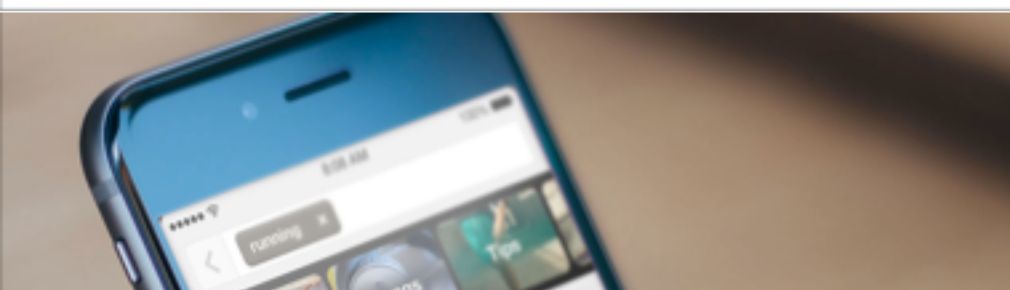
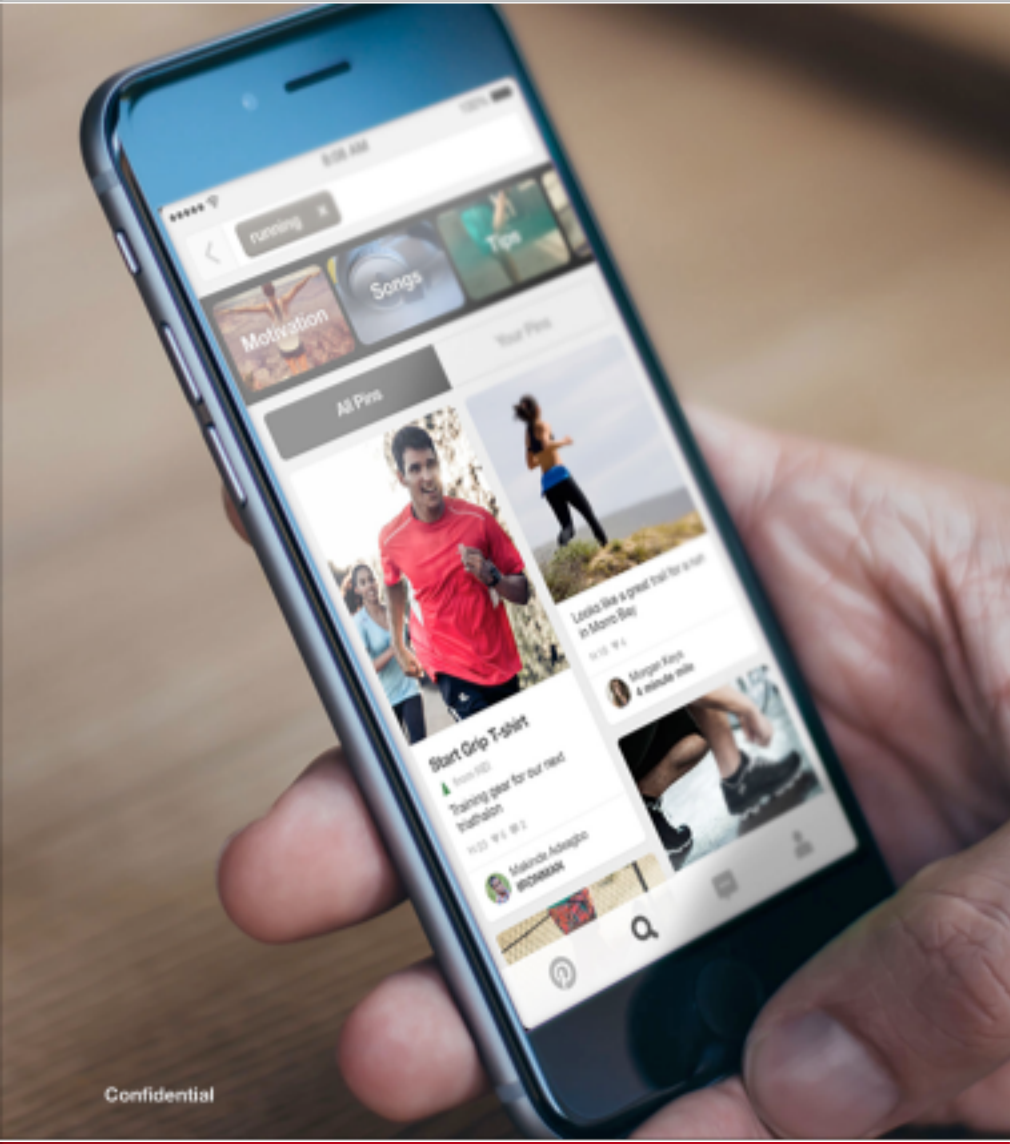
Personalizing the Pinterest Homefeed

Dmitry Chechik

QCon



What is Pinterest?

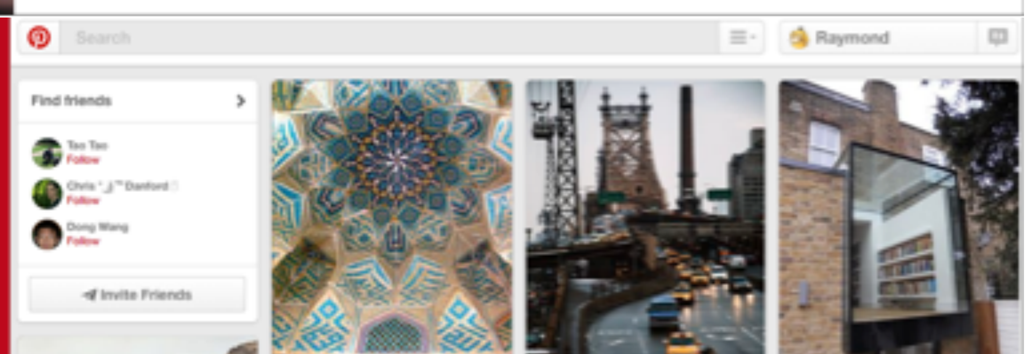


What is Pinterest?

