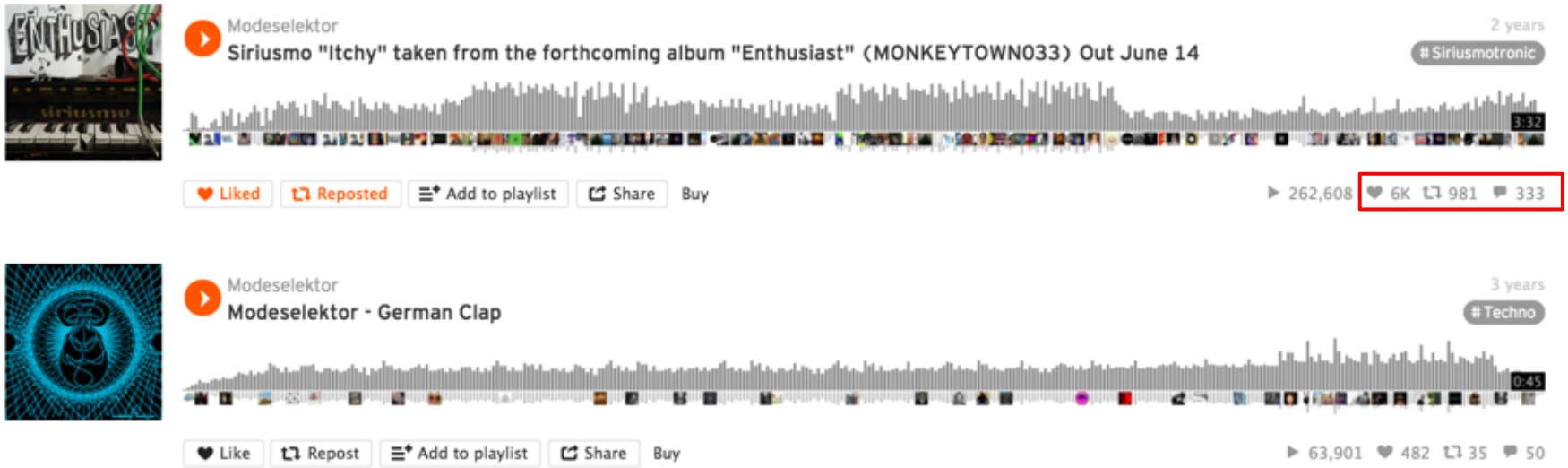
SoundCloud is an online audio distribution platform that enables its users to upload, record, promote, and share their originally-created sounds. The API exposes resources like sounds, sets and users



## 2. Approach & Data Engineering:

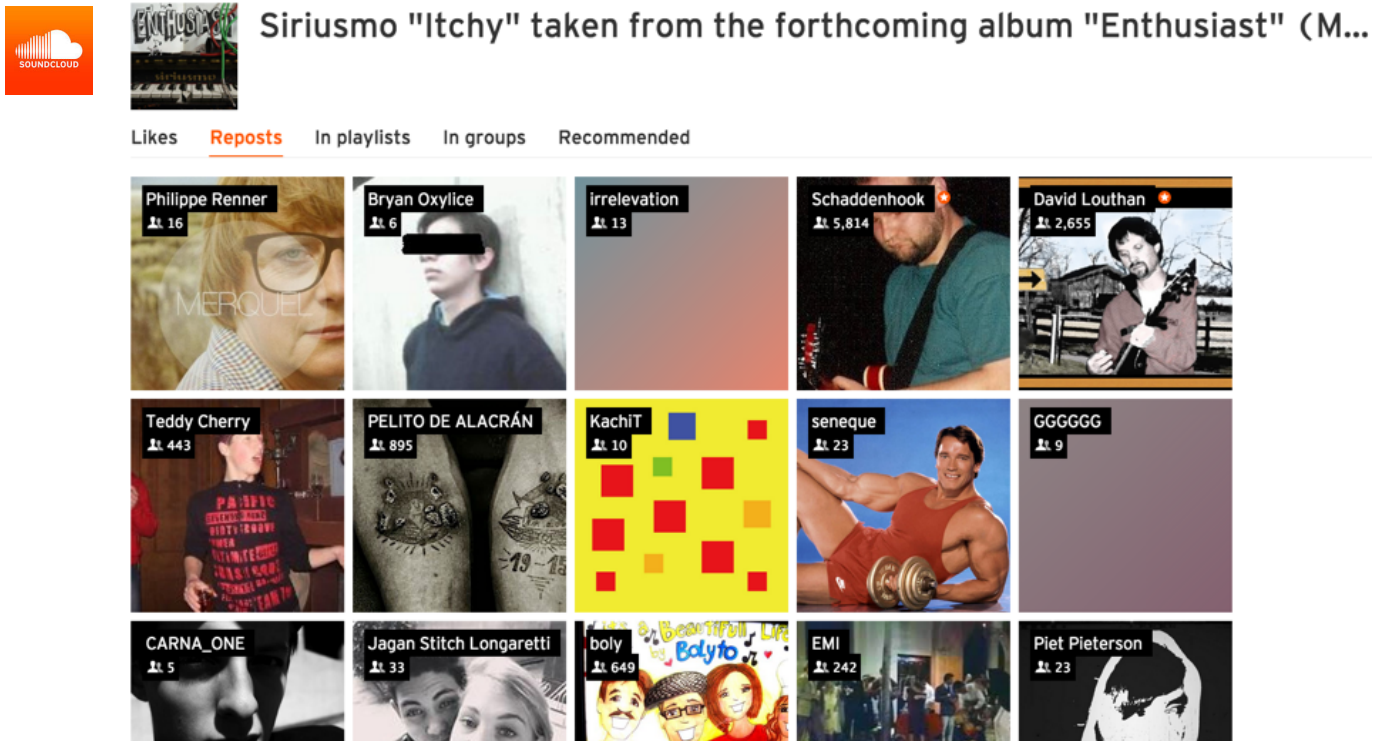
# Extract soundcloud user preferences

API: Find the set of users that have liked
*Modeselektor: Siriusmo "Itchy"*



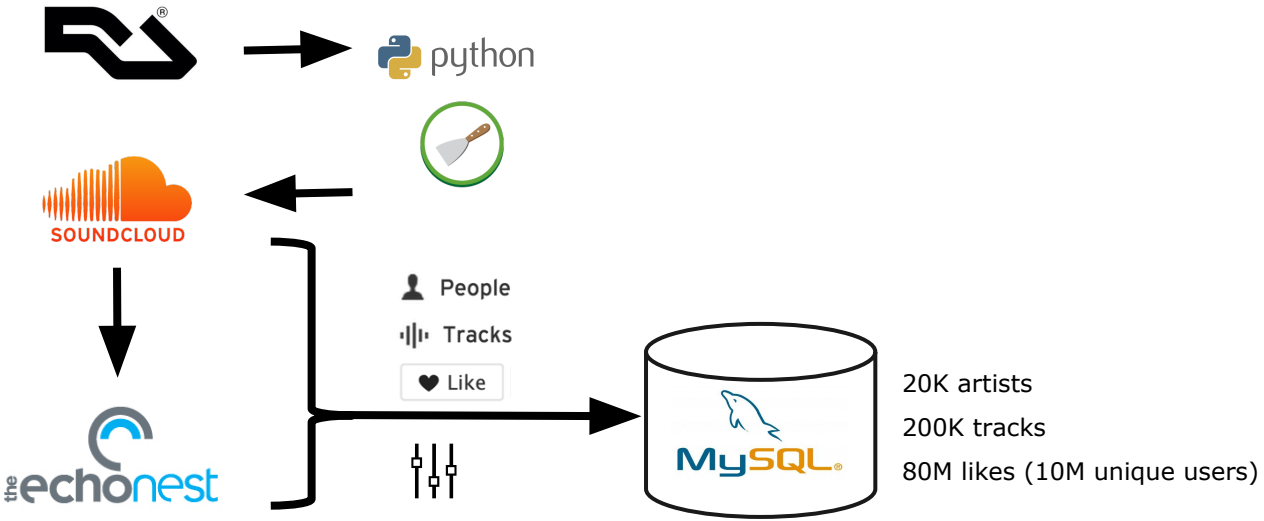
## 2. Approach & Data Engineering:

# Extract soundcloud user preferences



## 2. Approach & Data Engineering:

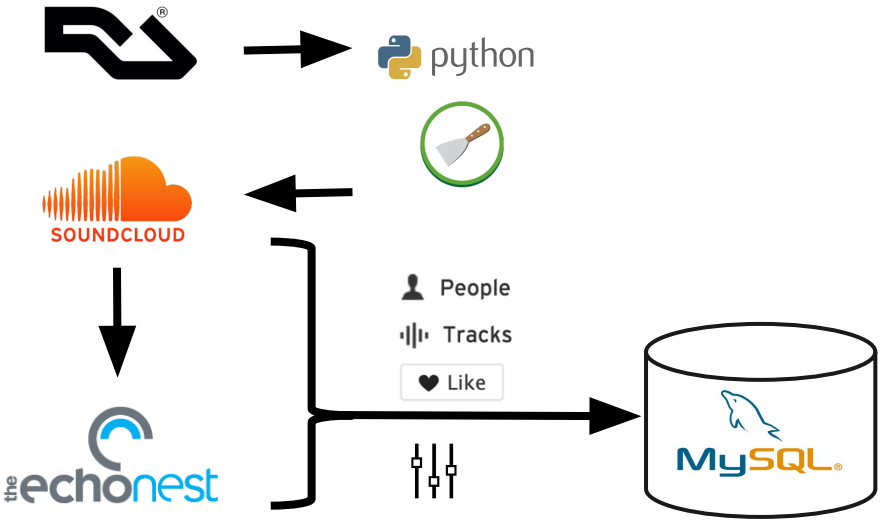
# Extracting Audio Features



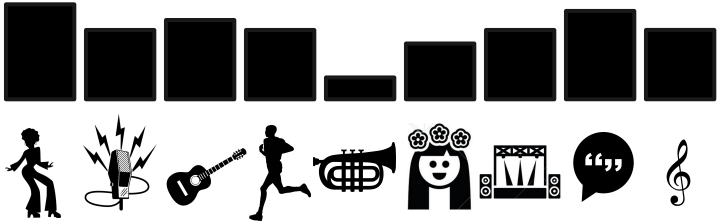

amazon web services™

## 2. Approach & Data Engineering:

# Extracting Audio Features




The Echo Nest is the industry's leading music intelligence company. The API offers a service for analyzing audio files:



amazon web services™

# Agenda

1. Motivation
2. Approach & Data Engineering
3. Data Science:
    - Generating Recommendations
    - Measuring success
4. Launching the radio service
5. Demo

