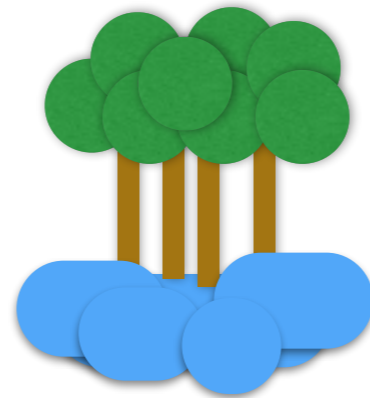


Growing Randomized Trees in the Cloud



Budapest BI Forum - 2013



Olivier Grisel

@ogrisel

Datageek, contributor to scikit-learn, works with Python / Java / Clojure / Pig, interested in Machine Learning, NLProc, {Big|Linked|Open} Data and braaaains!

Paris, France · <http://github.com/ogrisel>



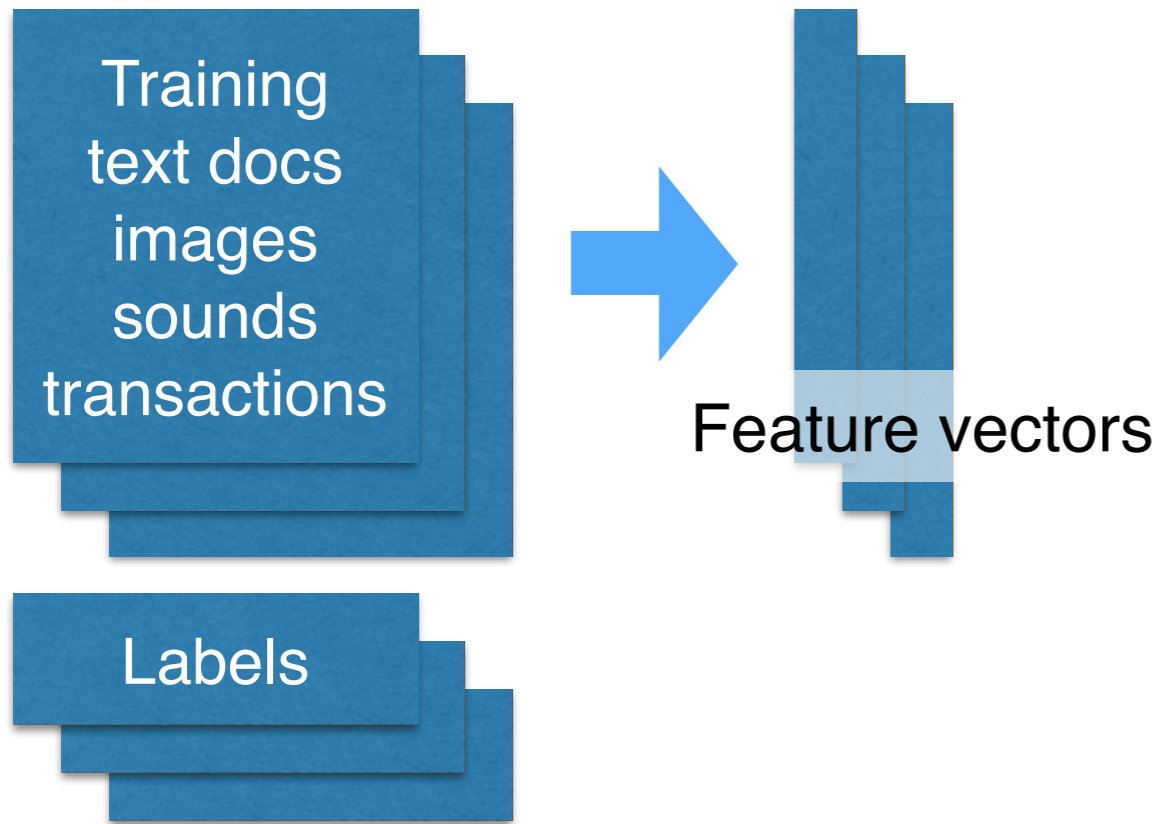
Outline

- Predictive Modeling
- Application to “Learning to Rank” for web search
- Forests of Randomized Trees
- The Python ecosystem: Scikit-learn, IPython, StarCluster

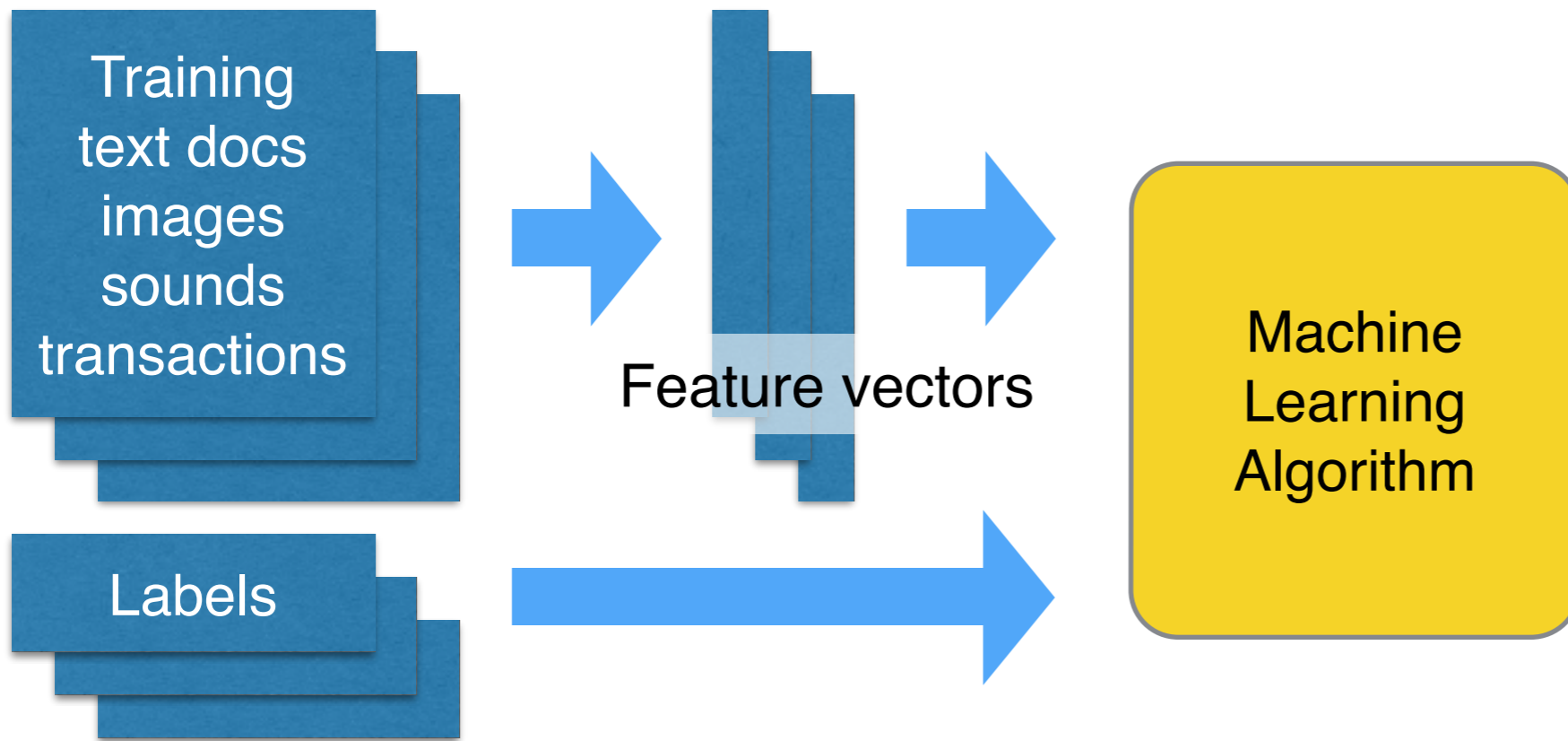
Training
text docs
images
sounds
transactions

