Data Science ≠ Machine Learning: Some Thoughts on the Role of Data Management in the new AI-Tsunami

Jens Dittrich
Saarland Informatics Campus

Talk Manuscript for my Keynote at DEEM Workshop at SIGMOD 2018
(accompanied by slides, **scroll down for slides**)
will post it on http://twitter.com/jensdittrich

Prelude: **Buzzword Bullshit Bingo**

Buzzword [People not understanding each other]

<something>

**[Babel tower,
https://en.wikipedia.org/wiki/Tower_of_Babel#/media/File:Pieter_Bruegel_the_Elder_-_The_Tower_of_Babel_(Vienna)_-_Google_Art_Project.jpg]**

The political game **[parliament]**

Data Science vs Machine Learning

clarify term "data science" **[slides from intro lecture**]

data science pipeline

where is data management in that pipeline?

similarities with DWH

show different topics

mention projects,

DAWN

HoloClean

Spark, Flink

tons of exciting work going on at this workshop and aiDM, thumbs up, guys!

1.) opportunities for doing research at the intersection of <something> and data management,

Software 2.0, Andrey Karpathy

Where this leads to is:

f(X) = Y

software is many of those functions

RNNs are turing complete

In principle any function could be learned by some model

In principle…

which are suitable functions to investigate?

f(X) = Y

f("SELECT * FROM ...") = [table]

f("SELECT * FROM ...") = [cost estimate]

f("SELECT * FROM ...") = {suitable indexes}

f("SELECT * FROM A", "SELECT * FROM B", "SELECT * FROM ...") = {suitable indexes}

f("SELECT * FROM ...") = [LLVM bit code]

basically, **any language translation**

Software 2.0: finding out which functions to replace with learned functions

I briefly spoke about this in my VLDB 2017 keynote **[title slide]**

suitable candidates for functions are in general:

everything that is currently done by humans:

- data cleaning [Stanford, HoloClean]

- data integration

- schema design

- physical design including index selection [we did that using reinforcement learning, still fiddling around with that]

- knob tuning, a fully automatic DBA, "self-driving DBA" [see Pavlo work, our own ongoing stuff]

- DBMS installation and setup, parallel setups

- DBMS maintenance, bug fixing, trouble shooting

- DBMS security, threat prevention, mitigation, and counter measures

- DBMS performance analysis and debugging

- DBMS calibration to hardware

- again: **everything currently done by humans: DBAs and consultants**

trying to automate these things is not a new idea (some of this was researched 30 years ago already)

but these things should be revisited in the light of deep learning

--

another option is to look at functions like

f(dataset, attribute) = Index structure

So, given a dataset, and an attribute, learn a suitable index.

Indexes are currently implemented and run by machines super well already.

indexes are lightning fast, tons of research, insane performance: >10 million operations per second and thread

---

** **"The Case for Learned Indexes [Kraska et al]"**

** Idea of LI:

- observation: in an index, a KEY predicts the physical position of a VALUE

- so why not treat indexing as a regression problem, e.g.

-[screenshot from paper]

- the index IS the model

- we want to have a function f(KEY) = physical location of the VALUE

- the index _predicts_ the physical location of a VALUE

- so we train a model

- f(dataset, attribute) = Index structure

- learn cumulative density function (CDF)

- learn maximum error during training to guarantee correctness for testing

in more detail:

- train a model (deep learning in this case) to learn the data distribution of a particular attribute

- author's observation: predictions (e.g. index lookups) too slow in tensorflow

- therefore generate C++-index code based on the trained model

- eventually use a hybrid structure, e.g. a **model that is recursively refined**

- so far read-only structures only, no inserts/updates

- authors discuss B-trees, hash tables, and bitmap indexes

- extensions to multiple dimensions planned

** result summary:

show something from the paper

- performance of the learned indexes in the same ballpark as main-memory optimized indexes

- but: memory footprint much better (~2 orders of magnitude)

**** A Criticism of "Learned Indexes" ****

Let’s be precise here:

**** A Positive Criticism of "Learned Indexes" ****

nice high-level observation:

- "an index predicts the position of a value"

- "an index is a model"

- a paper that **stirs discussion** (we need much of this at SIGMOD/VLDB) rather than super-polished (possibly) incremental papers

- Food for thought

we need more of these kind of controversial papers, much better than some super-polished paper!