## 3.1 Data Science: Recommendations

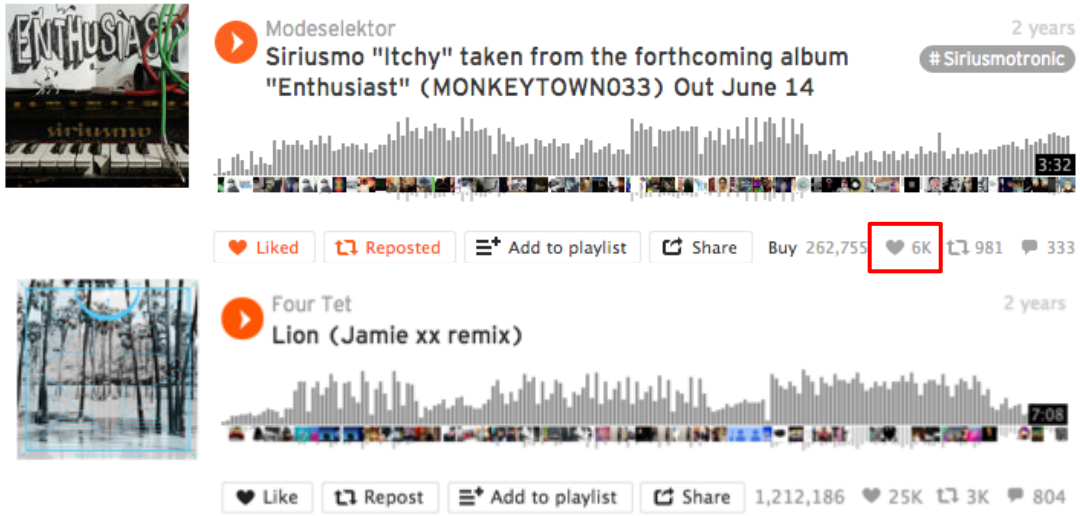
# a) Collaborative Filtering

1) Find users that like *Modeselektor: Siriusmo "Itchy"*

## 3.1 Data Science: Recommendations

# a) Collaborative Filtering

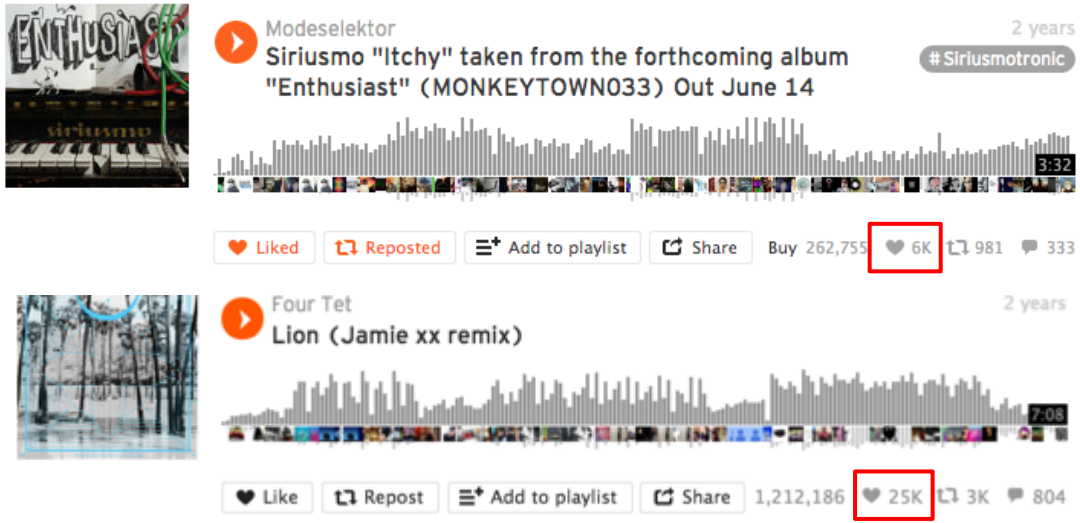
1) Find users that like *Modeselektor: Siriusmo "Itchy"*
2) Find other favorites of these users



## 3.1 Data Science: Recommendations

# a) Collaborative Filtering

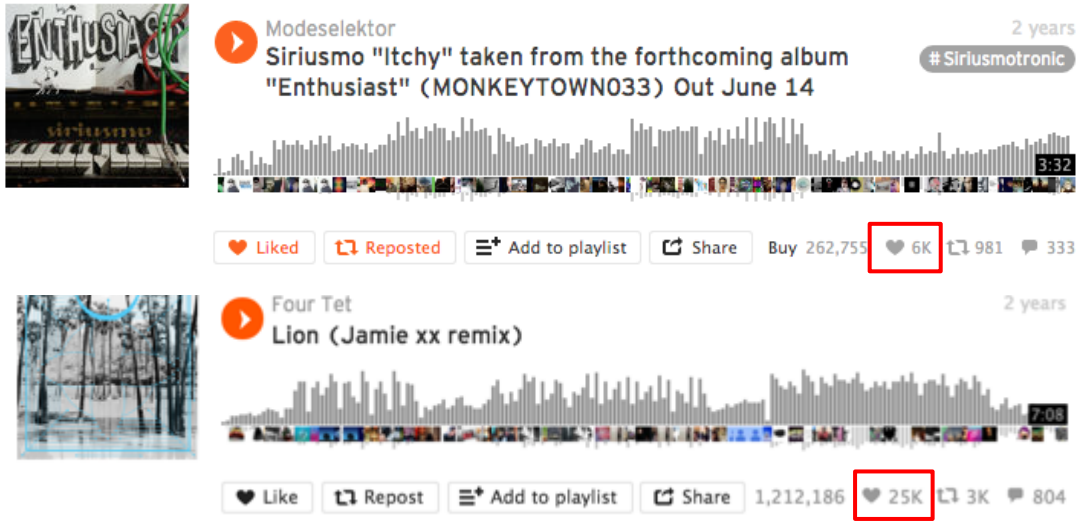
1) Find users that like *Modeselektor: Siriusmo "Itchy"*
2) Find other favorites of these users
3) Rank other tracks by # of co-occurrence



## 3.1 Data Science: Recommendations

# a) Collaborative Filtering

1) Find users that like *Modeselektor: Siriusmo "Itchy"*
2) Find other favorites of these users
3) Rank other tracks by # of co-occurrence




**Log-Likelihood-Ratio:**
Normalizing co-occurrence

| | A | !A |
|---|---|---|
| B | A & B<br>212 | !A & B<br>5.8K |
| !B | A & !B<br>24.8K | !A & !B<br>10M |

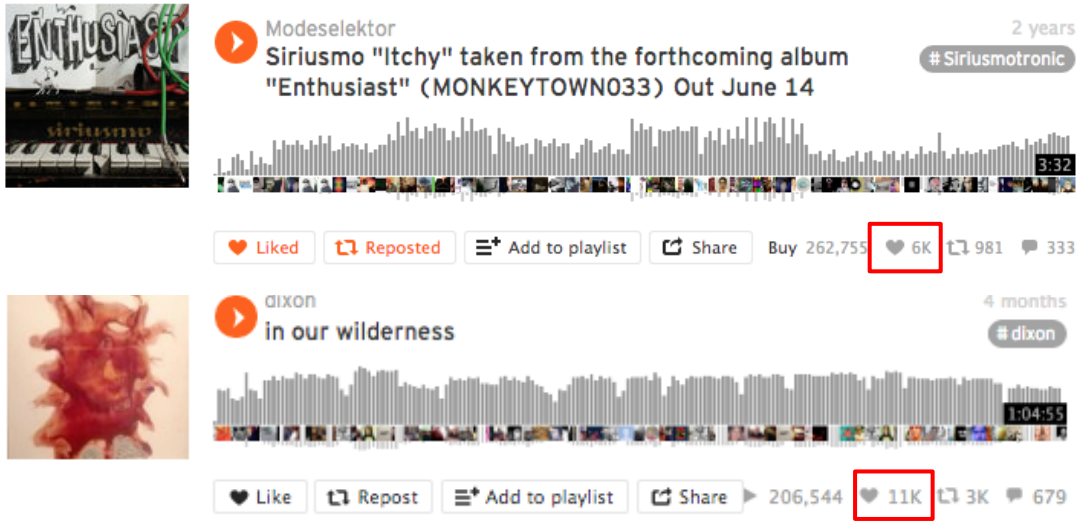
LLR = 525.2563

http://tdunning.blogspot.de/2008/03/surprise-and-coincidence.html

## 3.1 Data Science: Recommendations

# a) Collaborative Filtering

1) Find users that like *Modeselektor: Siriusmo "Itchy"*
2) Find other favorites of these users
3) Rank other tracks by # of co-occurrence




**Log-Likelihood-Ratio:**
Normalizing co-occurrence

| | A | !A |
|---|---|---|
| B | A & B<br>86 | !A & B<br>5.9K |
| !B | A & !B<br>9.9K | !A & !B<br>10M |

LLR = 202.2691

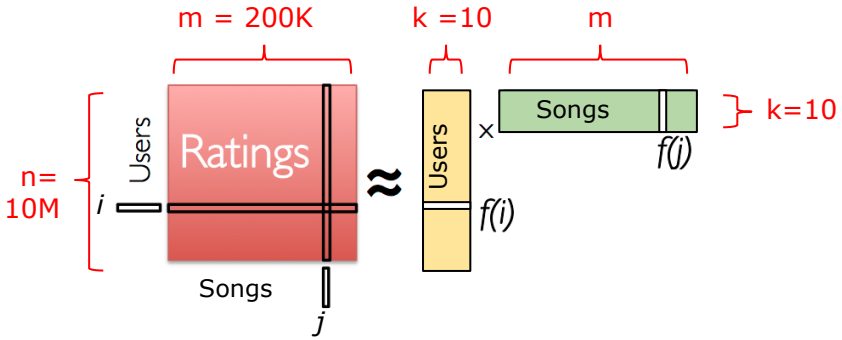
http://tdunning.blogspot.de/2008/03/surprise-and-coincidence.html

## 3.1 Data Science: Recommendations

# b) Matrix factorization

**Alternating Least Squares (Apache Spark):**

Assume small # of k latent factors determine preference:



## 3.1 Data Science: Recommendations

# b) Matrix factorization

Approximate ratings to hopefully generalize to new users:

Songs

Users

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |

≈

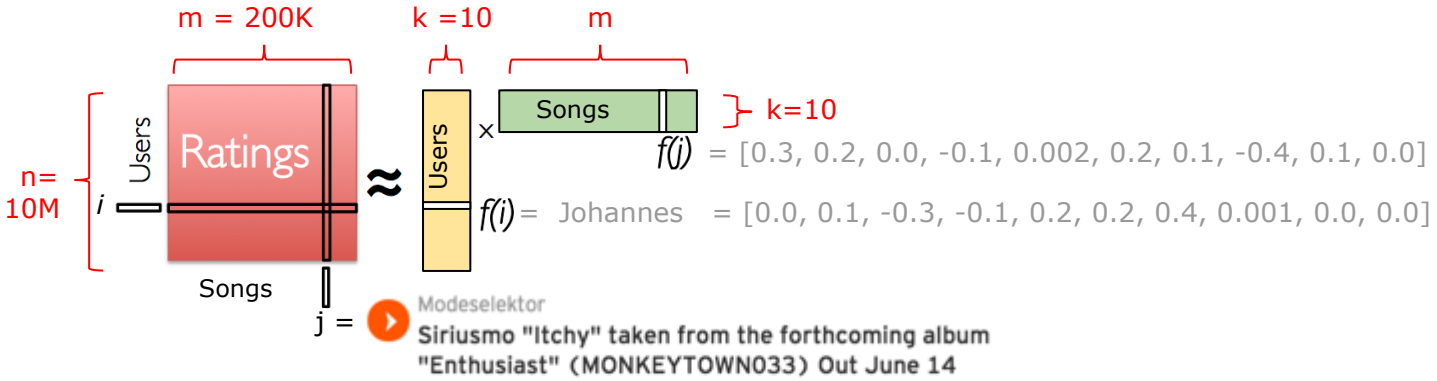
| | | | | |
|---|---|---|---|---|
| 0.96 | 0.99 | 0.99 | 0.38 | **0.93** |
| **0.44** | 0.39 | 0.98 | -0.11 | 0.39 |
| **0.70** | 0.99 | 0.42 | 0.98 | 0.98 |
| 1.00 | **1.04** | 0.99 | 0.44 | 0.98 |
| 0.11 | 0.51 | -0.13 | 1.00 | **0.57** |
| 0.97 | 1.00 | 0.68 | 0.47 | **0.91** |

## 3.1 Data Science: Recommendations

# b) Matrix factorization

**Alternating Least Squares (Apache Spark):**

Assume small # of k latent factors determine preference:

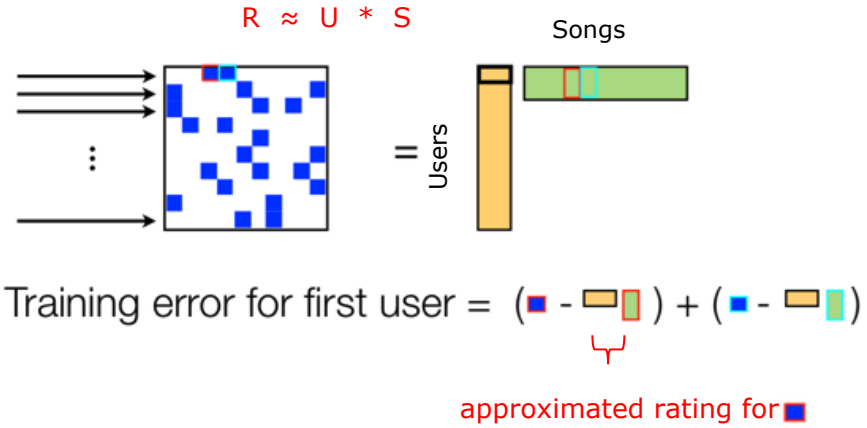


## 3.1 Data Science: Recommendations

# b) Matrix factorization: ALS in Spark

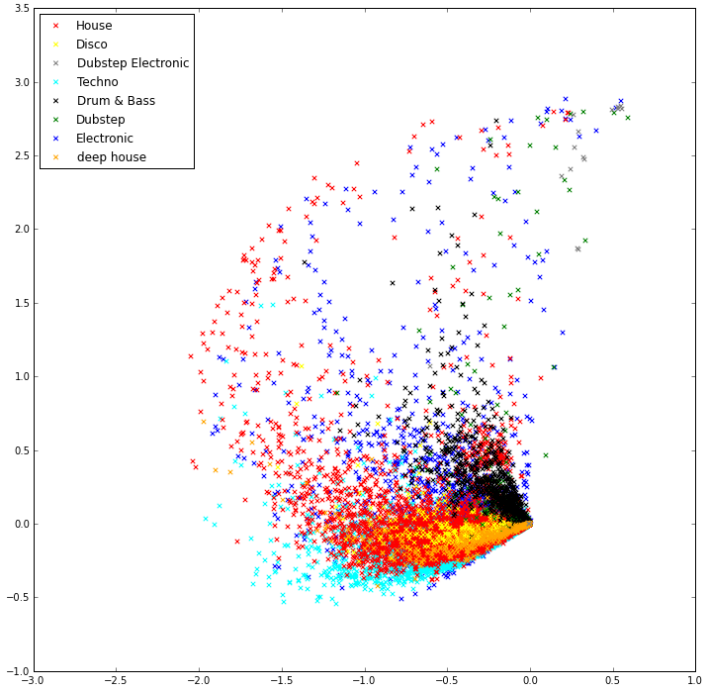
**Alternating least squares:** $\sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y)^2}{n}}$.