Labels

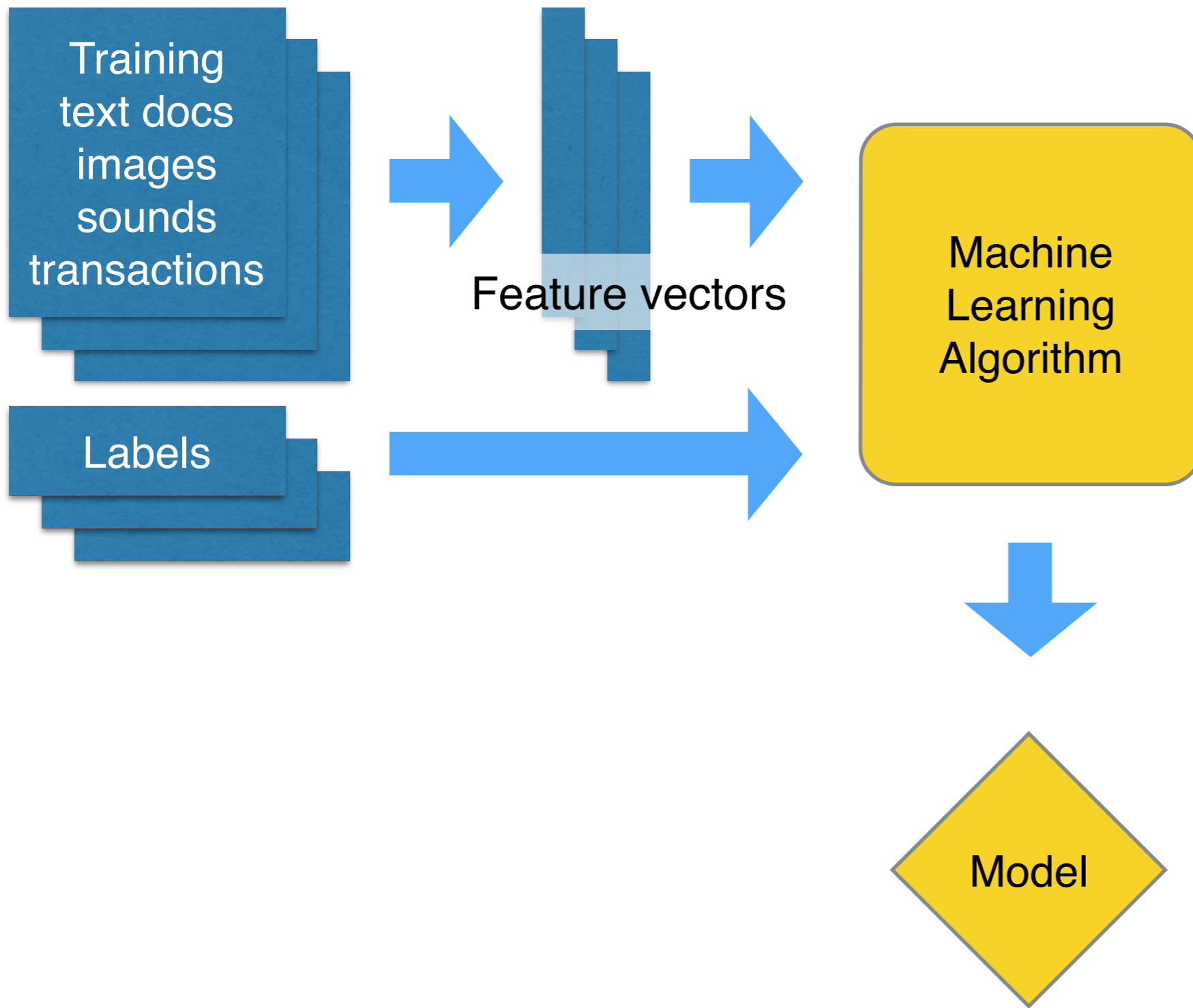
Predictive Modeling Data Flow



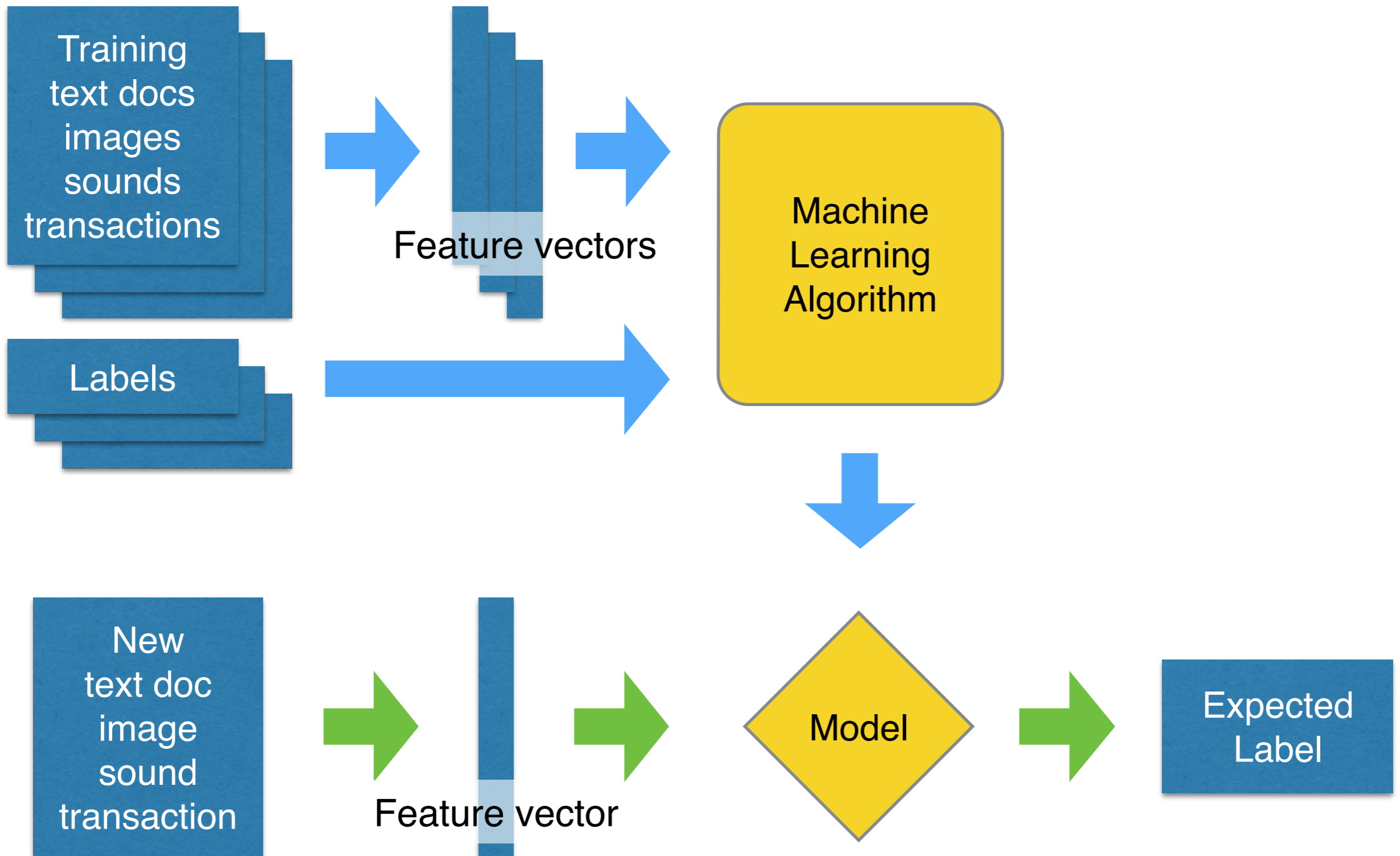
Predictive Modeling Data Flow



Predictive Modeling Data Flow





Predictive Modeling Data Flow



Predictive Modeling Data Flow

Example: Learning to Rank

WEB IMAGES VIDEOS NEWS MORE

 budapest business intelligence conference 

46,000 RESULTS

[Conference Alerts - Topic Listing - Academic conferences worldwide](#)
...
www.conferencealerts.com/topic-listing?topic=Business ▾
Business Conferences Worldwide Upcoming events in **business** and related fields
Hosted by Conference Alerts - Find details about academic **conferences** worldwide.