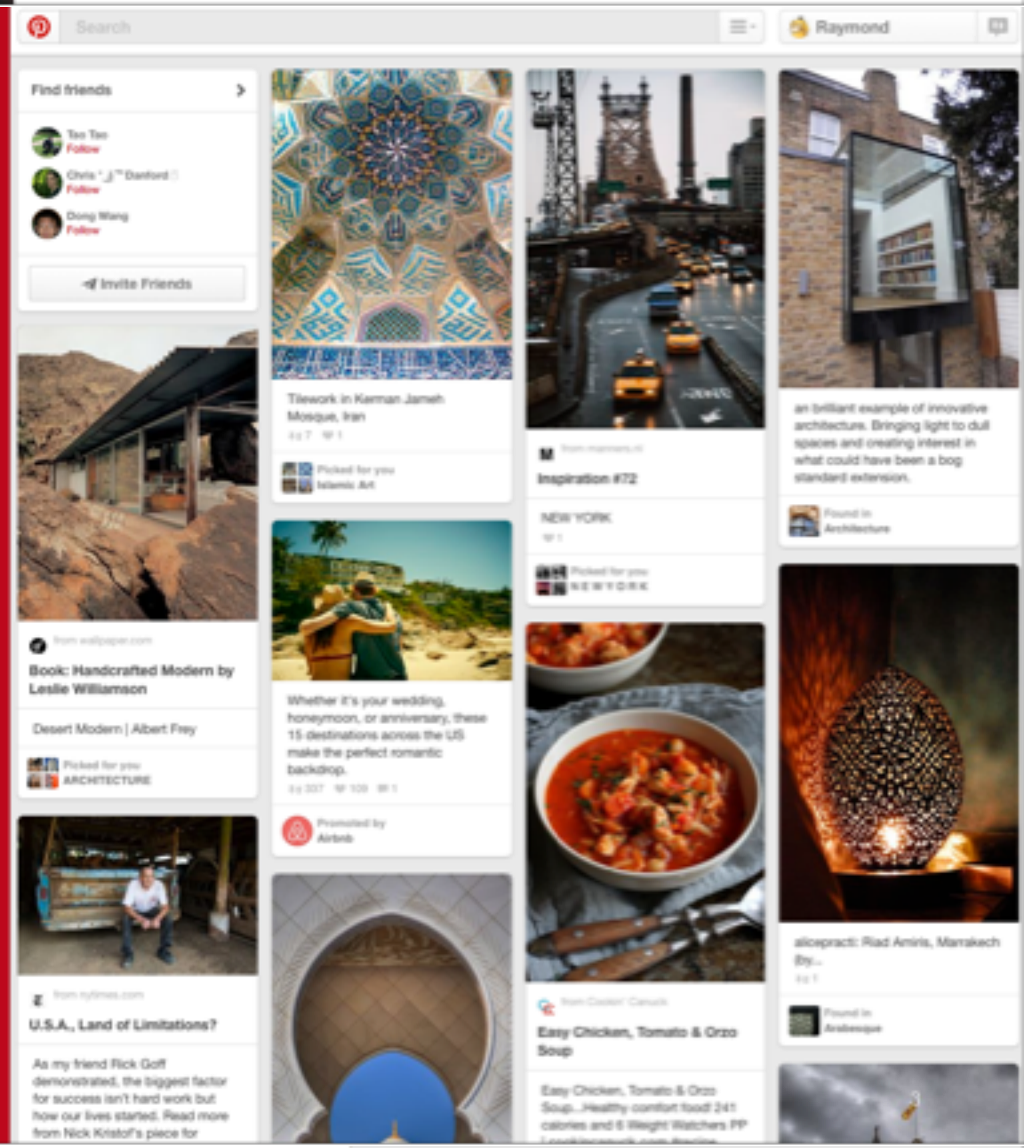
Pinterest is a visual bookmarking tool and discovery engine.

Users pin images and sites they like onto boards. Every pin on Pinterest is added by a human and lives on a board.

Users heavily curate their content.



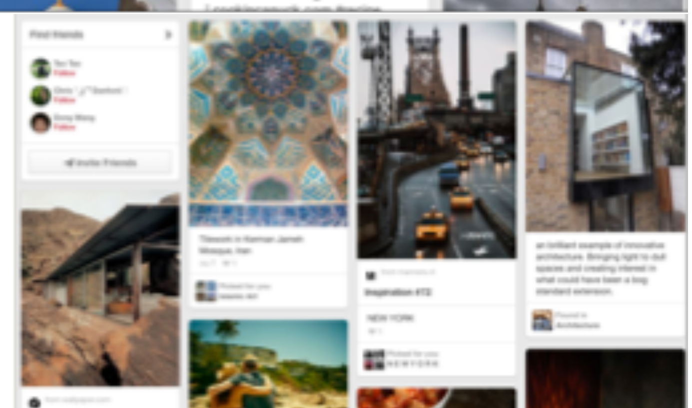
Homefeed



Homefeed

What it is

Diverse, Relevant, Endless **set of pins to a user**



Homefeed

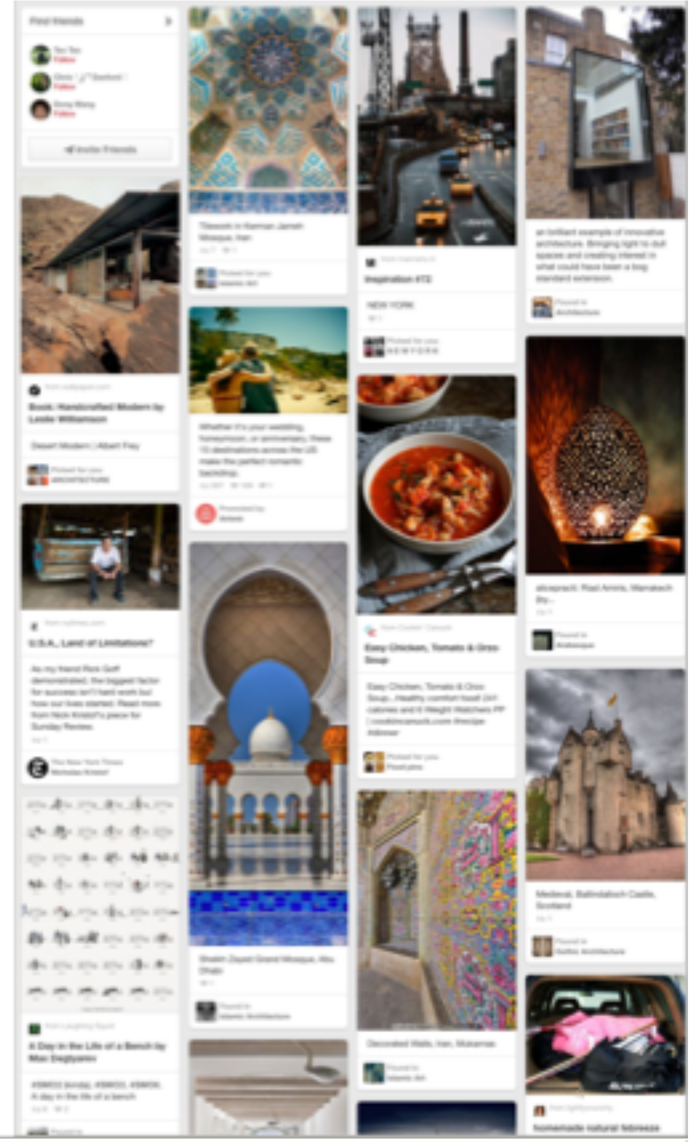
What it is

Diverse, Relevant, Endless **set of pins to a user**

Show pins and content meaningful to a user without a specific query

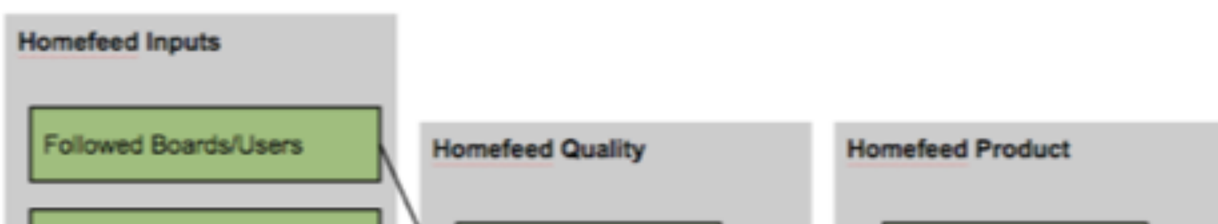
Combines content from:

- Users or boards you follow
- Interests you follow
- Recommendations

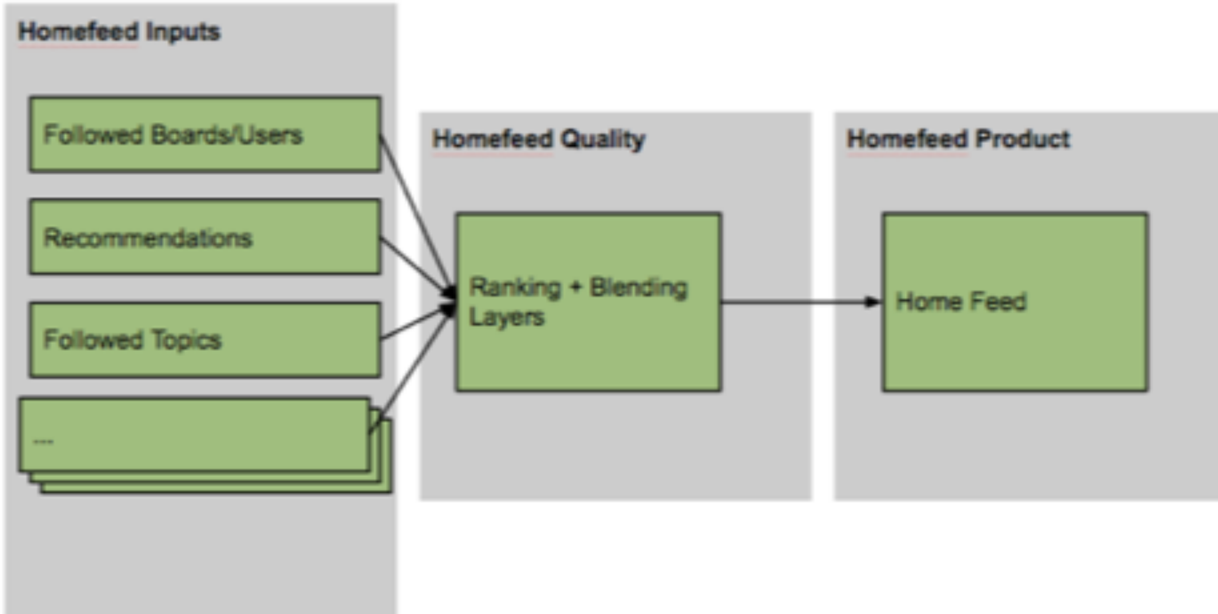


Confidential

Homefeed Logical Architecture



Homefeed Logical Architecture



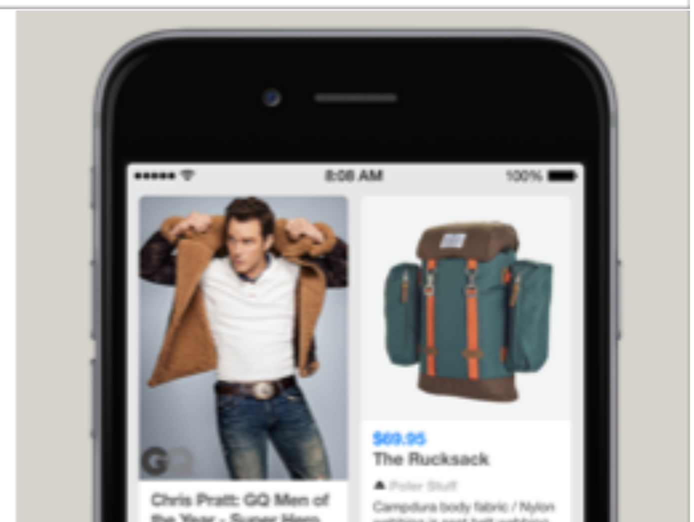
Homefeed

Problems we're trying to solve

Generating candidates

- Find pins that we think you'll like

Scoring and ranking



Homefeed

Problems we're trying to solve

Generating candidates

- Find pins that we think you'll like

Scoring and ranking

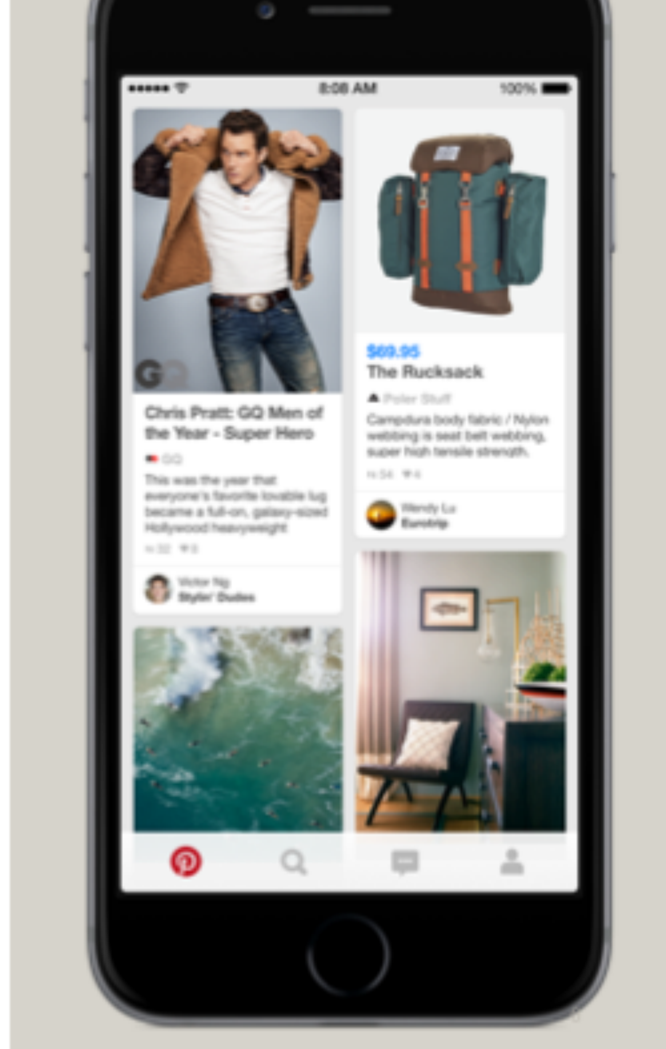
- Picking the best of the best among candidates

Blending of different sources

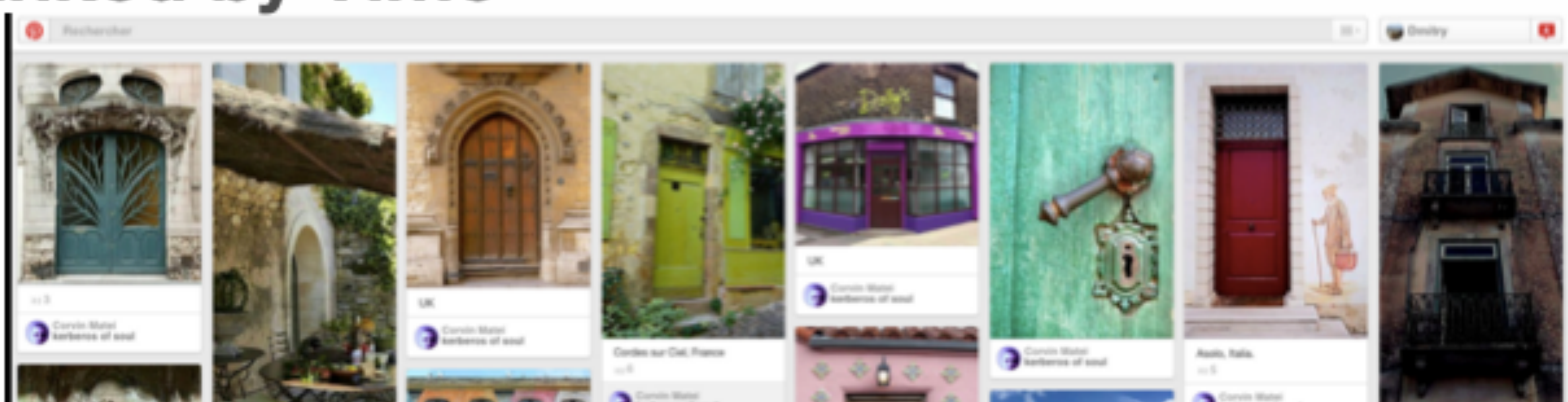
- Followed boards/users/interests, recommendations