Parallel optimization



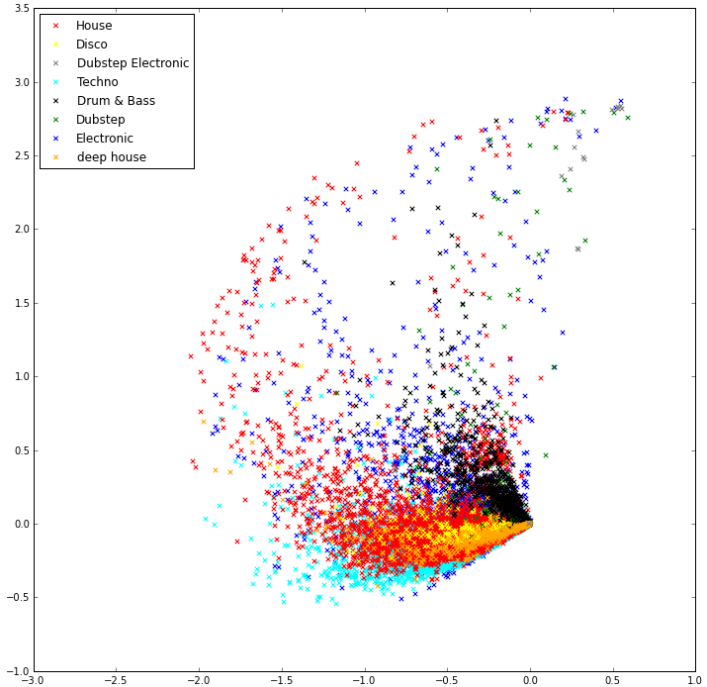
## 3.1 Data Science: Recommendations

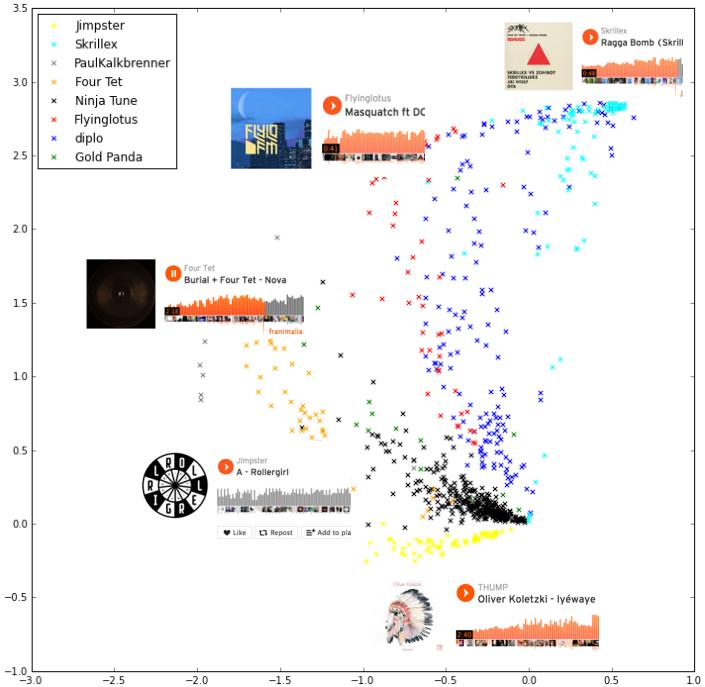
# b) Matrix factorization results: k=2



## 3.1 Data Science: Recommendations

# b) Matrix factorization results: k=2



Skrillex: 0:35
https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/178436255

Electronic: Four Tet: 1:40
https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/38720262

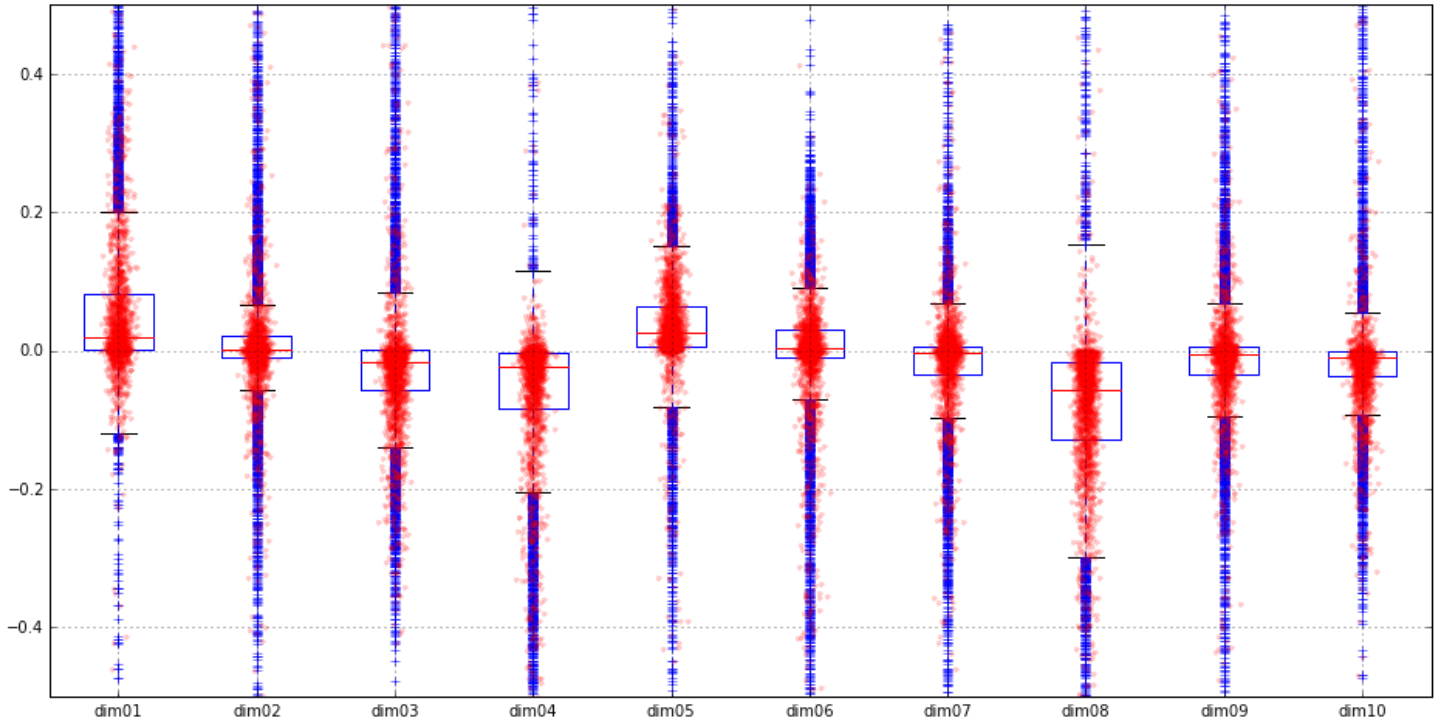
Deep House: Jimpster 1:30
https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/88568930

Techno: Oliver Koletzki 3:11
https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/198557427

# backup links

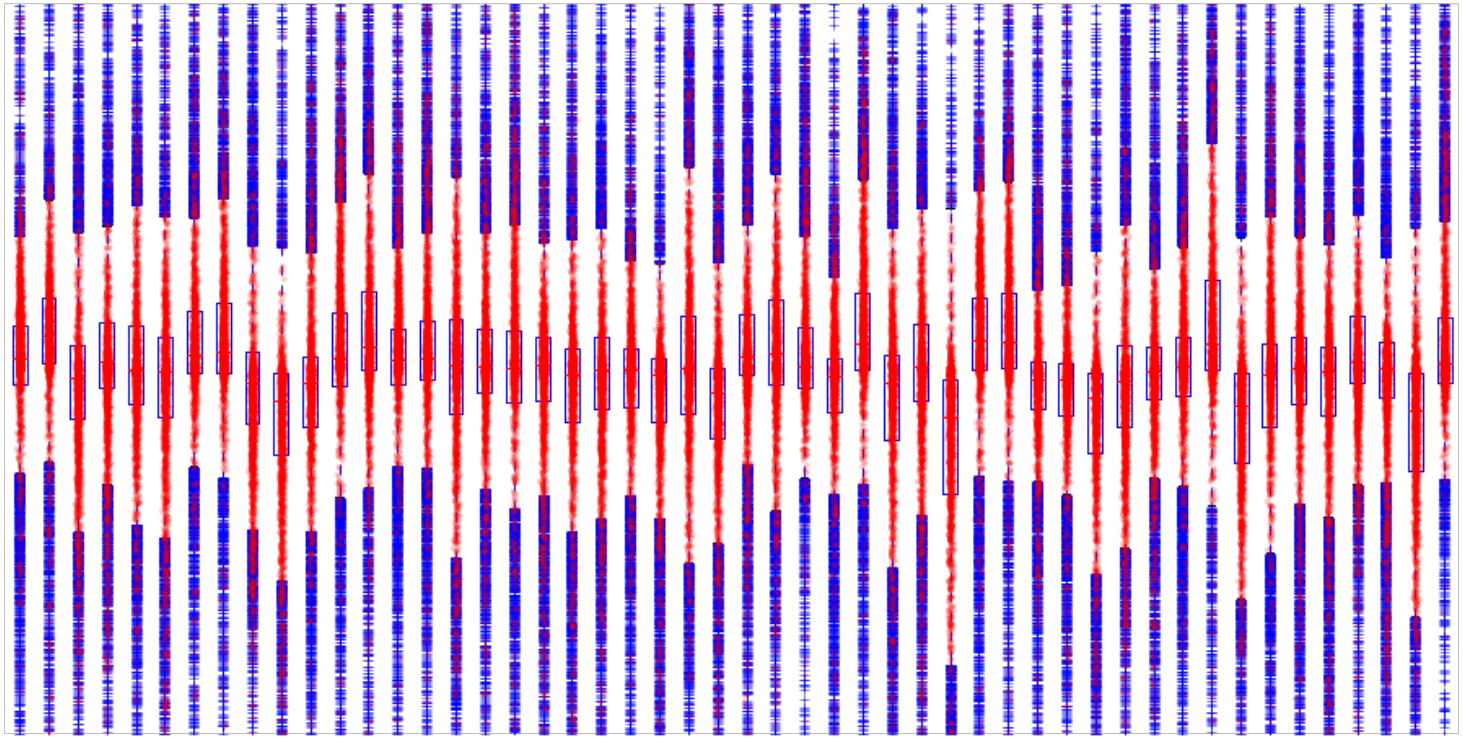
## 3.1 Data Science: Recommendations

# b) Matrix factorization results: k=10



## 3.1 Data Science: Recommendations

# b) Matrix factorization results: k=50



## 3.1 Data Science: Recommendations

# c) Content based/ hybrid approach

- Same artist
- Same genre
- Related artists
- Audio similarity

# 3.1 Data Science: Recommendations:
# Audio similarity

**Danceability:** Describes how suitable a track is for dancing using a number of musical elements: tempo, rhythm stability, beat strength, and overall regularity.