**** Some Constructive Criticism of "Learned Indexes" ****

I am not the first to criticize this work, see blog articles by Thomas Neumann, TUM and the Stanford DAWN group. The LI work has pros and cons (as any work).

some thoughts...

****Trees vs Models****

[show B-tree]

every node in a tree/trie is a model of the data underneath anyways,

not only the entire B- or whatever tree is a model, every node is already a model

this is a somewhat old observation

B-trees are already regression trees (as mentioned in the talk)

But also: **each node in a tree is a model**!

**** coarse-granular/sparse indexes/hybrid indexes ****

[show one-level tree]

[show two-level tree]

this is recursive modeling of the data!

hybrid figure from paper

a b-tree is NOT a black-box model (as claimed in the talk), it is white-box

a b-tree is a model of a model of a model, each node and even the leaves in a sparse tree are models!

has been known and exploited for long

e.g. bulkloading, (co-)partitioning, any technique using data distribution for tuning, statistics, data partitioning for joins [e.g. Mirek Riedewald's join papers for instance]

This hybridness is also prominent in index structures that adapt their node types to the particular distribution in a range, e.g. ARTful index, or any other multi-level hybrid index

**** Work on modeling the data domain ****

histograms, density estimation, any other classical method to describe the data distribution, entropy-based encoding, index compression, succinct trees, minimal perfect hashing, etc. etc. some overlap here, not clear to me what LI improves here, basically these are also statistical learning methods

[show train/test graph]

Example: decision trees!

[decision tree]

trade-off between generalization and overfitting very visible

more layers: more fitting, eventually overfitting **[show]**

less layers: less fitting to the data, more generalization **[show]**

sweet spot

So, let's talk about **A.I.**

I mean **Adaptive Indexing** of course.

** briefly introduce [from Immanuel's Damon 2018 paper]

[show cracking graphic]

adaptive indexing partitions the data according to the search predicates found in the queries

**** a classification of tree-based partitioning strategies:**

tree/trie

tree: according to data inserts -> data distribution
trie: according to the domain of the data inserts

trai/tria
trai: according to the query predicates -> query predicate distribution
(tria: according to the domain of the query predicates)

trie and tria are typically more or less the same

adaptive indexes LEARN the query predicates and partition the column according like that

standard cracking is a trai!

stochastic cracking is a **blend of a trai and a trie! A traiie?** (just kidding)

many methods in-between, but all about LEARNING the search predicates

the family of **adaptive indexing LEARNS the search predicates**

in contrast**, LI LEARNs the data distribution**

this relationship is unfortunately not discussed at all in the LI paper

note that there is also: "**Adaptive Adaptive indexing [Schuhknecht et al ICDE 2018]"**

**** Code generation ****

LI uses code-generation, baselines however are off-the-shelf, not a fully fair comparison

baselines should at least be calibrated to the data distributions/domains

system and code calibration correlates heavily with code generation, very vague bounsdary between the two

**** Repeatability (in general) ****

In general (from my humble experience): index structures in main memory are very sensitive to minor programming tricks.

Also see our previous work on indexing:

tree-indexing [ICDE 2015] and **hashing: [PVLDB 2016]**

partitioning [VLDB 2015] and **cracking [VLDB 2014, bpa]**

**** The insanity of evaluating indexes in main memory **** **[scream]**

Change a single code line, and all of a sudden the CPU is enabled to perform completely different optimizations

e.g. dependent vs independent statements, memory stalls, parallelization

there are tons of weird CPU-tricks/optimizations that change everything, e.g. SIMD, pipelining, branch mis-predictions, etc.

So, you assume that your index is fast, because you played some clever algorithmic trick.

Then you find out that the CPU just played some dirty trick you were not aware off.

see: basically, every database on new hardware-paper every published.

the theorists are kind of right here (unfortunately, and it hurts to say this ;-)): the complexity of your index often won't change, it is just some constant that changes (which, however, may be huge and plays a role in real life…)

so, tiny little code changes may lead to factors in performance differences [visual support, maybe from DAMON 2018 paper?]

as LI are in the same ballpark as state-of-the-art

for LI, as for any other index, I believe it is important to investigate whether these are principal differences or just due to implementation/code generation variations

looking forward to repetitions of LI ;-)

code available?

LI summary:

The glass is half full: a nice fresh view on indexing, nice observations, food for thought

The glass is half empty: (still) some doubts on practicality of this approach, and diff to related work.