[IDC BUSINESS INTELLIGENCE ROADSHOW 2008](#)
atre.com/pdf/idc-2008/hungary/Postevent_Report_BI08HU.pdf · PDF file
IDC Business Intelligence Roadshow 2008, **Budapest**, 14 October 2008, Novotel
Centrum - 1 - ... **Conference** Agenda. 08:30 Registration and Welcome Coffee.

[Conference of the European Decision Sciences Institute, EDSI 2013 ...](#)
events.hellotrade.com/conferences/conference-of-the-european... ▾
About. **Conference** of the European Decision Sciences Institute is a most important
conference. It is going to be held in **Budapest**, Hungary for four consecutive days.

[What is an executive woman like in 2014? | The Budapest Business ...](#)
www.bbj.hu/events ▾
Lecturers at the **Budapest Business** Journal's **business conference** on Women in
Leadership 2014 demonstrated both their ability to argue their case as well as their wit.

[Budapest EPCA - a culinary journey | business travel, conferences |](#)
www.icis.com/blogs/icis-chemicals-confidential/2010/10/budapest... ▾
10/1/2010 · Trusted market **intelligence** ... 2010 in **business** travel, **conferences**. ...
Amsterdam APIC aromatics aromatics **conference** baseoils Berlin Brussels **Budapest** ...

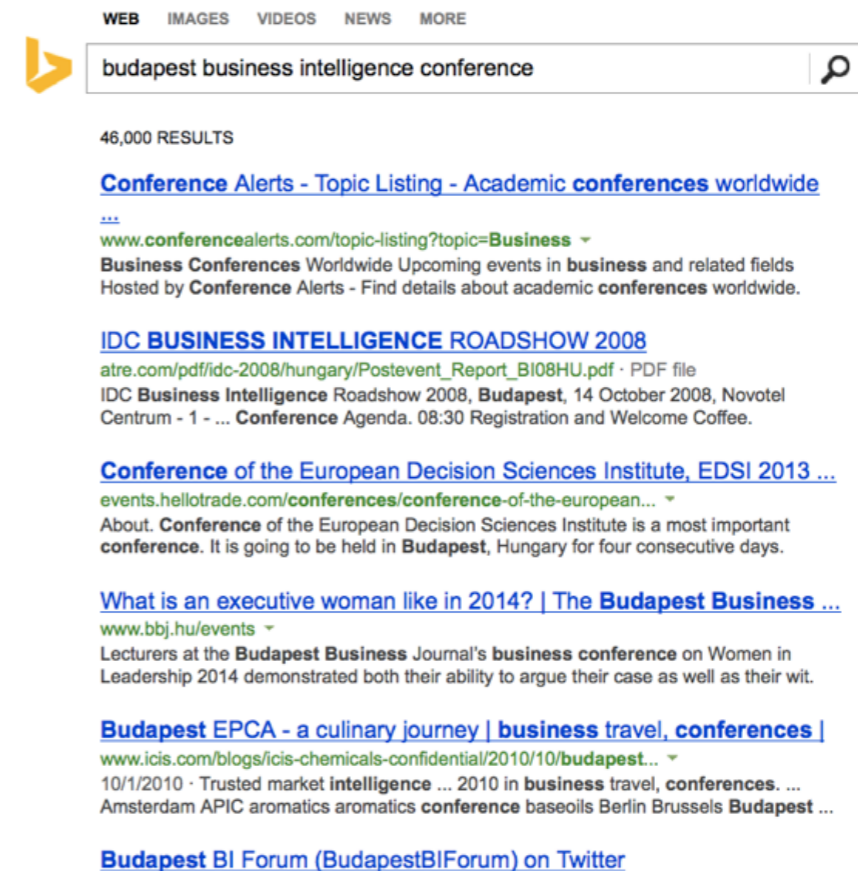
[Budapest BI Forum \(BudapestBIForum\) on Twitter](#)

Example: Learning to Rank

- Learning to rank web search results
- Input: numerical descriptors for query / results pairs
- Target: relevance score
 - 0: irrelevant
 - 1: somewhat relevant
 - 2: relevant

Input Features

- Result page descriptors:
 - PageRank, Click Through Rate, last update time...
- Query / Result page descriptors
 - BM25, TF*IDF cosine similarity
 - Ratio of covered query terms
- User context descriptors: past user interactions (clicks, +1), time of the day, day of the month, month of the year and user language
- ... typically more than 40 descriptors and up to several hundreds



Quantifying Success

- Measure discrepancy between predicted and true relevance scores
- Traditional Regression Metrics:
 - Mean Absolute Error
 - Explained Variance
- But the ranking quality is more important than the predicted scores...

NDCG: a ranking metric

```
In [85]: ndcg([2, 4, 0, 1, 1, 0, 0], rank=5)
```

```
Out[85]: 0.86253003992915656
```

```
In [86]: ndcg([0, 0, 0, 1, 1, 2, 4], rank=5)
```

```
Out[86]: 0.13201850690866795
```

```
In [87]: ndcg([0, 0, 0, 1, 1, 2, 4], rank=3)
```

```
Out[87]: 0.0
```

```
In [88]: ndcg([4, 2, 1, 1, 0, 0, 0], rank=5)
```

```
Out[88]: 1.0
```

NDCG in Greek

$$DCG_k(rel) = \sum_{i=1}^k \frac{rel_i}{\log_2(i+1)}$$

DCG_k == Discounted Cumulative Gains at rank k

Data from Microsoft Bing

- <http://research.microsoft.com/en-us/projects/mslr>
- 10K or 30K anonymized queries (terms and results URLs)
- 10K queries:
 - ~1.2M search results
 - 136 descriptors
 - 5 target relevance levels
 - ~650MB in NumPy

Datasets

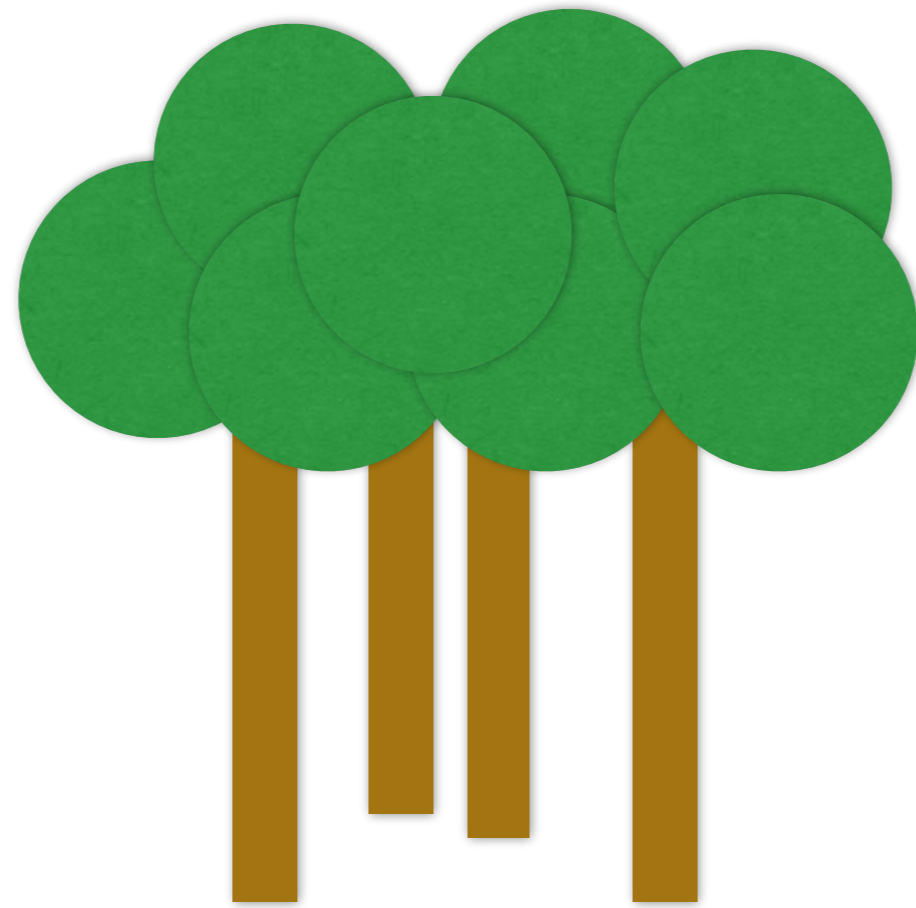
The datasets are released on June 16, 2010.