**Energy:** Represents a perceptual measure of intensity and powerful activity released throughout the track. Typical energetic tracks feel fast, loud, and noisy.

**Acousticness:** Represents the likelihood a recording was created by solely acoustic means such as voice and acoustic instruments as opposed to electronically such as with synthesized, amplified, or effected instruments.

**Tempo:** the overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.


**Instrumentalness:** Represents a measure of how likely a song is to be all instrumental with no spoken words

**Valence:** Describes the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g., happy, cheerful, euphoric)

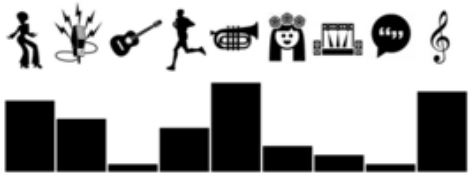
**Liveness:** Detects the presence of an audience in the recording.

**Speechiness:** Speechiness indicates the likelihood the track contains spoken words. Tracks with singing have low speechiness AND low instrumentalness, whereas rap tracks could have low instrumentalness and MEDIUM speechiness.

**Key:** the estimated overall key of a track. The key identifies the tonic triad, the chord, major or minor, which represents the final point of rest of a piece.

## 3.1 Data Science: Recommendations:

# Audio similarity



Nina Kraviz:
Jessie Ware - Keep On Lying

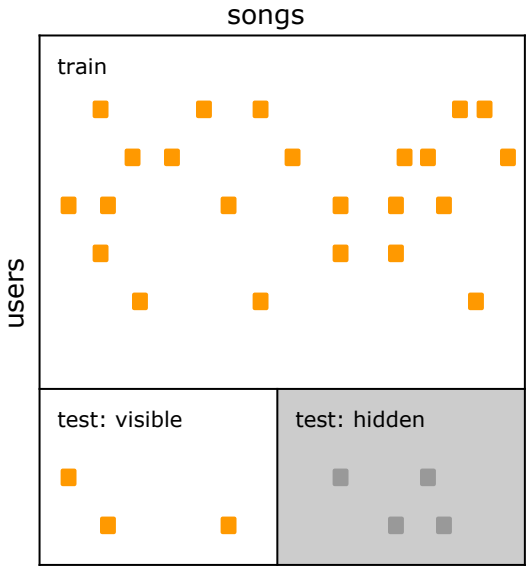

Martin landsky:
City Nord (Martin Landsky)

# Agenda

1. Motivation
2. Approach & Data Engineering
3. Data Science:
    - Generating Recommendations
    - Measuring success
4. Launching the radio service
5. Demo

## 3.2 Data Science: Measuring success

# Formulating the challenge




**Predict which songs a user will like:**

Given:
1) all favourites for 10M users
2) 50% of the favourites for 100K users

What is the task?
Predict the hidden 50%

## 3.2 Data Science: Measuring success

# Formulating the challenge

| userid<br>5506106 | favourite tracks: | predicted tracks: | prediction score: |
|---|---|---|---|
| visible | a) Modeselektor - Siriusmo “Itchy” | - | - |
| visible | b) Four Tet - Lion (Jamie xx remix) | - | - |
| hidden | c) Jimpster - Hijackin' Berlin | e) Bonobo- Essential Mix | 0.8 |
| hidden | d) Four Tet - Final Plastic People | c) Jimpster - Hijackin' Berlin | 0.6 |
| | | d) Four Tet - Final Plastic People | 0.1 |

## 3.2 Data Science: Measuring success

# Formulating the challenge

| userid 5506106 | favourite tracks: | predicted tracks: | prediction score: |
|---|---|---|---|
| visible | a) Modeselektor - Siriusmo "Itchy" | - | - |
| visible | b) Four Tet - Lion (Jamie xx remix) | - | - |
| hidden | c) Jimpster - Hijackin' Berlin | e) Bonobo- Essential Mix | 0.8 |
| hidden | d) Four Tet - Final Plastic People | c) Jimpster - Hijackin' Berlin | 0.6 |
| | | d) Four Tet - Final Plastic People | 0.1 |

$$\text{MSE} = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y)^2}{n}}.$$

$$\text{AUC} = \text{sum}\left(\text{score}([c,d]) > \text{score}([r1,r2])\right)/2$$

AUC may be viewed as the probability that a random positive item scores higher than a random negative one.

$$\text{MAP}(n=3) = (0 + 1/2 + 2/3) / 2 = 0.58333$$

Mean average precision for a set of users is the mean of the average proportion of correct recommendations for each user

## 3.2 Data Science: Measuring success

# Formulating the challenge

| userid 5506106 | favourite tracks: | predicted tracks: | prediction score: |
|---|---|---|---|
| visible | a) Modeselektor - Siriusmo "Itchy" | - | - |
| visible | b) Four Tet - Lion (Jamie xx remix) | - | - |
| hidden | c) Jimpster - Hijackin' Berlin | e) Bonobo- Essential Mix | 0.8 |
| hidden | d) Four Tet - Final Plastic People | c) Jimpster - Hijackin' Berlin | 0.6 |
| | | d) Four Tet - Final Plastic People | 0.1 |

$$MSE = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y)^2}{n}}.$$

AUC = sum ( score([c,d]) > score([r1,r2]) ) /2

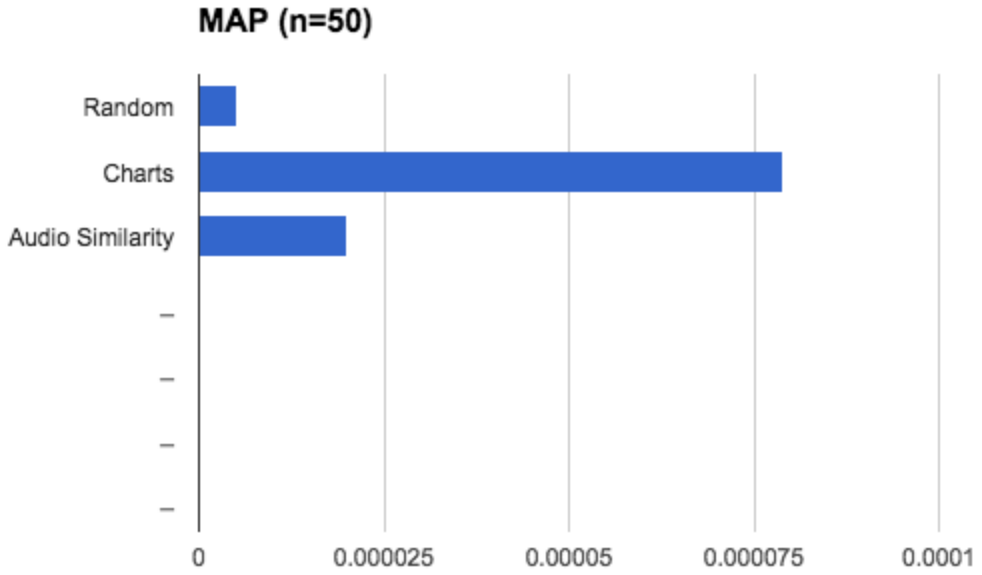
AUC may be viewed as the probability that a random positive item scores higher than a random negative one.

MAP(n=3) = (0 + 1/2 +2/3) / 2 = 0.58333

Mean average precision for a set of users is the mean of the average proportion of correct recommendations for each user

## 3.2 Data Science: Measuring success

# Performance overview




*MAP(50): average proportion of correct recommendations*

**Charts:**


**#1**

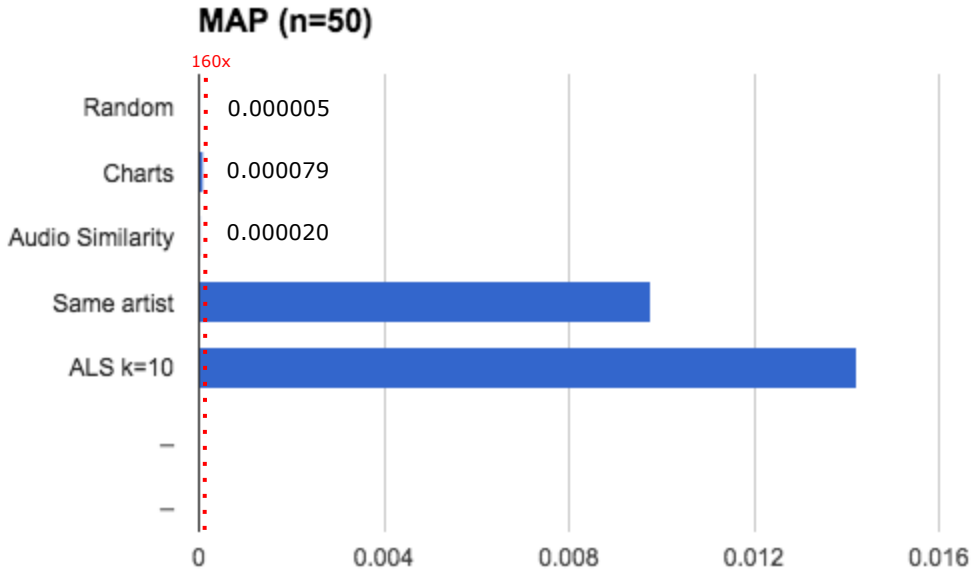

**#2**


**#3**

**...** **#200.000**

## 3.2 Data Science: Measuring success

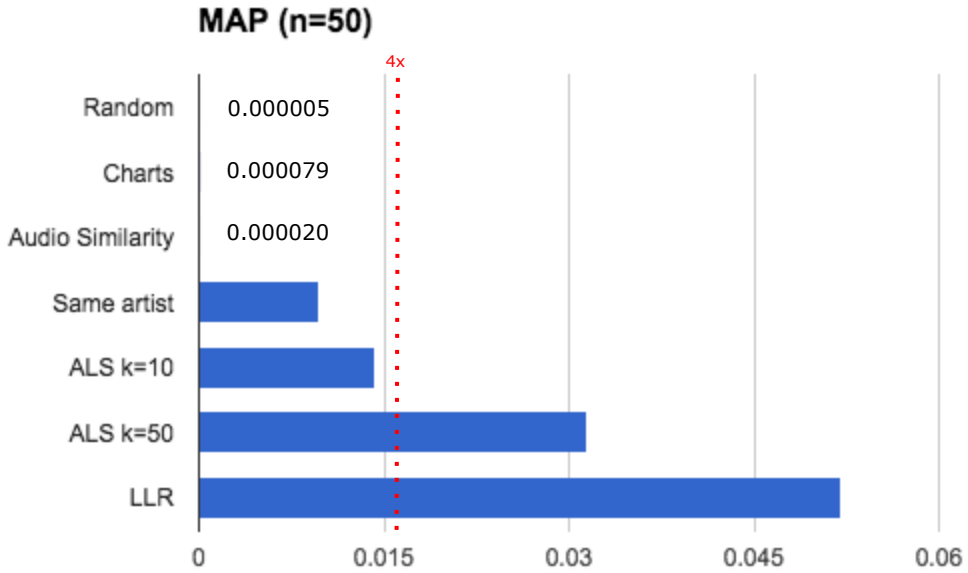
# Performance overview




*MAP(50): average proportion of correct recommendations*

## 3.2 Data Science: Measuring success

# Performance overview

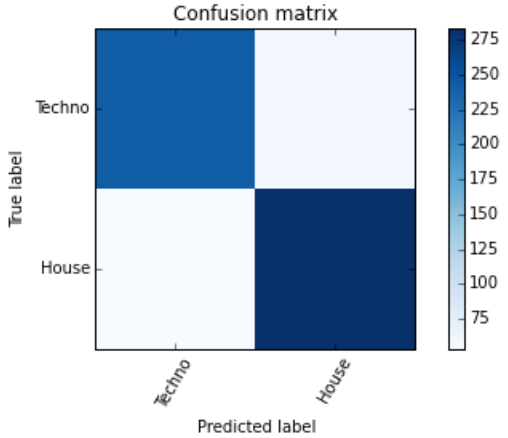



*MAP(50): average proportion of correct recommendations*

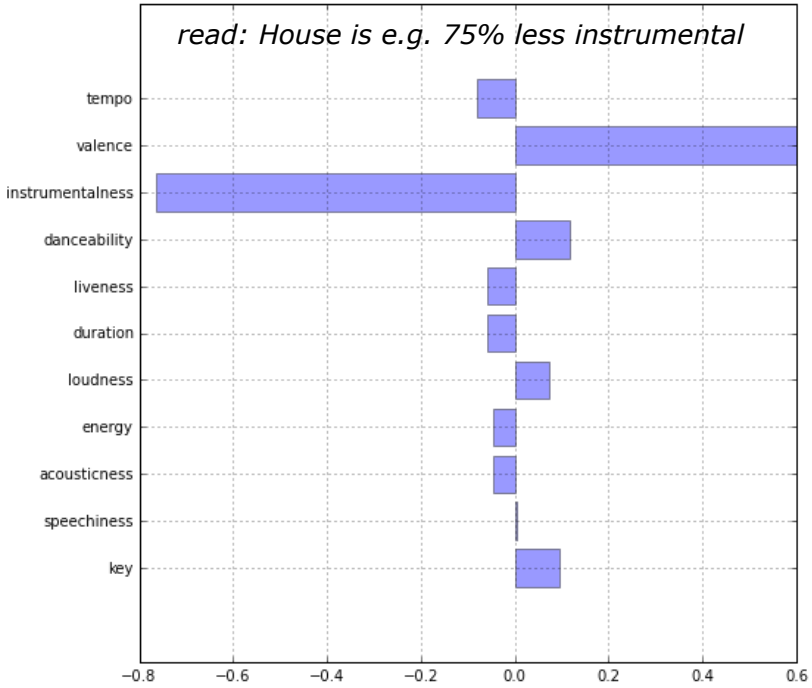
## 3.2 Data Science: Recommendations

# c) House vs. Techno audio classifier

| genre | precision | recall | f1-score |
|---|---|---|---|
| techno | 0.82 | 0.80 | 0.81 |
| house | **0.83** | **0.84** | 0.83 |