2.) experiences from teaching <something>

undergrad seminar on "data science", teaching award from student's council for the best seminar

intro lecture "Intro to Databases" -> "Intro to Data Science" (ongoing this semester)

syllabus

screenshots from the system

slides

problems

3.) experiences from solving problems in the <something>-domain together with domain experts.

** project with German weather service (DWD)

[some screen shot]

idea: the bug is the feature

error in short term weather forecasts (nowcating) -> lightning prediction

nowcasting: 2D models

background: thunderstorm: wind upward movement in the cloud (I mean a real cloud)

not modelled by nowcasting

hypothesis: error in nowcasting correlates to a cloud where we will see lightning

**** data science consulting**

daimond.ai

~80-90% relatively simple problems, you do not need super fancy researchy deep learning stuff for that

Executive Summary:

1.
remind people that we are one third of what people mean when they say “data science”

2.
We should investigate Software 2.0, f(X) = Y

In particular, when f(X) is a function currently implemented by humans.

3.
We need to support and push programs and lectures in data science

4.
get out of your LAB and solve some real problems

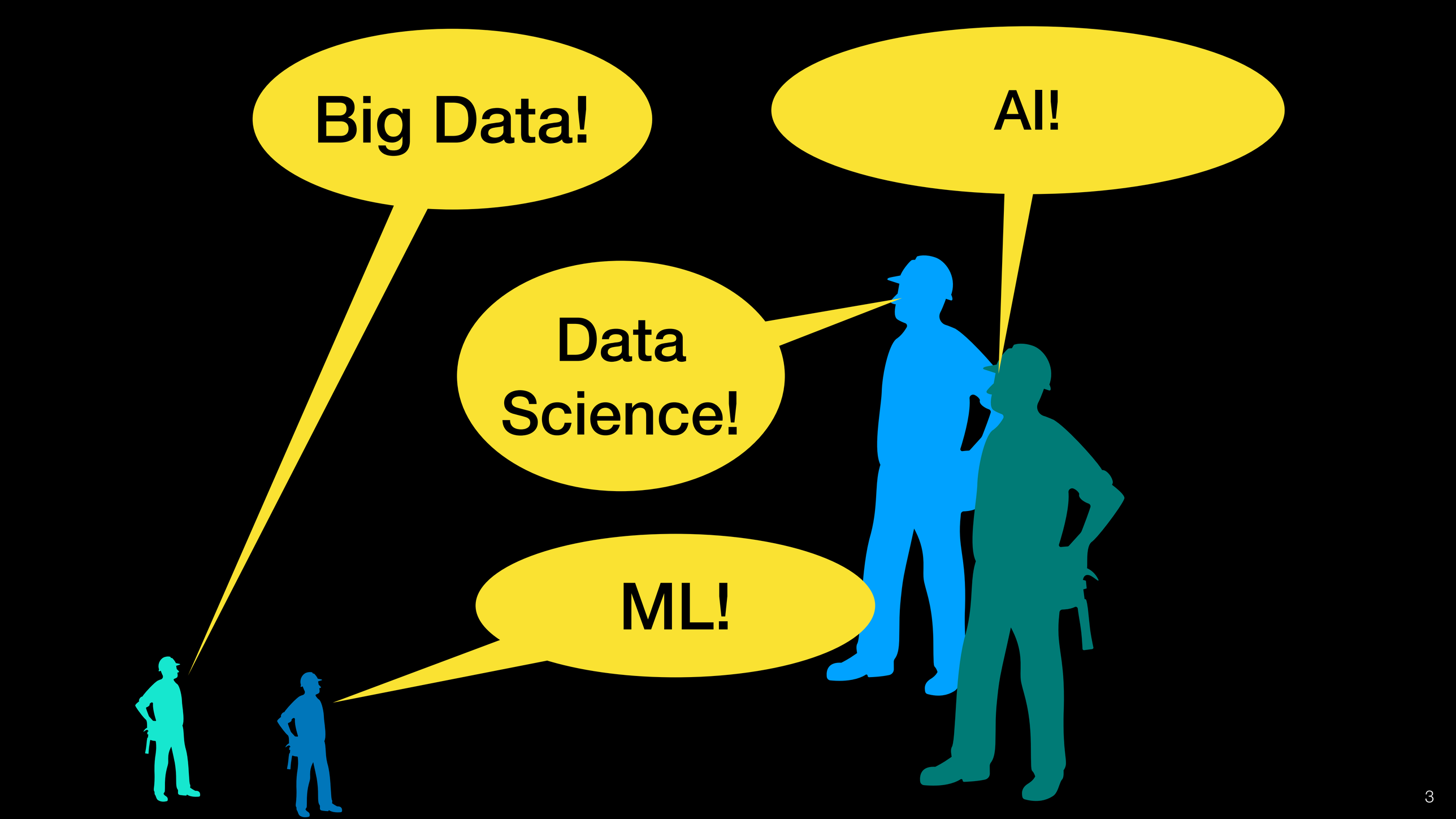
Data Science ≠ Machine Learning

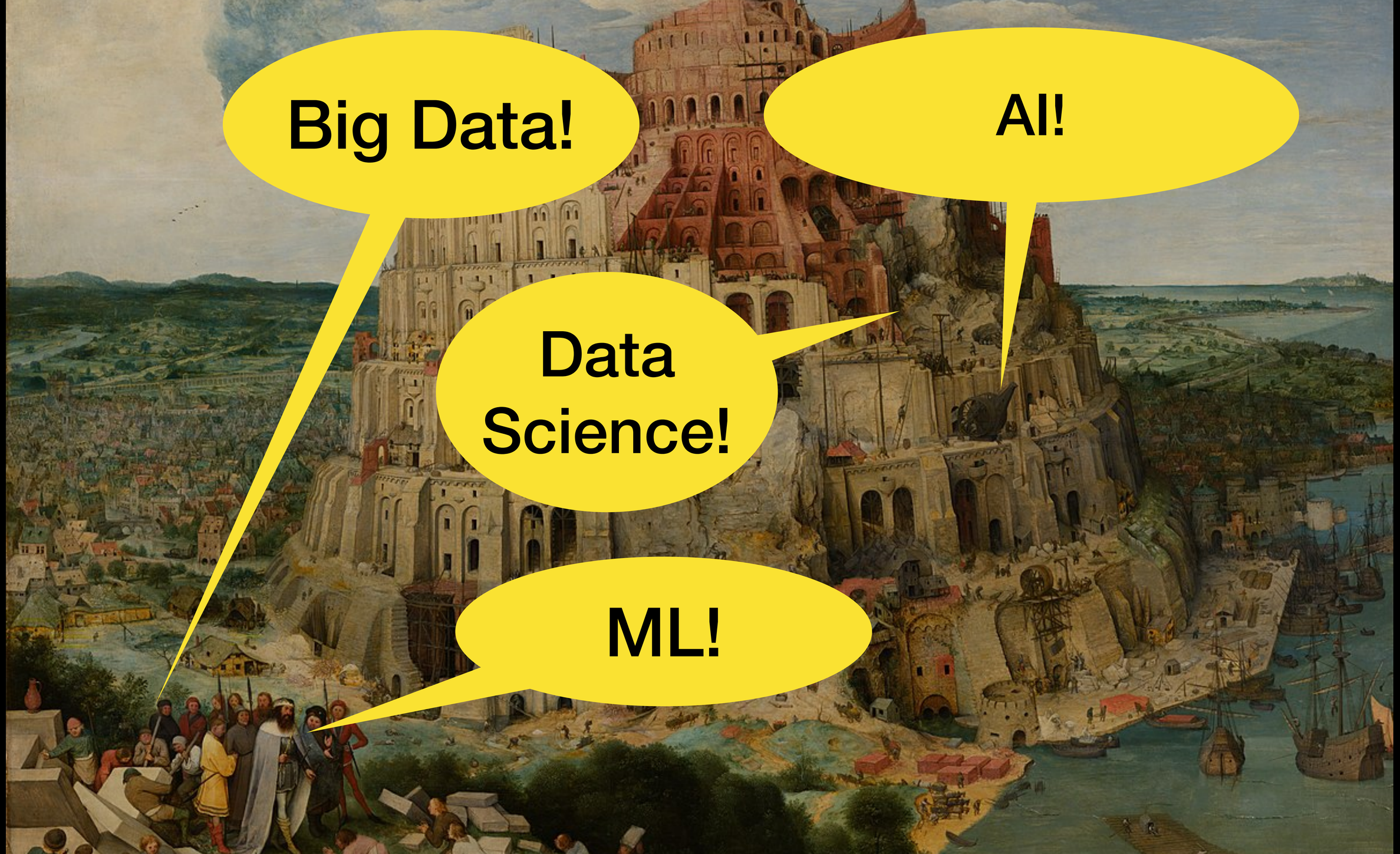
Some Thoughts on the Role of Data Management in the new AI-Tsunami