Datasets	Size	MD5
MSLR-WEB10K	~ 1.2G	97c5d4e7c171e475c91d7031e4fd8e79
MSLR-WEB30K	~ 3.7G	4beae4bee0cd244fc9b2aff355a61555

Disclaimer:

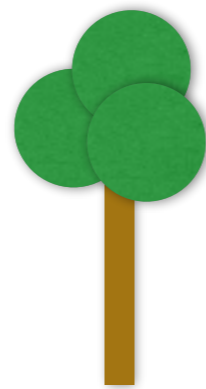
this is **not Big Data**

- Couple of GB: fits in RAM on my laptop
- But painful to download / upload over the internet.
- Processing and modeling can be CPU intensive (and sometimes distributed).

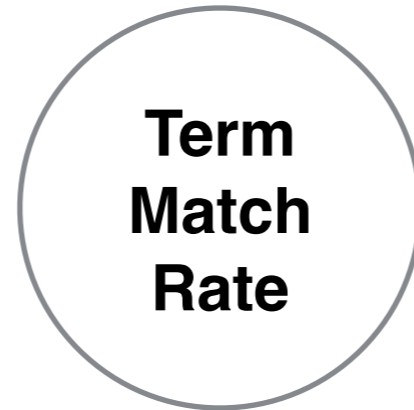


Growing randomized trees

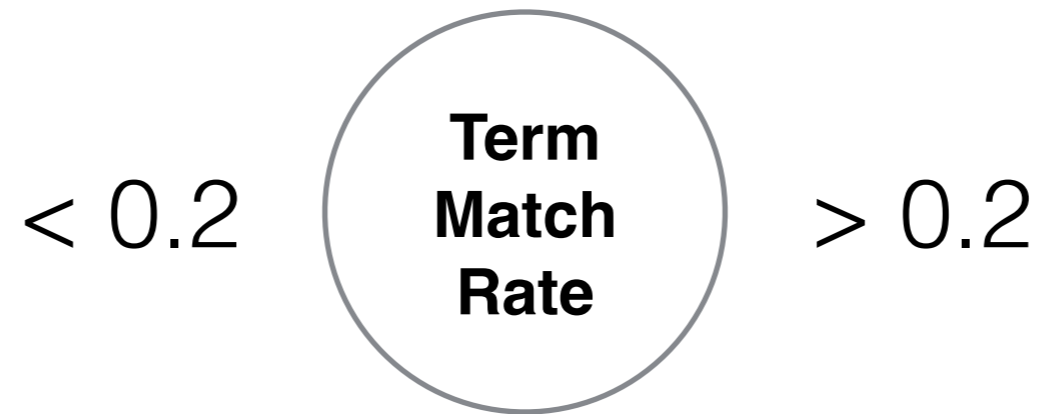
Training a Decision Tree



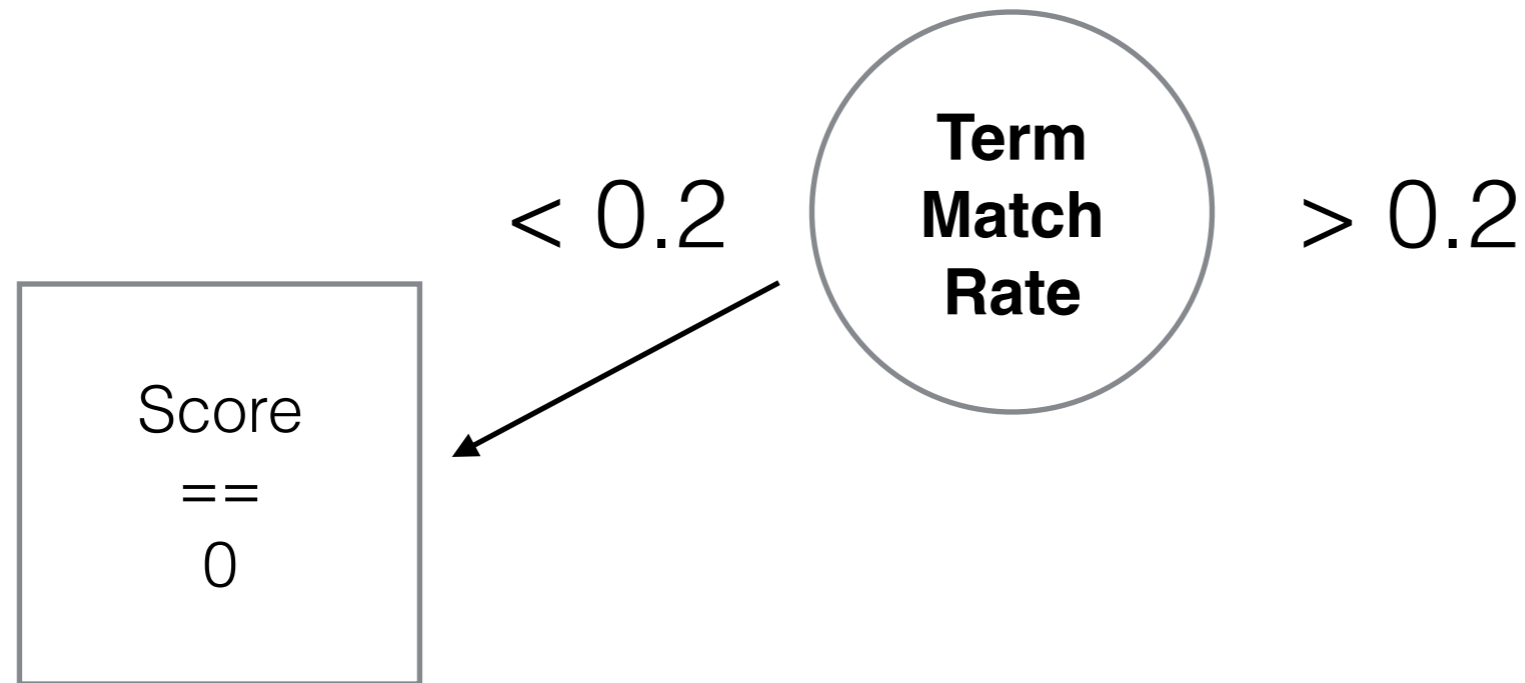
Training a Decision Tree



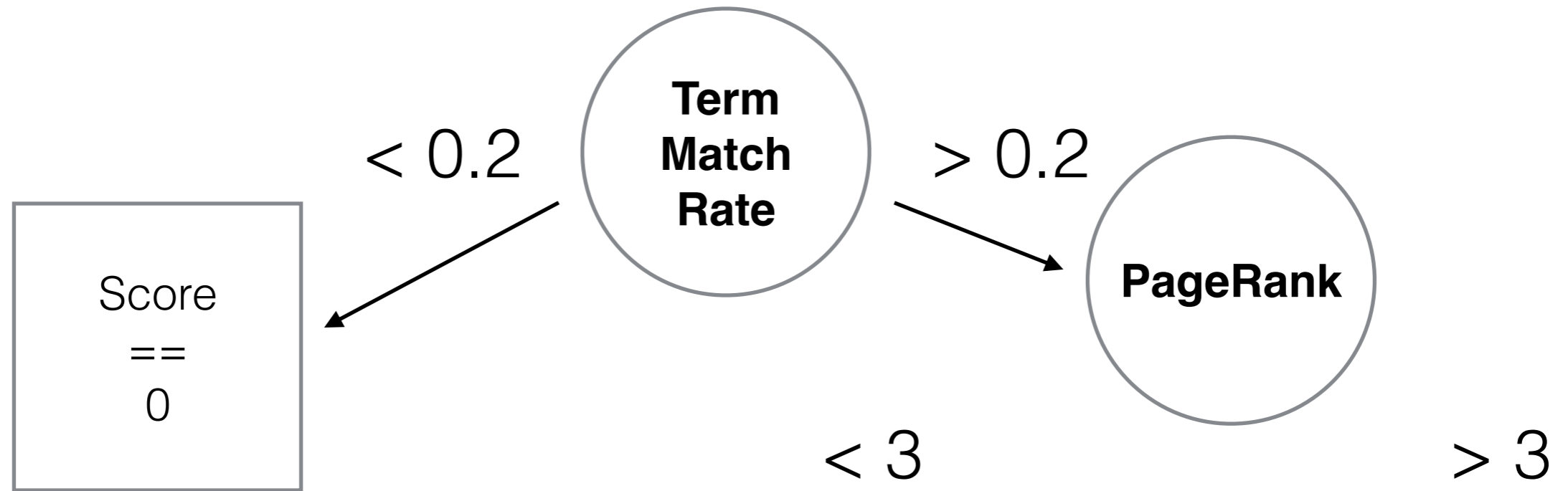
Training a Decision Tree



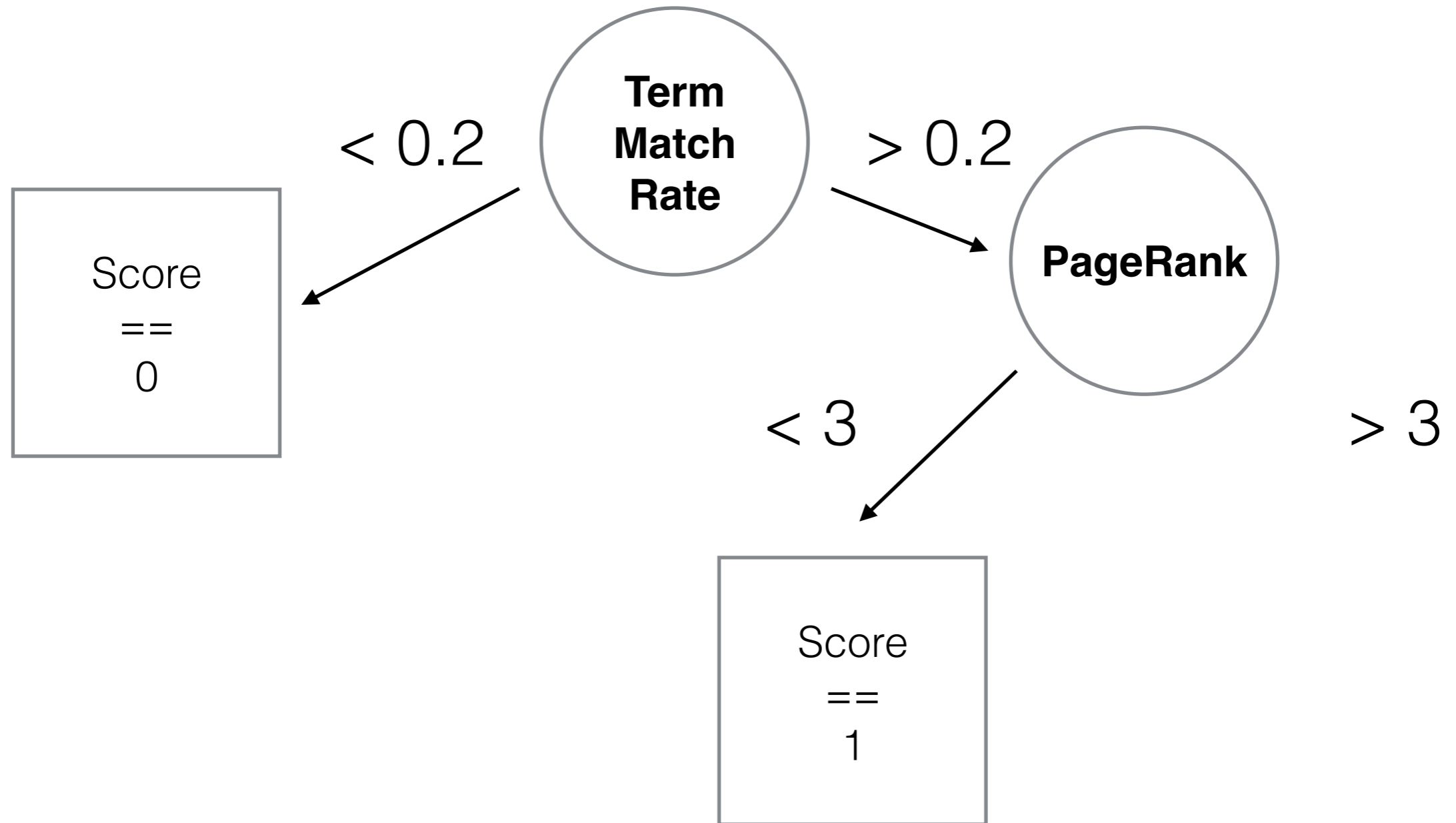
Training a Decision Tree



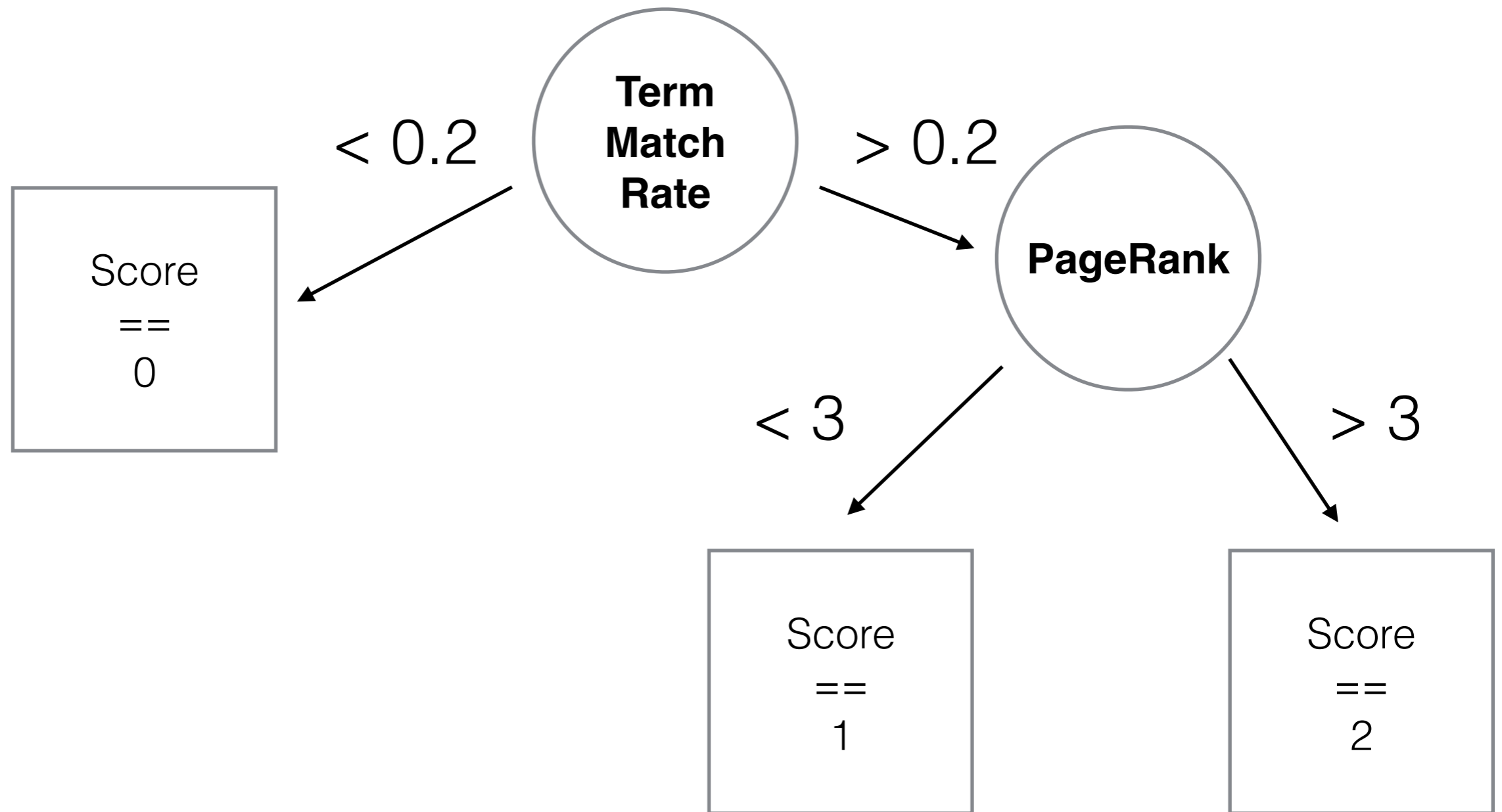
Training a Decision Tree