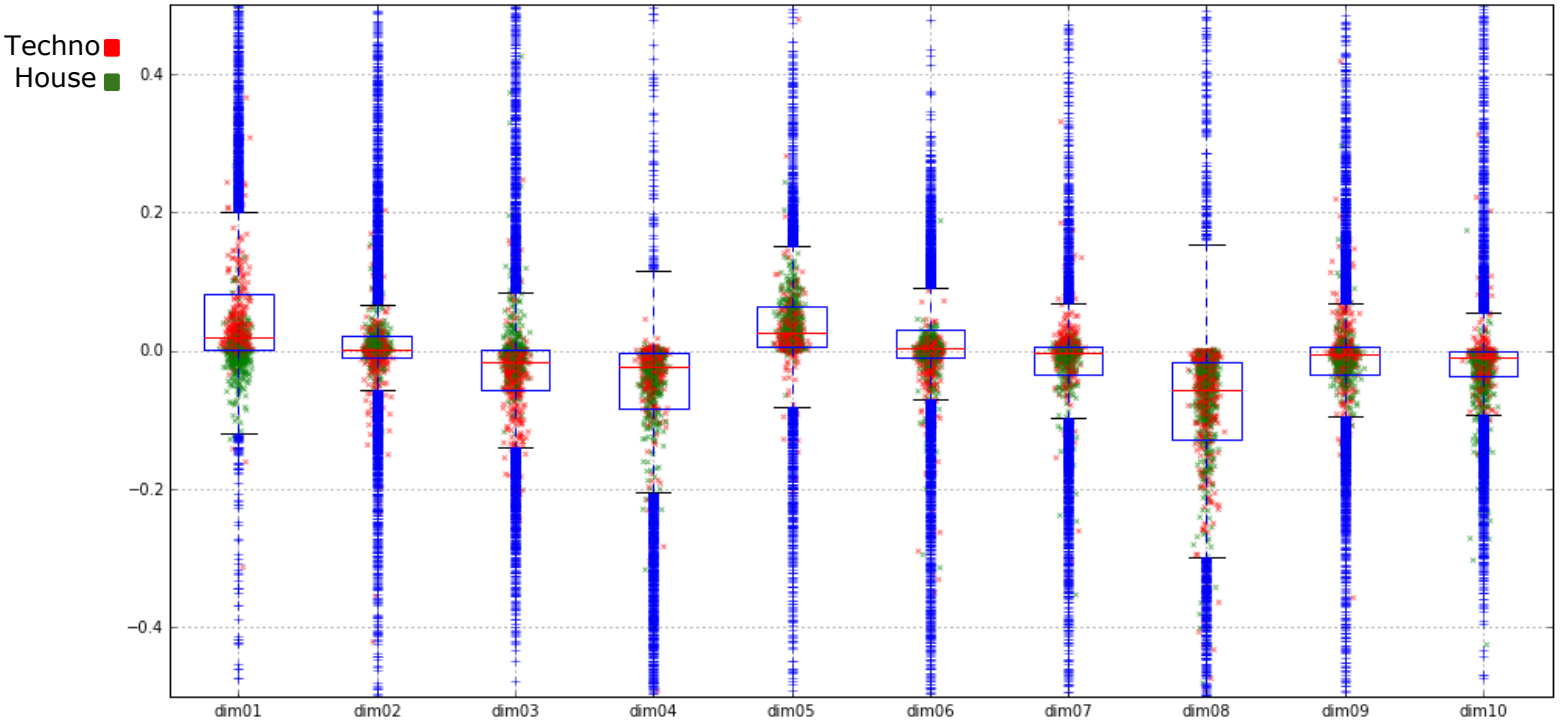

**Random Forest classifier based on the echonest features**




| weights: |
|---|
| 0.328416 |
| 0.188782 |
| 0.112463 |
| 0.086643 |
| 0.066062 |
| 0.050931 |
| 0.041969 |
| 0.040186 |
| 0.034053 |
| 0.033277 |
| 0.013488 |

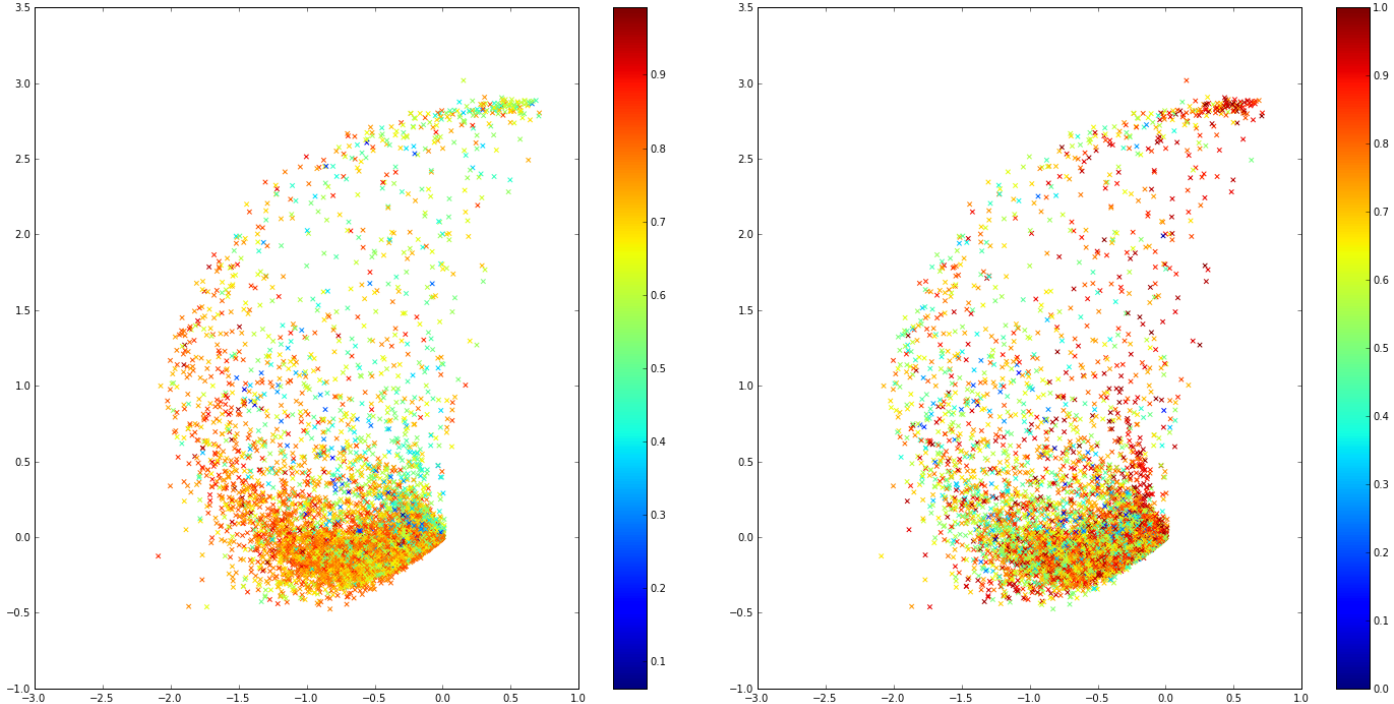
## 3.2 Data Science: Recommendations

# c) Techno vs House in 10D



## 3.2 Data Science: Recommendations

# c) Danceability vs Energy

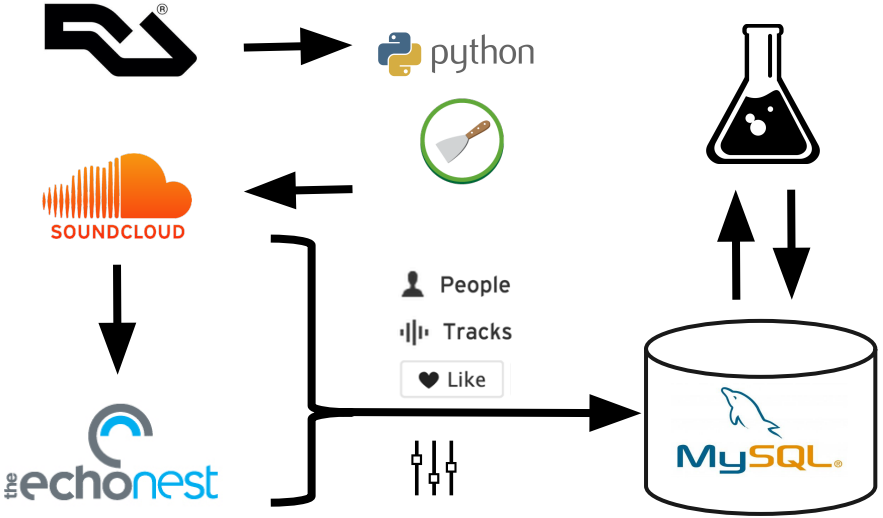


# Agenda

1. Motivation
2. Approach & Data Engineering
3. Data Science:
    - Generating Recommendations
    - Measuring success
4. Launching the radio service
5. Demo

## 4. Launching the radio service

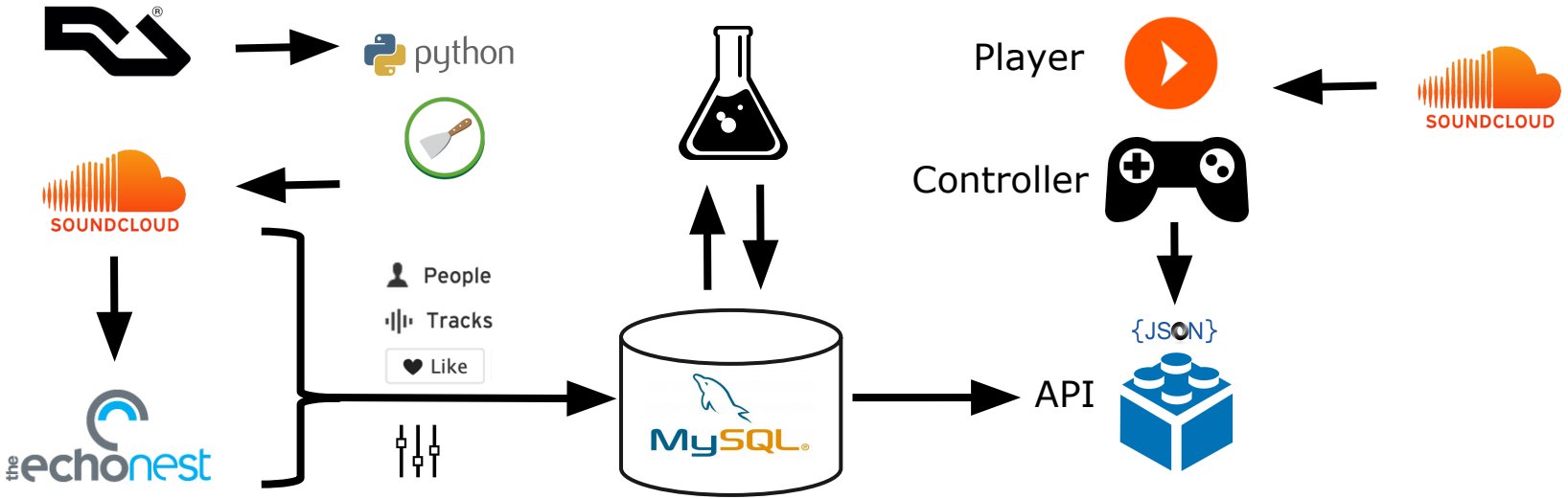
# Behind the scenes:




amazon web services™

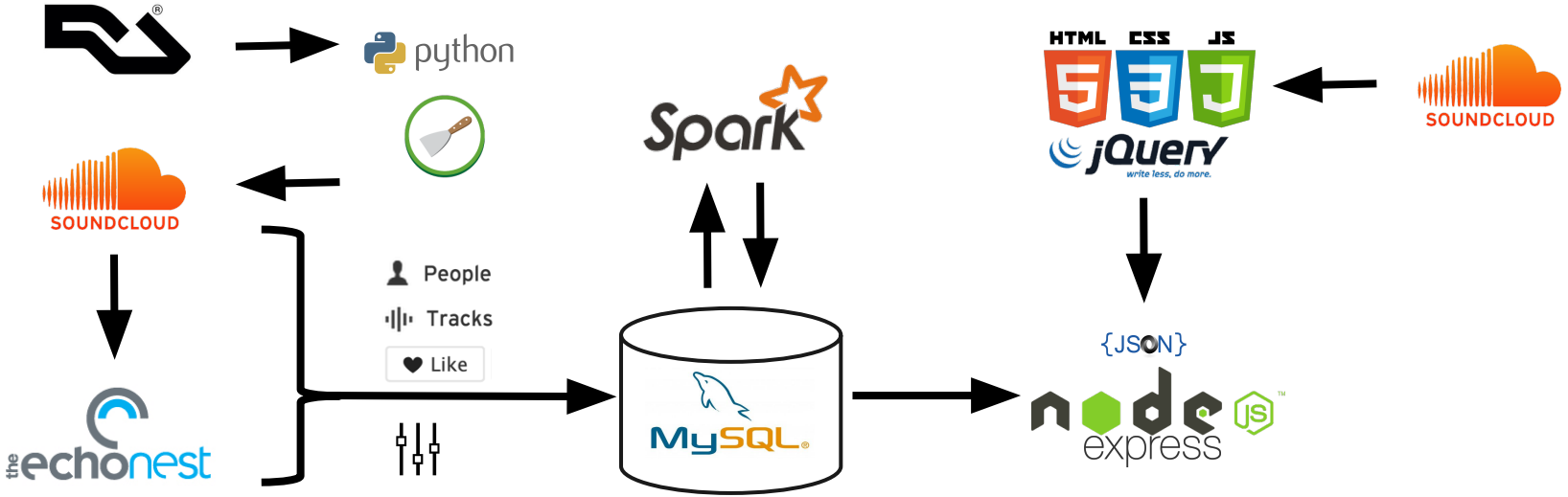
## 4. Launching the radio service

# Behind the scenes:



## 4. Launching the radio service

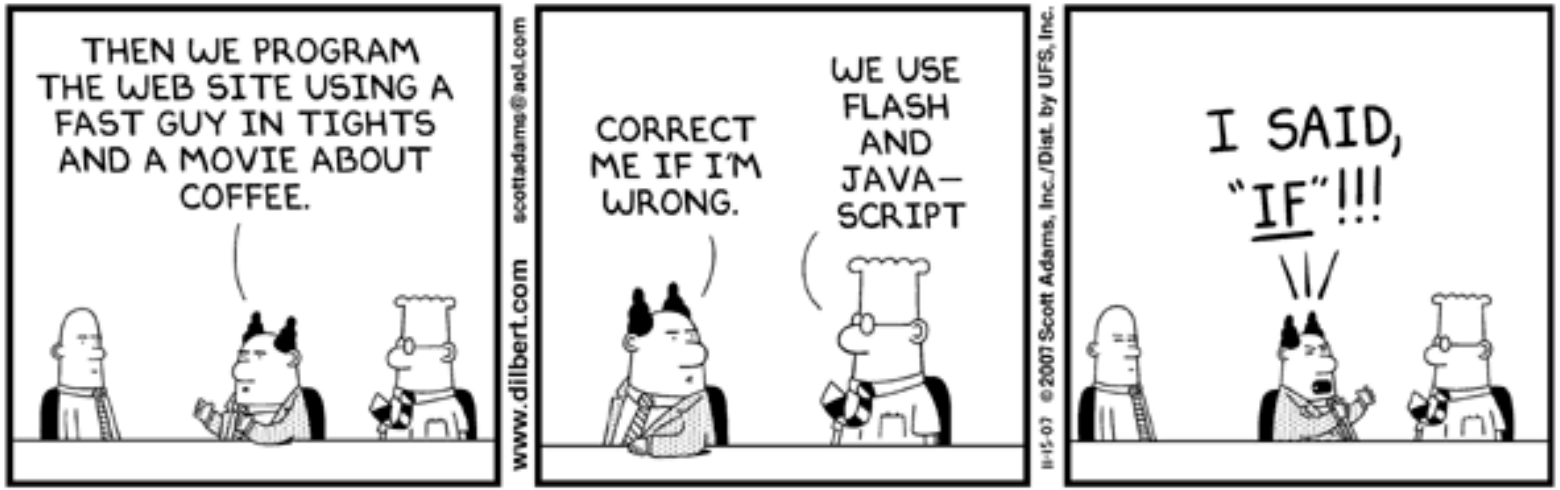
# Behind the scenes:




amazon web services™

heroku

# Demo



# Project:

## Personal local electronic music radio

Scope :

- **Local**: Berlin
- **Radio**: Autoplay of generated playlists
- **Personal**: Responsive to feedback 
- **Music**: Focus on electronic music

Motivation :

- Scoop into Berghain

# Demo

# Johannes Braun


**Final project, 30/04/15**

braun.johannes@gmail.com
linkedin.com/in/johannesbraun1
johannesbraun.github.io
github.com/johannesbraun


```python
def apk(actual, predicted, k=10):

    if len(predicted)>k:
        predicted = predicted[:k]

    score = 0.0
    num_hits = 0.0

    for i,p in enumerate(predicted):
        if p in actual and p not in predicted[:i]:
            num_hits += 1.0
            score += num_hits / (i+1.0)

    if not actual:
        return 1.0

    return score / min(len(actual), k)

def mapk(actual, predicted, k=10):
    return np.mean([apk(a,p,k) for a,p in zip(actual, predicted)])
```

```
visible
```

```
[166057101, 190972494, 74927480, 97610023, 181657050, 1
```

```
zip([actual], [predicted[0:10]])
```

```
[([111221516, 162642747, 184927536, 112604679, 14309598
  [117805508,
   177358723,
   111244653,
   108807680,
   142565321,
   102378251,
   168290694,
   142738302,
   145502026,
   112604679])]
```

```
mapk([actual], [predicted],10)
```

```
0.02
```

# Mean Average Precision at rank k=10

Source: https://**github**.com/benhamner/Metrics

## 2. Approach & Data Engineering:

# Extracting Audio Features

**Danceability:** Describes how suitable a track is for dancing using a number of musical elements: tempo, rhythm stability, beat strength, and overall regularity.

**Energy:** Represents a perceptual measure of intensity and powerful activity released throughout the track. Typical energetic tracks feel fast, loud, and noisy.

**Acousticness:** Represents the likelihood a recording was created by solely acoustic means such as voice and acoustic instruments as opposed to electronically such as with synthesized, amplified, or effected instruments.

**Tempo:** the overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.

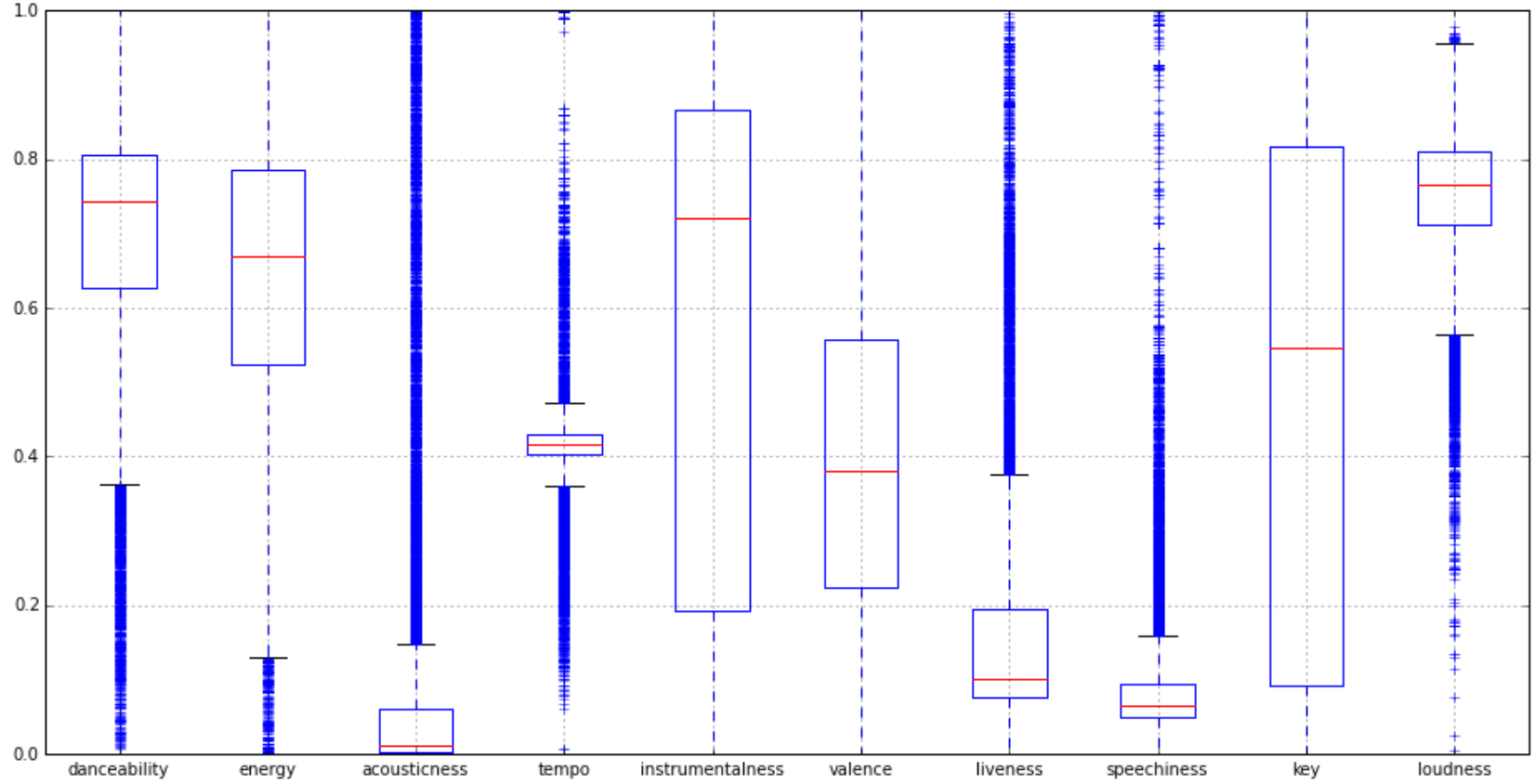

**Instrumentalness:** Represents a measure of how likely a song is to be all instrumental with no spoken words

**Valence:** Describes the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g., happy, cheerful, euphoric)

**Liveness:** Detects the presence of an audience in the recording.

**Speechiness:** Speechiness indicates the likelihood the track contains spoken words. Tracks with singing have low speechiness AND low instrumentalness, whereas rap tracks could have low instrumentalness and MEDIUM speechiness.

**Key:** the estimated overall key of a track. The key identifies the tonic triad, the chord, major or minor, which represents the final point of rest of a piece.