Creating final feed

- Doing this for 10s of millions of users multiple times a day

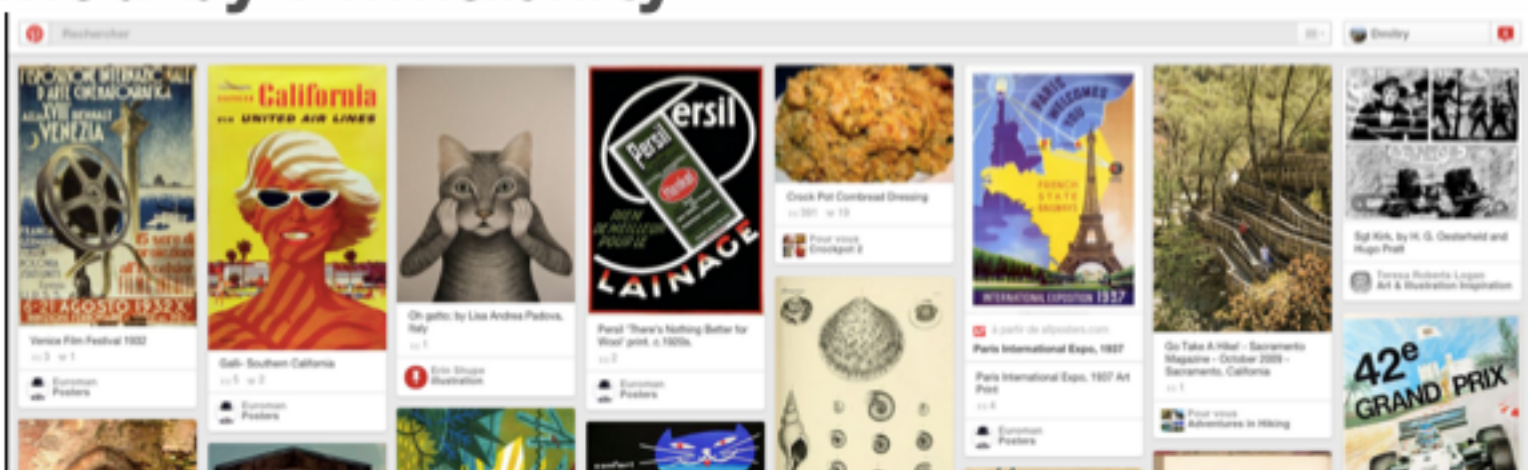


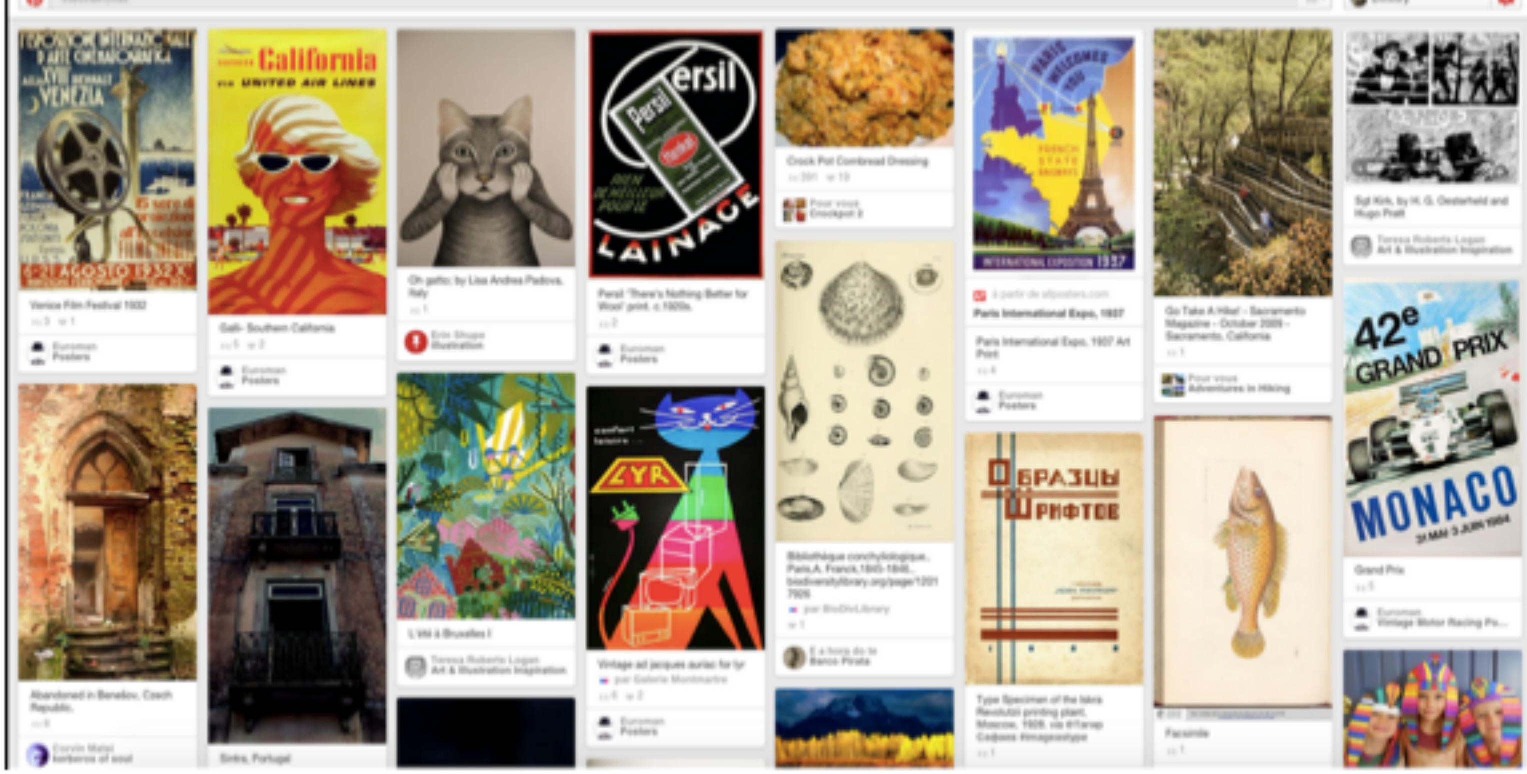
Ranked by Time



Ranked by Time

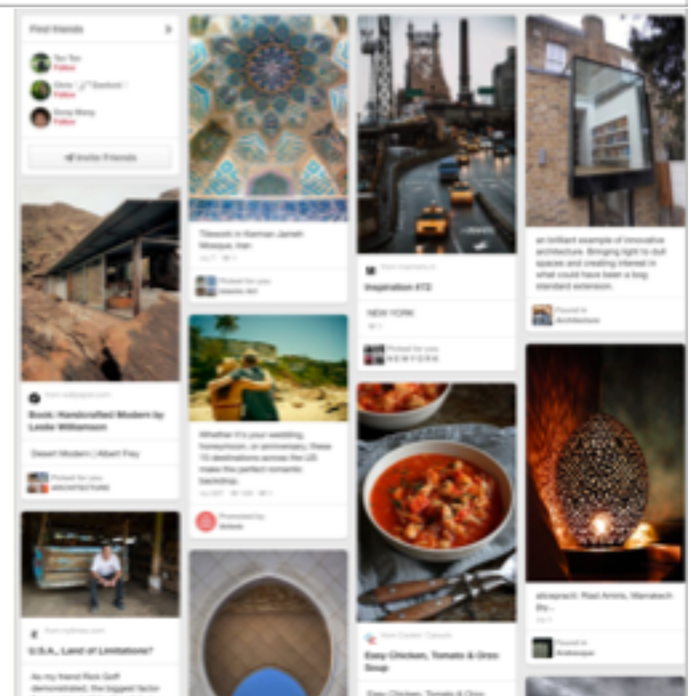
Ranked by Pinnability



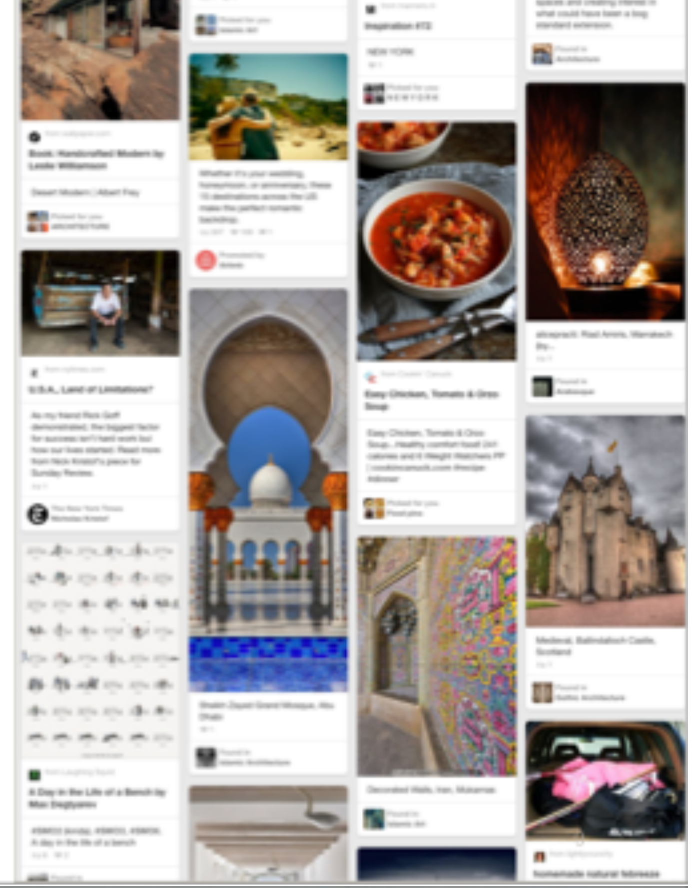


This Talk

- Homefeed Ranking Model Overview
- Evolving a Model in Production
- Problems to Solve
- Pinterest Homefeed's Model Framework

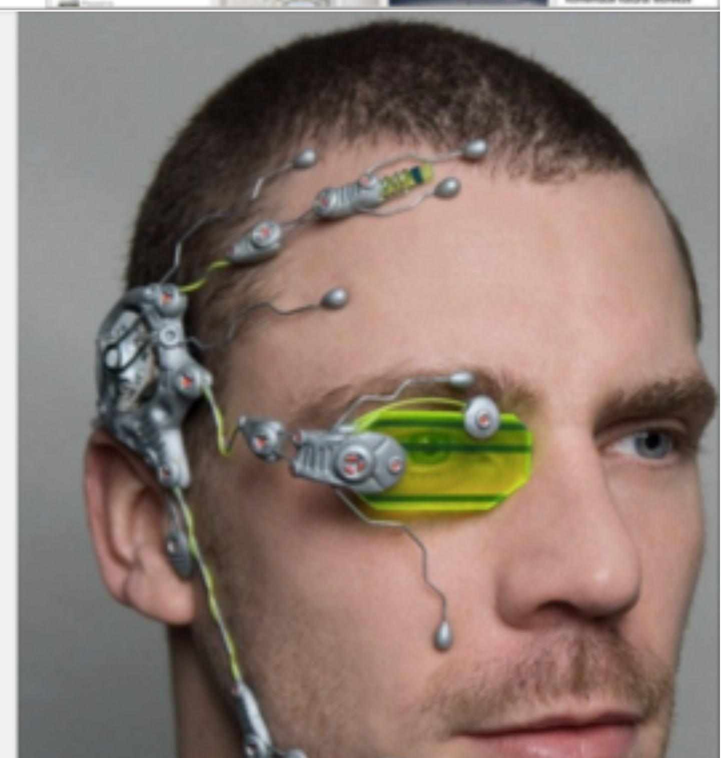


- Homefeed Ranking Model Overview
- Evolving a Model in Production
- Problems to Solve
- Pinterest Homefeed's Model Framework



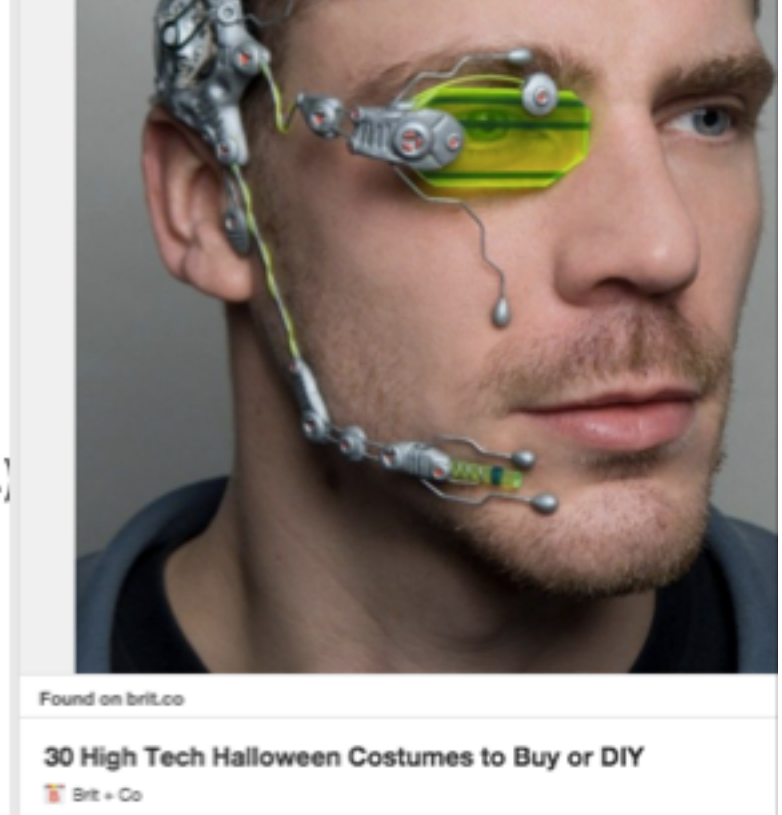
Confidential

What's in a Pin



- User-generated details
- URL: http://www.brit.co/…