Training a Decision Tree



Training a Decision Tree



Training a Randomized Tree

- Pick a **random subset of features** (e.g. TFIDF, BM25, PageRank, CTR...)
- Find the feature that best splits the dataset
- **Randomize the split threshold** between observed min and max values
- Send each half of the split dataset to build the 2 subtrees

Training a Forest

- Train n random trees independently
 - Use different PRNG seeds
- At prediction time, make each tree predict its best guess and:
 - make them vote (classification)
 - average predicted values (regression)

Extra Trees

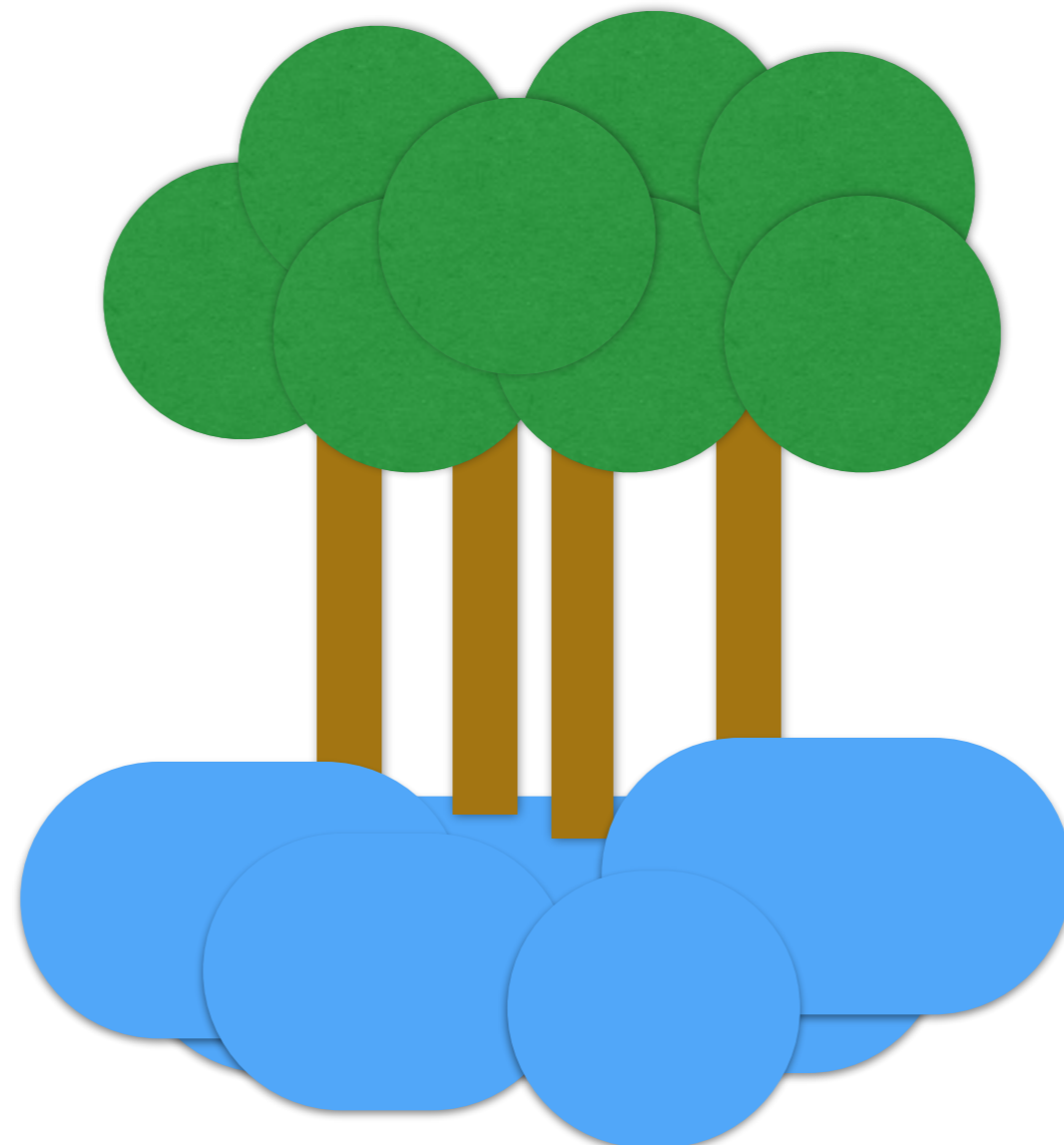
one node with 8 CPUs



```
from sklearn.ensemble import ExtraTreesRegressor

trees = ExtraTreesRegressor(n_estimators=100, n_jobs=8)
trees.fit(X_train, y_train)
```

```
y_predicted = trees.predict(X_vali)
r2_score(y_vali, y_predicted)
```



Growing randomized trees
on the cloud

10x8 cores cluster on EC2 in 20min



```
0 [~]$ starcluster start ip -s 10 --force-spot-master
StarCluster - (http://star.mit.edu/cluster) (v. 0.9999)
Software Tools for Academics and Researchers (STAR)
Please submit bug reports to starcluster@mit.edu

>>> Using default cluster template: ip
>>> Validating cluster template settings...
>>> Cluster template settings are valid
>>> Starting cluster...
>>> Launching a 10-node cluster...
>>> Launching master node (ami: ami-765b3e1f, type: c1.xlarge)...
>>> Creating security group @sc-ip...
SpotInstanceRequest:sir-66705632
>>> Launching node001 (ami: ami-765b3e1f, type: c1.xlarge)
SpotInstanceRequest:sir-d2bcf232
```