# Distribution of Audio Features

## 3.4 Data Science:
# Extensions

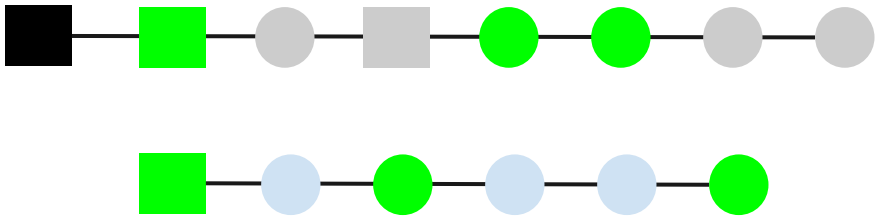
1) Music math: jimpster - kalkbrenner = ??
2) Upload music track so it can be seen as combination of djs
3) Compare Spark ALS with
    - Vowpal Wabbit
    - MS Azure ML: Matchbox
4) Better hybrid approaches:
    - Rank aggregation
    - Mine comments
    - Mine echonest data in more depth
    - Integrate external knowledge of DJ labels etc.

## 3.3 Data Science: Personalization

# Update playlists with user feedback

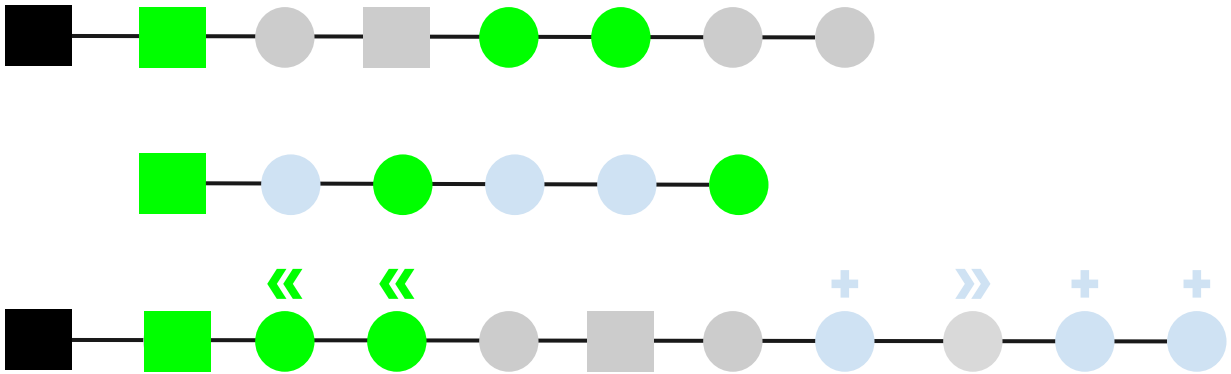
## 3.3 Data Science: Personalization

# Update playlists with user feedback



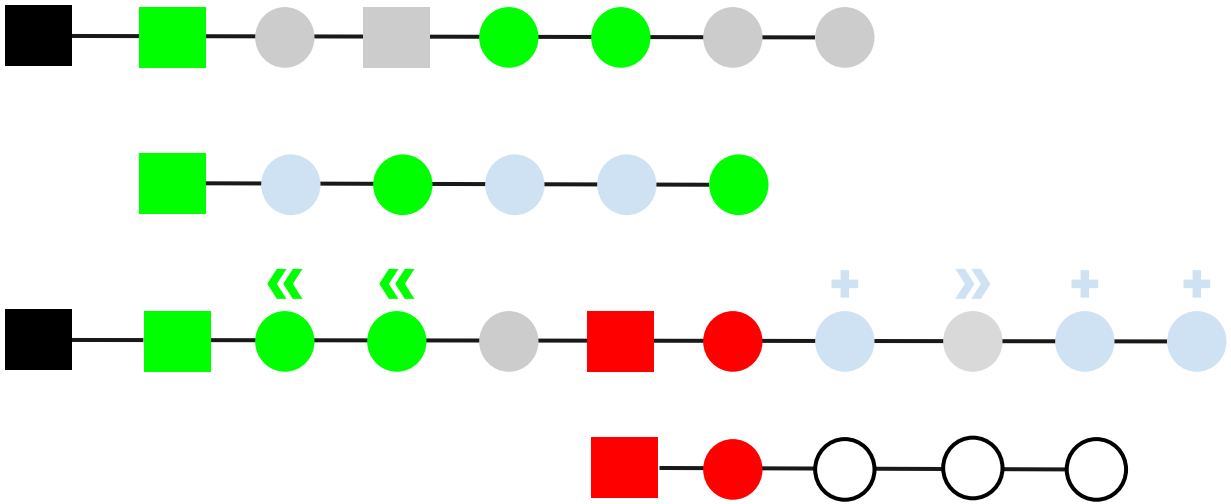
## 3.3 Data Science: Personalization

# Update playlists with user feedback



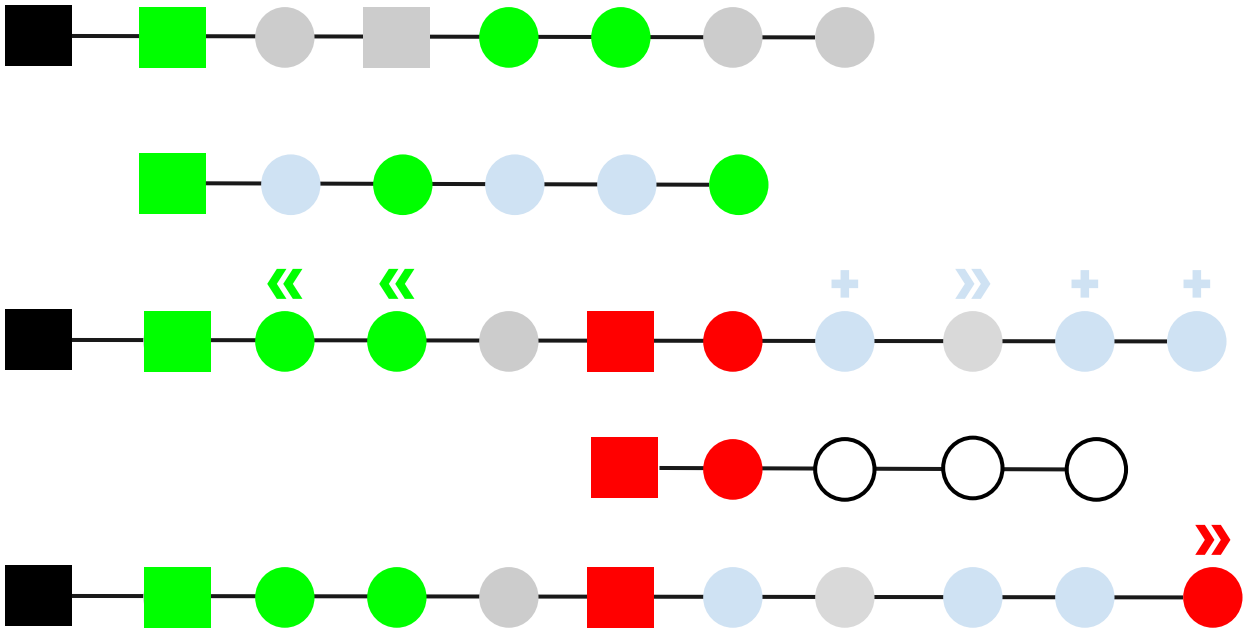
## 3.3 Data Science: Personalization

# Update playlists with user feedback



## 3.3 Data Science: Personalization

# Update playlists with user feedback

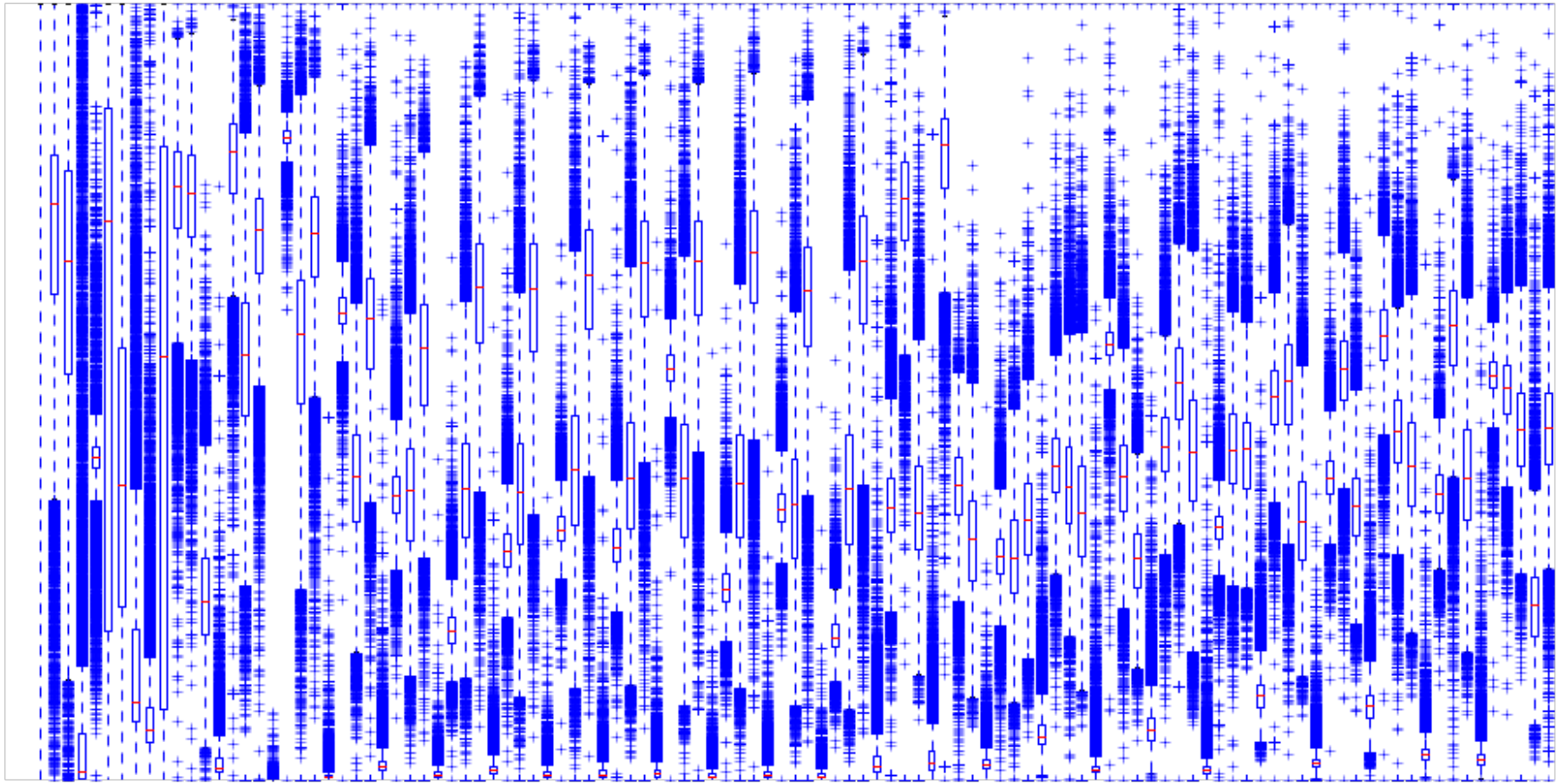


# Update playlists with user feedback

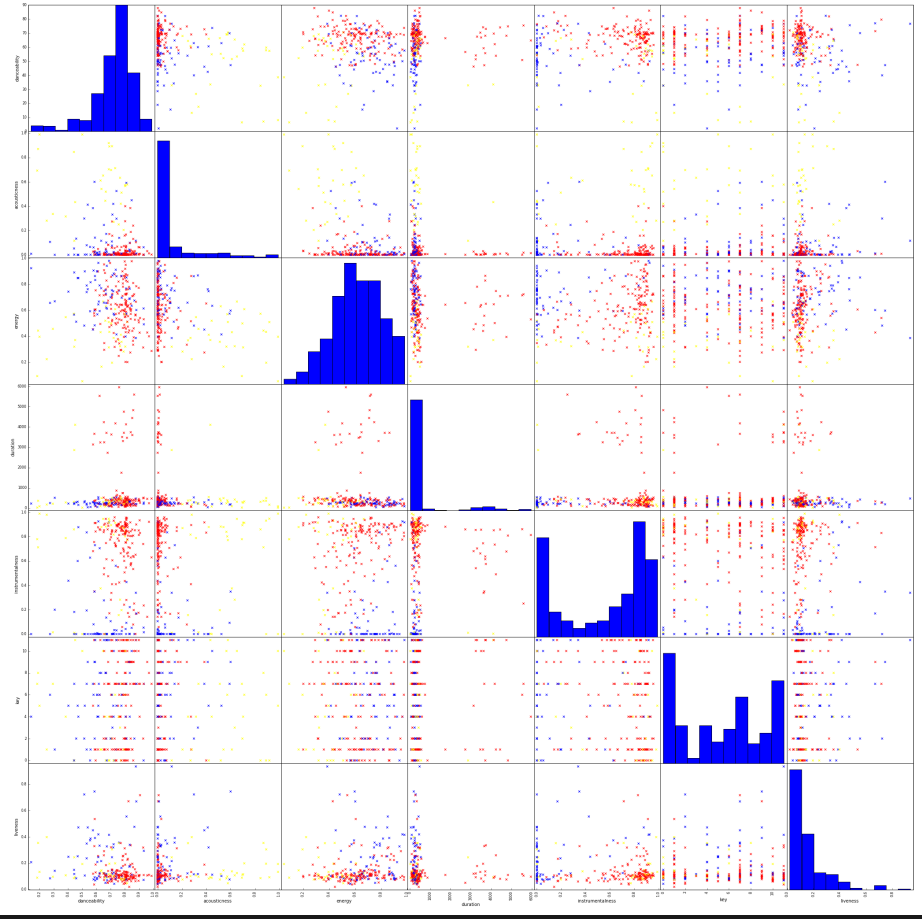
**Naive update rule:**

time decayed aggregated score of positive and negative recommendations

- LLR: neighborhood scores
- ALS: cosine similarities or aggregate user vector



Distribution of Audio Features incl. Timbre and Pitch features




# Attempt to cluster songs by The Echo Nest Features

```
llr = function(k) {
  2 * sum(k) * (H(k) - H(rowSums(k)) - H(colSums(k)))
}

H = function(x) {
  N = sum(x) ;
  return (sum(x/N * log(x/N + (x==0))))
}
```

```
k = matrix(c(2,10,12,400),nrow=2)
k
```

```
      [,1]      [,2]      colSums
[1,]  k11: 2    k12: 12   14
[2,]  k22: 10   k22: 400  410
rowSums    12        412  424

llr(k)
> 3.704182
```

```
      [,1]      [,2]      colSums
[1,]  k11: 2    k12: 12   14
[2,]  k22: 1    k22: 400  401
rowSums    3         412  415

llr(k)
> 10.08757
```

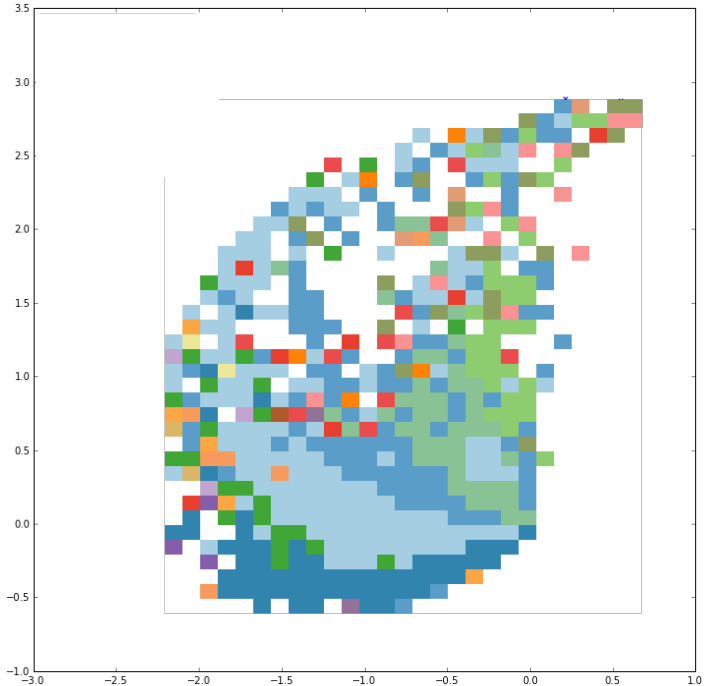
| | Song A | Everything but A |
|---|---|---|
| Song B | A and B together (k_11) | B, but not A (k_12) |
| Everything but B | A without B (k_21) | Neither A nor B (k_22) |

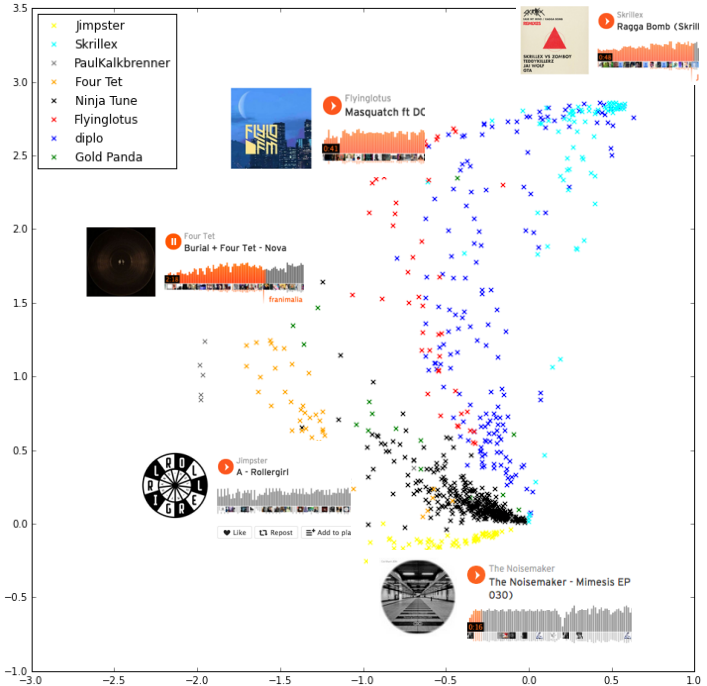
# Neighborhood approach: LLR

http://tdunning.blogspot.de/2008/03/surprise-and-coincidence.html

## 3.1 Data Science: Recommendations

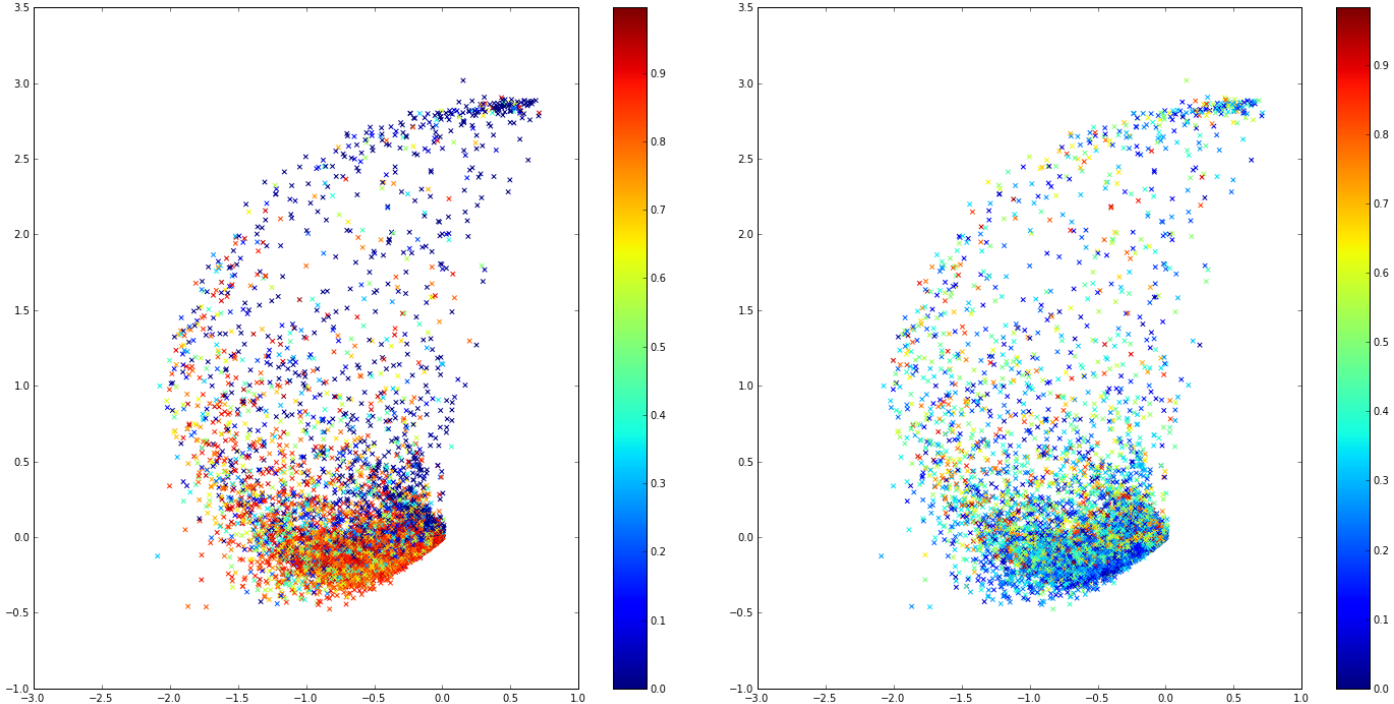