- Image features
- User-curated pin-board graph
- User-curated annotations

- User-generated details
- URL: http://www.brit.co/…
- Image features
- User-curated pin-board graph
- User-curated annotations
- On-site performance (click actions, impressions, …)
- Web crawl data



What's in a User

- Explicit Signals
  - Pins, with curated text and annotations
  - Pins classified to Boards
  - Followed topics, users, boards
- Implicit Signals
  - Clicks, closeups, browses
  - Search Queries

- Pins, with curated text and annotations
- Pins classified to Boards
- Followed topics, users, boards

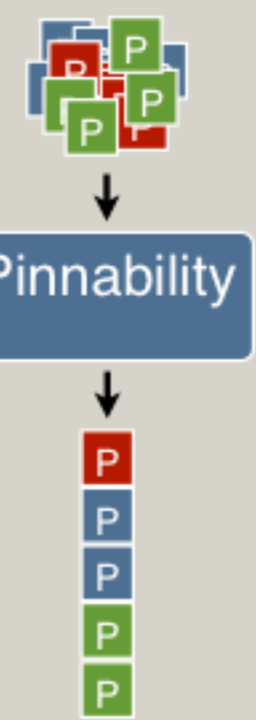
- **Implicit Signals**
  - Clicks, closeups, browses
  - Search Queries
- **Context Data**
  - App, current time, …