EC2 Dashboard

Events

Tags

INSTANCES

Instances

Spot Requests

Reserved Instances

IMAGES

AMIs

Bundle Tasks

ELASTIC BLOCK STORE

Volumes

Snapshots

NETWORK & SECURITY

Launch Instance

Connect

Actions ▾



Filter: All instances ▾

All instance types ▾

Search Instances

1 to 10 of 10 Instances

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
<input type="checkbox"/>	node008	i-20b02b44	c1.xlarge	us-east-1b	● running	✓ 2/2 checks...
<input type="checkbox"/>	node009	i-2eb02b4a	c1.xlarge	us-east-1b	● running	✓ 2/2 checks...
<input type="checkbox"/>	node002	i-33122654	c1.xlarge	us-east-1b	● running	✓ 2/2 checks...
<input type="checkbox"/>	node004	i-38b02b5c	c1.xlarge	us-east-1b	● running	✓ 2/2 checks...
<input type="checkbox"/>	node005	i-54b02b30	c1.xlarge	us-east-1b	● running	✓ 2/2 checks...
<input type="checkbox"/>	node003	i-58b02b3c	c1.xlarge	us-east-1b	● running	✓ 2/2 checks...
<input type="checkbox"/>	master	i-a80849ce	c1.xlarge	us-east-1b	● running	✓ 2/2 checks...
<input type="checkbox"/>	node001	i-ae0849c8	c1.xlarge	us-east-1b	● running	✓ 2/2 checks...
<input type="checkbox"/>	node006	i-e10d0984	c1.xlarge	us-east-1b	● running	✓ 2/2 checks...
<input type="checkbox"/>	node007	i-ed0d0988	c1.xlarge	us-east-1b	● running	✓ 2/2 checks...



```
>>> Waiting for all nodes to be in a 'running' state...
10/10 | 100%
>>> Waiting for SSH to come up on all nodes...
10/10 | 100%
>>> Waiting for cluster to come up took 6.726 mins
>>> The master node is ec2-174-129-190-133.compute-1.amazonaws.com
>>> Configuring cluster...
>>> Running plugin starcluster.clustersetup.DefaultClusterSetup
>>> Configuring hostnames...
10/10 | 100%
>>> Creating cluster user: ipuser (uid: 1001, gid: 1001)
10/10 | 100%
>>> Configuring scratch space for user(s): ipuser
10/10 | 100%
>>> Configuring /etc/hosts on each node
10/10 | 100%
>>> Starting NFS server on master
>>> Configuring NFS exports path(s):
/home
>>> Mounting all NFS export path(s) on 9 worker node(s)
9/9 | 100%
>>> Setting up NFS took 0.663 mins
>>> Configuring passwordless ssh for root
>>> Configuring passwordless ssh for ipuser
>>> Running plugin ipython
>>> Installing Python packages on all nodes:
>>> $ pip install python-msgpack
```

```
>>> Configuring cluster took 12.865 mins
>>> Starting cluster took 20.144 mins
```

IP[y]: IPython Interactive Computing

- Notebook interface: in-browser, interactive data exploration environment
- IPython.parallel: async load-balancing API for interactive dispatching processing
- Based on ZeroMQ and msgpack for IPC

```
In [1]: from IPython.parallel import Client
lb = Client().load_balanced_view()
len(lb)
```

Out[1]: 79

```
In [2]: def compute_stuff(a, b):
import time, random

time.sleep(random.randint(0, 10))
return a ** 2 + b - 42

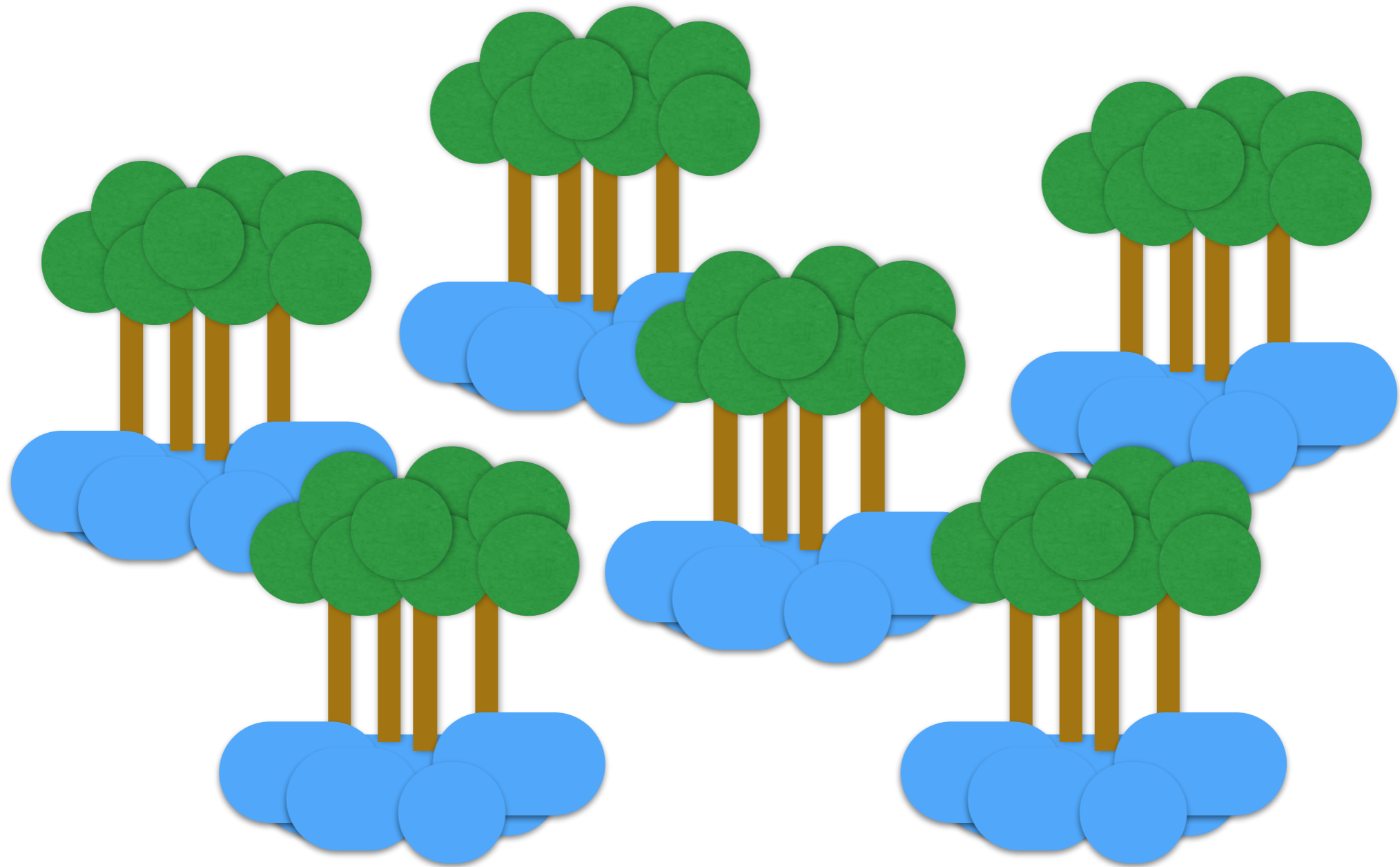
tasks = []
for a, b in zip(range(100), range(0, 200, 2)):
    tasks.append(lb.apply(compute_stuff, a, b))
```

```
In [3]: sum(t.ready() for t in tasks)
```

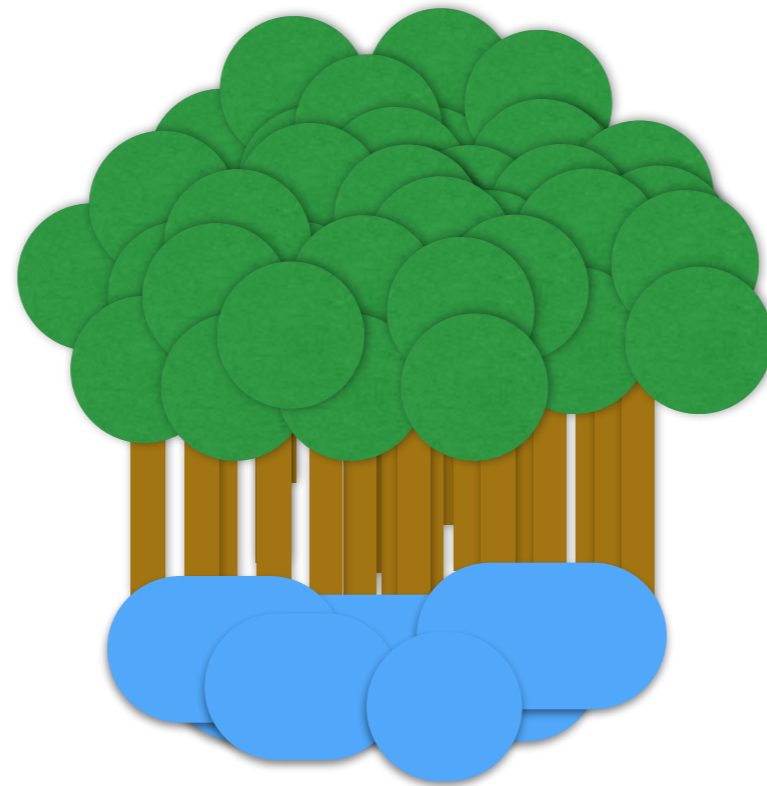
Out[3]: 29

```
In [4]: [t.get() for t in tasks][:10]
```