# b) Matrix factorization results: k=2



## 3.1 Data Science: Recommendations

# c) Instrumentalness vs Valence



## 5.2 Truncated Mean Average Precision

Conformingly to the challenge, we used the truncated mAP (mean average precision) as the evaluation metric [10]. Let $y$ denote a ranking over items, where $y(p) = i$ means that item $i$ is ranked at position $p$. The mAP metric emphasizes the top recommendations. For any $k \leq N$, the *precision at k* ($\pi_k$) is defined as the proportion of correct recommendations within the top-$k$ of the predicted ranking (assuming the ranking $y$ does not contain the visible songs),

$$\pi_k(u, y) = \frac{1}{k} \sum_{p=1}^{k} r_{uy(p)}$$

For each user the (truncated) average precision is the average precision at each recall point:

$$AP(u, y) = \frac{1}{N_u} \sum_{p=1}^{N} \pi_k(u, y) r_{uy(p)}$$

where $N_u$ is the smaller between $N$ and the number of user $u$'s positively associated songs. Finally, the average of $AP(u, y_u)$'s over all users gives the mean average precision (mAP).

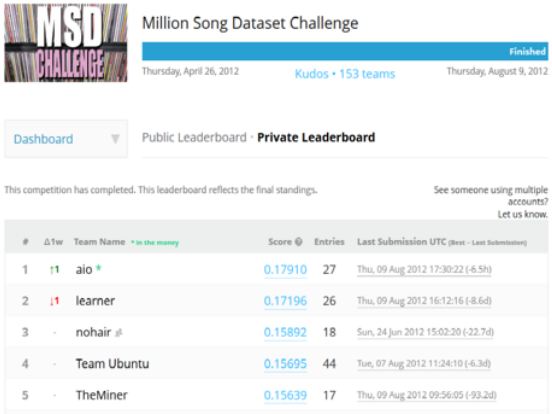



Figure 2: Screenshot of the final MSD challenge leaderboard.

**Fabio Aiolli, Winner of the Million Song Dataset challenge | Kaggle**

taken from: Efficient Top-N Recommendation for Very Large Scale Binary Rated Datasets, F. Aiolli, 2013

## 7. Area under an ROC curve (AUC)

An ROC curve is a two-dimensional depiction of classifier performance. To compare classifiers we may want to reduce ROC performance to a single scalar value representing expected performance. A common method is to calculate the area under the ROC curve, abbreviated **AUC** (Bradley, 1997; Hanley and McNeil, 1982). Since the AUC is a portion of the area of the unit square, its value will always be between 0 and 1.0. However, because random guessing produces the diagonal line between (0,0) and (1,1), which has an area of 0.5, no realistic classifier should have an AUC less than 0.5.

The AUC has an important statistical property: the AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. This is

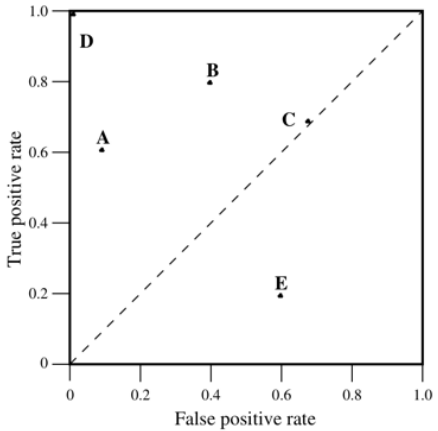



Fig. 2. A basic ROC graph showing five discrete classifiers.

# Casting Recommendations as a Classification Problem

taken from: An introduction to ROC analysis, Pattern Recognition Letters 27, T. Fawcett, 2006

# blasta.me

*Cartoon character "**Blaster**" from the japanese animated series "Transformers: The Headmasters" (He transforms into a Ghettoblaster)*