Pinterest Interest Graph

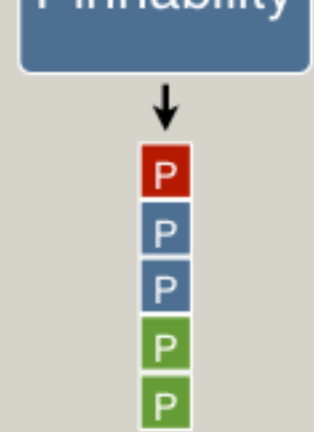
“Fried Fantasies” “Summer Fashion”

Homefeed model scale

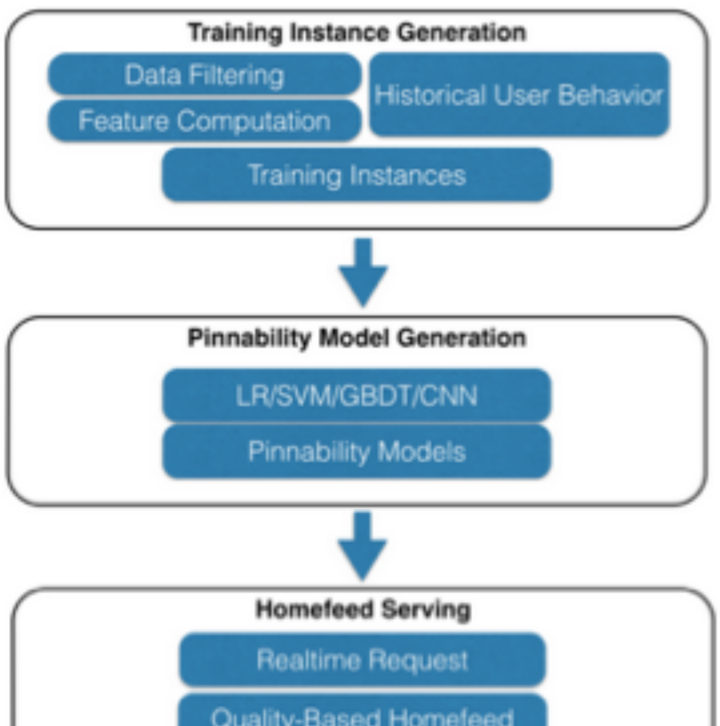
- 100M+ users
- 5B+ recommendations ranked daily
- 1B+ unique pins
- Content is evergreen - pins from 3 years ago matter
- Many different types of content (not just organic and paid)

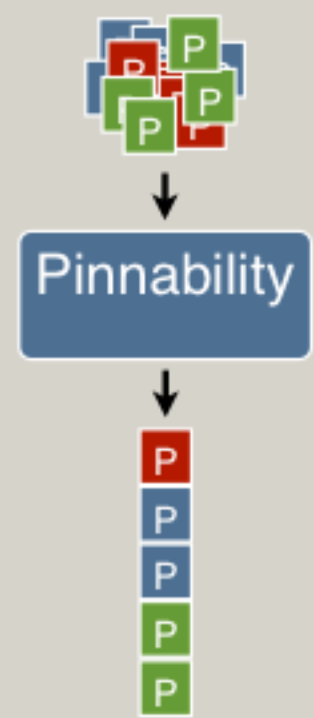


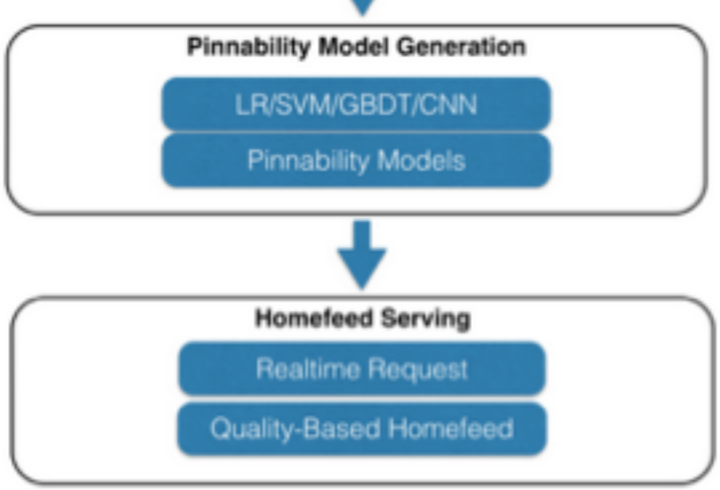
- 1B+ unique pins
- Content is evergreen - pins from 3 years ago matter
- Many different types of content (not just organic and paid)



Building a Model

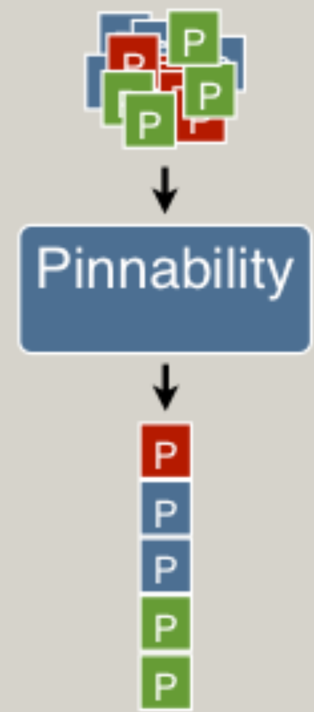






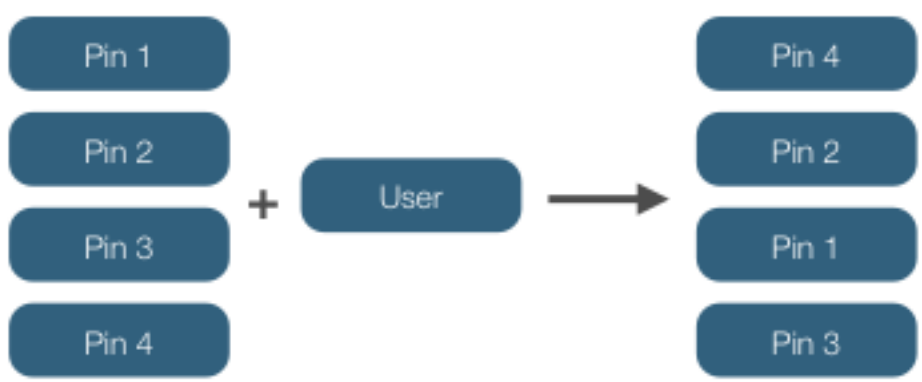
Building a Homefeed

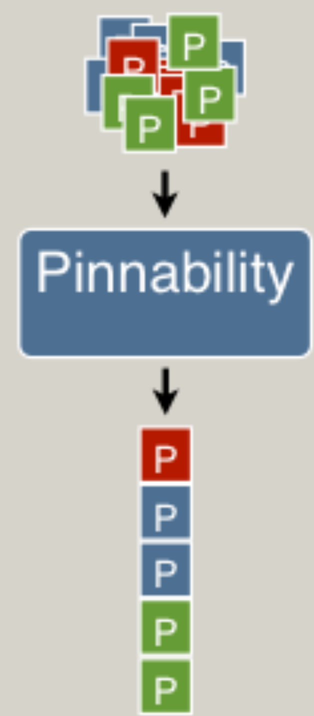
- User-independent features
  - Pin popularity
  - Web quality
  - Pagerank
  - Image Quality
- User-dependent (interaction) features
  - Interest match
  - Previous interaction with author
  - …