Out[4]: [-42, -39, -34, -27, -18, -7, 6, 21, 38, 57]



Grow random trees
in parallel in the cloud



Fetch back all the trees
as a big forest on one node

```
from copy import copy

def combine(all_forests):
    """Aggregate sub-forests into a big forest"""
    final_forest = copy(all_forests[0])
    final_forest.estimators_ = []

    for forest in all_forests:
        final_forest.estimators_ += forest.estimators_

    final_forest.n_estimators = len(final_forest.estimators_)

    return final_forest
```

Demo

<http://j.mp/parallel-mslr>

Results

- NDGC@5: ~0.52 for 500 trees on MSLR-WEB10K
- Could *maybe* be improved by:
 - increasing the number of trees (but model gets too big in memory and slower to predict)
 - replacing base trees by bagged GBRT models
 - pairwise or list-wise ranking models (not in sklearn)
- Linear regression model baseline: NDGC@5: ~0.43

Your turn now!



Personalized Web Search Challenge




2 months to go


Friday, October 11, 2013

\$9,000 • 16 teams

Friday, January 10, 2014

Dashboard

Home 
Data 
Make a submission 

Information 
Description
Evaluation
Rules
Prizes
Logs format
Organizers
Related papers
Timeline

Forum 

Leaderboard 

Competition Details » [Get the Data](#) » [Make a submission](#)

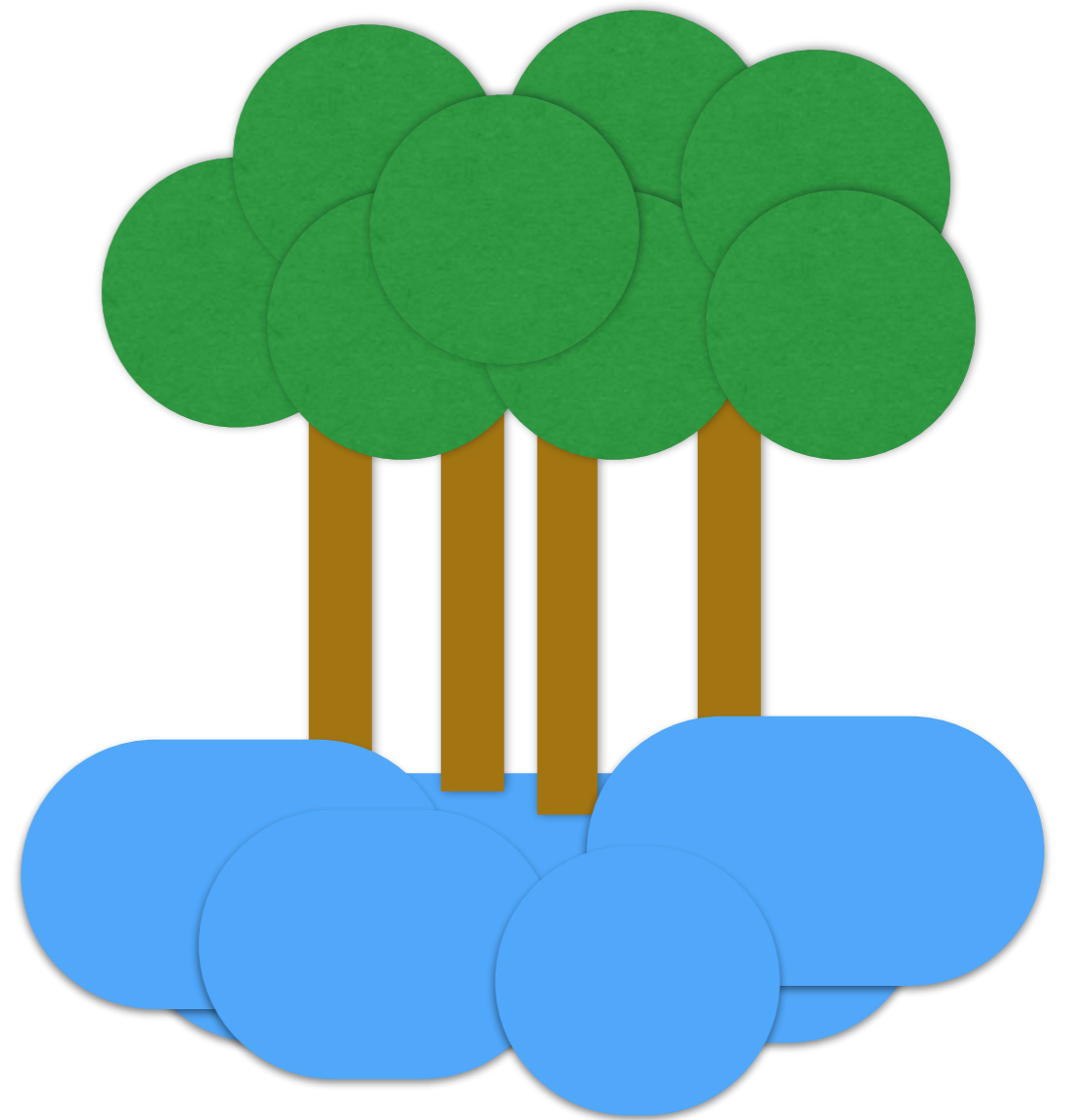
Re-rank web documents using personal preferences

The Personalized Web Search Challenge provides a unique opportunity to consolidate and scrutinize the work from industrial labs on personalizing web search using user-logged search behavior context. It provides a fully anonymized dataset shared by Yandex, which has anonymized user ids, queries, query terms, urls, url domains and clicks.

This Challenge and the shared dataset will enable a whole new set of researchers to study the problem of personalizing web search experience. The Personalized Web Search Challenge is a part of series of contests organized by Yandex over many years. This year's event is the eighth since 2004. In previous years, participants tried to [learn to rank documents](#), [predict traffic jams](#), [find similar images](#), [predict relevance of documents using search logs](#) and [detect search engine switchings in search sessions](#).

Questions?

- <http://ipython.org>
- <http://scikit-learn.org>
- <http://star.mit.edu/cluster>
- <https://github.com/pydata/pyrallel>
- <http://github.com/ogrisel/notebooks>



Backup slides

Loading the data with Scikit-learn

```
from os.path import expanduser
from sklearn.datasets import load_svmlight_file

filepath_train = expanduser('~/.data/MSLR-WEB10K/Fold1/train.txt')

X_train_sparse, y_train, qid_train = load_svmlight_file(
    filepath_train, dtype=np.float32, query_id=True)

X_train = X_train_sparse.toarray()
```

NDCG in Python

```
def dcg(relevances, rank=10):  
    """Discounted cumulative gain at rank (DCG)"""  
    relevances = np.asarray(relevances)[:rank]  
    n_relevances = len(relevances)  
    if n_relevances == 0:  
        return 0.  
  
    discounts = np.log2(np.arange(n_relevances) + 2)  
    return np.sum(relevances / discounts)  
  
def ndcg(relevances, rank=10):  
    """Normalized discounted cumulative gain (NDGC)"""  
    best_dcg = dcg(sorted(relevances, reverse=True), rank)  
    if best_dcg == 0:  
        return 0.  
  
    return dcg(relevances, rank) / best_dcg
```