- Image Quality
- **User-dependent (interaction) features**
  - Interest match
  - Previous interaction with author
  - …

Building a Homefeed





What Comes Next

- We want to do better
- How do we improve on a model?
  - New features
  - Better feature engineering
  - Cleaner data
  - Handle data distribution changes
  - New learning algorithms
  - Easy AB experimentation

0 to 1

- Handle data distribution changes
- New learning algorithms
- Easy AB experimentation

Example Feature Change

- Pin Popularity
  - raw popularity (# of repins)
  - log-normalized popularity = log(raw popularity)
  - normalized popularity (# of repins / # of impressions)
  - per-country popularity (# of repins from your country / # of impressions from your country)
  - popularity weighted by recent actions
  - popularity drawn from a real-time system

Ranking Model V1

```
    double canonicalPinPopularity = 0;

    categoryMatch = computeCategoryMatch(
           userCategoryVec,
           pinJoinRawData.getCategoryVec());
      }
      if (pinJoinRawData.isSetPinJoinInfo()) {
        canonicalPinPopularity =
pinJoinRawData.getPinJoinInfo().getNumPinsMainBatch();
      }
    }

   // . . . //
   Instance instance = new Instance();

    instance.addToFeatures(
      new FeatureValue().setNumericalVal(canonicalPinPopularity));
    instance.addToFeatures(
      new FeatureValue().setNumericalVal(categoryMatch));
    instance.addToFeatures(
      new FeatureValue().setNumericalVal(isBoardEngagedBefore));
    instance.addToFeatures(
      new FeatureValue().setNumericalVal(pinOwnerBoardCount));
   // . . . //
```

0 to 1

Ranking Model V1

```
// . . . //
```

Confidential

Ranking Model V1

```
    double canonicalPinPopularity = 0;

    categoryMatch = computeCategoryMatch(
           userCategoryVec,
           pinJoinRawData.getCategoryVec());
      }
      if (pinJoinRawData.isSetPinJoinInfo()) {
        canonicalPinPopularity =
pinJoinRawData.getPinJoinInfo().getNumPinsMainBatch();
      }
    }

   // . . . //
   Instance instance = new Instance();

    instance.addToFeatures(
      new FeatureValue().setNumericalVal(canonicalPinPopularity));
    instance.addToFeatures(
      new FeatureValue().setNumericalVal(categoryMatch));
    if (user.InExperiment(MODEL_123)) {
      instance.addToFeatures(
        new FeatureValue().setNumericalVal(newFeatureValue));
    }
    instance.addToFeatures(
      new FeatureValue().setNumericalVal(pinOwnerBoardCount));
   // . . . //
```

0 to 1

Confidential

Evolving a Model in production

Confidential

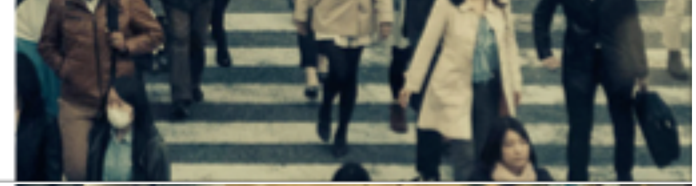
Evolving a Model in production

- Software Engineering Requirements We Want

Confidential

Evolving a Model in production

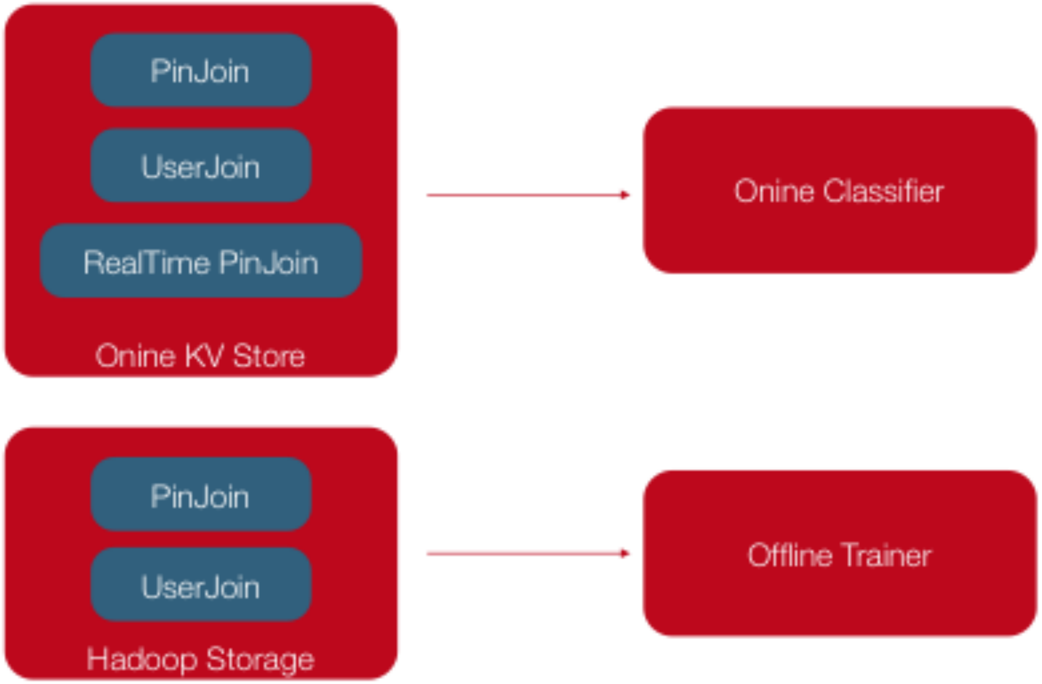
- Data comes from different sources



Confidential

Evolving a Model in production

- Data comes from different sources



Confidential

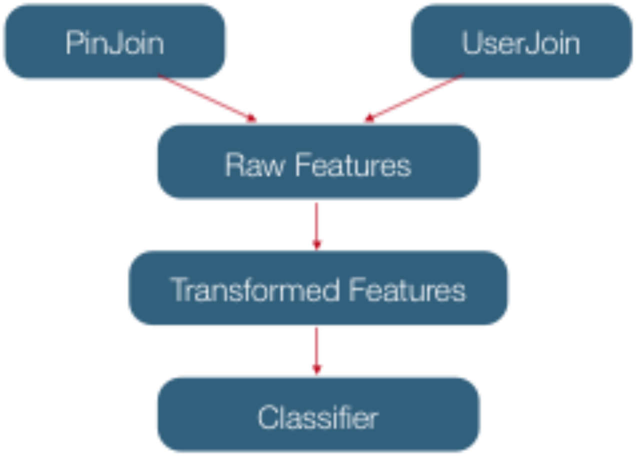
Evolving a Model in production

- Same model should run offline and online

Confidential

Evolving a Model in production

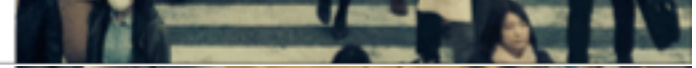
- Same model should run offline and online



Confidential

Evolving a Model in production

- Experimentation should be easy



Evolving a Model in production

- Experimentation should be easy

Evolving a Model in production

- Running multiple models long term

Evolving a Model in production

- Running multiple models long term

Pinterest Homefeed: DSL

```
data <- join(INPUTS=[pinData, followerData],
             JOIN_TYPE=CROSS_PRODUCT_KEY)
pinFollowerTopicMatch <- match(
             INPUTS=[data.pinTopicMap,
                     data.followerTopicMap])
```

Pinterest Homefeed: DSL

```
data <- join(INPUTS=[pinData, followerData],
             JOIN_TYPE=CROSS_PRODUCT_KEY)
pinFollowerTopicMatch <- match(
             INPUTS=[data.pinTopicMap,
                     data.followerTopicMap])

features <-union(INPUTS=[
  pinFollowerTopicMatch,
  pinData.pinPopularity])

scores <- linear(INPUTS=[features],
                 BIAS=0.016,
                 COEFFICIENTS=[0.15,0.23])

writeScores <- sink(scores)
```

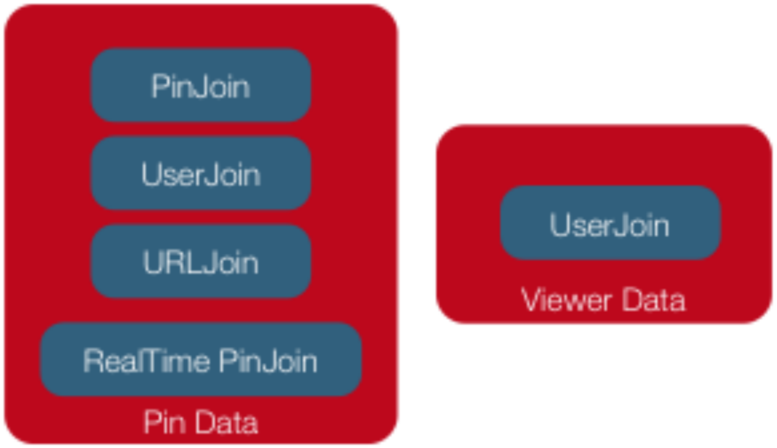
Confidential

DSL: Data Inputs

- Data is supplied by the environment
- Joins are supported within the DSL

DSL: Data Inputs

- Data is supplied by the environment
- Joins are supported within the DSL




```
pinSourceData <- join(INPUTS=[i0, i1],
                      JOIN_TYPE=LEFT,
                      JOIN_KEYS=[pinnerId, id])
```

DSL: Feature Engineering

- Operate and transform on any raw features
- Support UDFs for call out to native code

```
pinWCloseupRate <- ratio(
  INPUTS=[pinCloseupCount,
          pinWeightedImpression],
  NUMERATOROFFSET=0.002,
```

- Operate and transform on any raw features
- Support UDFs for call out to native code

```
pinWCloseupRate <- ratio(
  INPUTS=[pinCloseupCount,
          pinWeightedImpression],
  NUMERATOROFFSET=0.002,
  DENOMINATOROFFSET=0.1,
  OUTPUT_FOR_INVALID_INPUT=0.02,
  OUTPUT_FOR_NULL_INPUT=0.02)


v2SignalX <- udf(
  INPUTS=[data.follower, data.pinner, data.pin],
  CLASS=com.pinterest.pinnability.udf.V2SignalX)
```


DSL: Feature Engineering

- Easily support user-dependent features

```
data <- join(INPUTS=[pinData, followerData],
             JOIN_TYPE=CROSS_PRODUCT_KEY)
pinFollowerTopicMatch <- match(
             INPUTS=[data.pinTopicMap,
                     data.followerTopicMap])
```

- Easily support user-dependent features

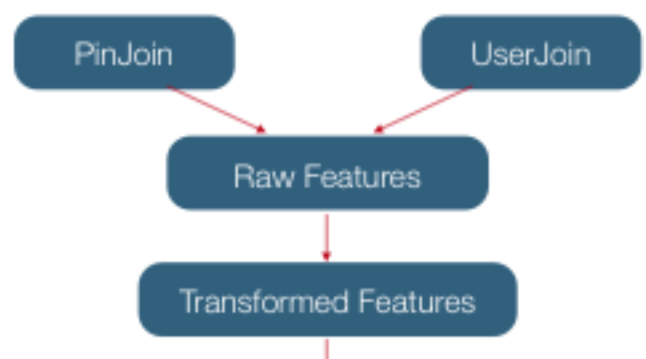
```
data <- join(INPUTS=[pinData, followerData],
             JOIN_TYPE=CROSS_PRODUCT_KEY)
pinFollowerTopicMatch <- match(
             INPUTS=[data.pinTopicMap,
                     data.followerTopicMap])
```


DSL: Multiple Outputs

- Can output different stages for different tools from same config

```
features <- union(INPUTS=[
    data.pinOnlyFeatures,
    data.followerOnlyFeatures])

writeFeatures <- sink(features)
scores <- linear(INPUTS=[features],
```

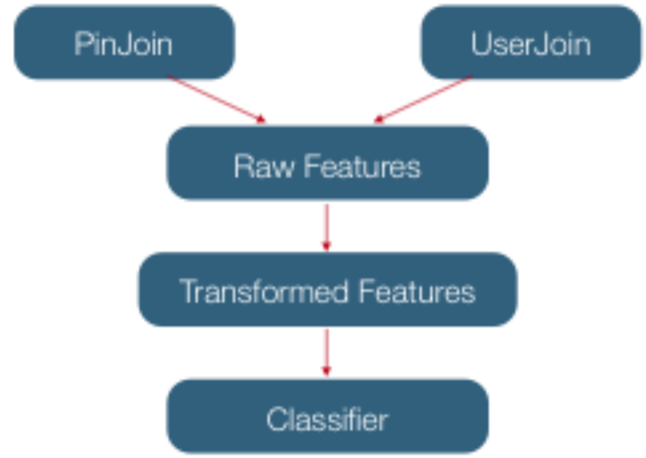


same config

```
features <- union(INPUTS=[
    data.pinOnlyFeatures,
    data.followerOnlyFeatures])

writeFeatures <- sink(features)
scores <- linear(INPUTS=[features],
                 BIAS=0.016,
                 COEFFICIENTS=[0.15,0.23])

writeScores <- sink(scores)
```



DSL: Different Classifiers

- Easily plug in different classifiers

```
logodds <- forest(INPUTS=[features],
                  ID=1c97748f_0a93_456b_b808_34c8b7917fe5)
scores <- sigmoid(INPUTS=[logodds],
                  MULTIPLIER=0.05, OFFSET=0.26449)
```

```
logodds <- forest(INPUTS=[features],
                  ID=1c97748f_0a93_456b_b808_34c8b7917fe5)
scores <- sigmoid(INPUTS=[logodds],
                  MULTIPLIER=0.05, OFFSET=0.26449)
```


Evolving a Model in Production

- Data comes from different sources
- Same model offline and online
- Can swap in infrastructure without affecting models
- Experiment is a single model file, can run independently of other experiments
- Deploying a model is a config push
- Can easily use different models for international users, new users, etc.

- Can swap in infrastructure without affecting models
- Experiment is a single model file, can run independently of other experiments
- Deploying a model is a config push
- Can easily use different models for international users, new users, etc.

Evolving a Model in Production

- Experiments
  - Can push a model to 1% and watch realtime stats
- Debugging
  - DSL interpreter can capture and log data and output at various stages of transform
  - Transparent to model developer
- Infra + Models are decoupled

- Debugging
  - DSL interpreter can capture and log data and output at various stages of transform
  - Transparent to model developer
- Infra + Models are decoupled
  - Models can be swapped in without knowing underlying infra

Data Source Changes

- Can now iterate quickly on feature changes, model changes, new data sources
- What about existing data source change?
  - Adding a feature should automatically add it to your entire dataset
  - Underlying features evolve and improve
  - Training data drifts over time

dataset

- Underlying features evolve and improve
- Training data drifts over time

Lessons from Homefeed

- More to a successful data product than ML
- Encapsulate as much of the model process as possible
- Separate code from model config
- Create consistent environments
- Make debugging easy
- Make experimentation easy

- Make debugging easy
- Make experimentation easy

thank you!