Jens Dittrich
Saarland Informatics Campus
twitter.com/jensdittrich

Note: presentation = slides ⌘ manuscript

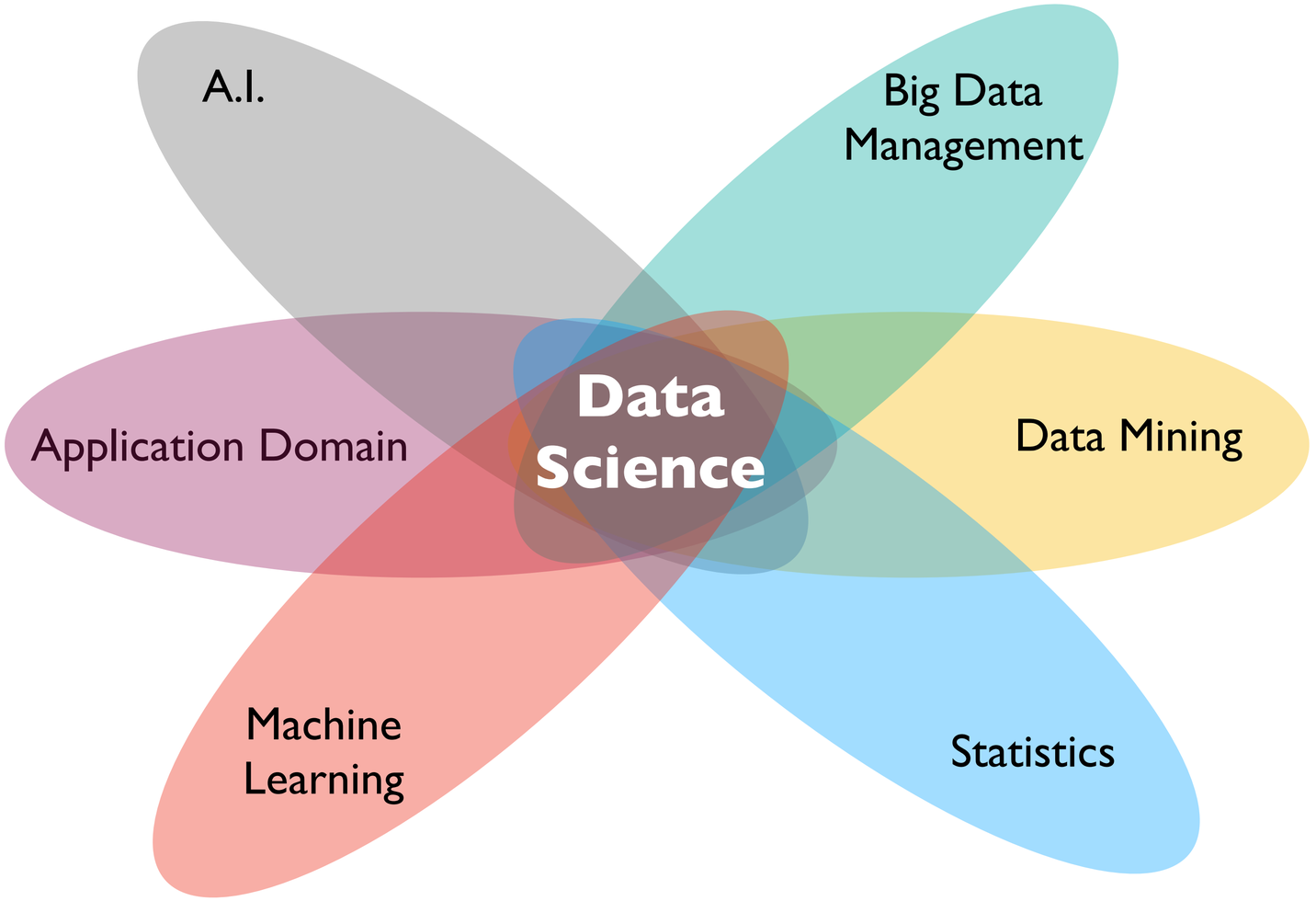
1 Prelude: Buzzword Bullshit Bingo

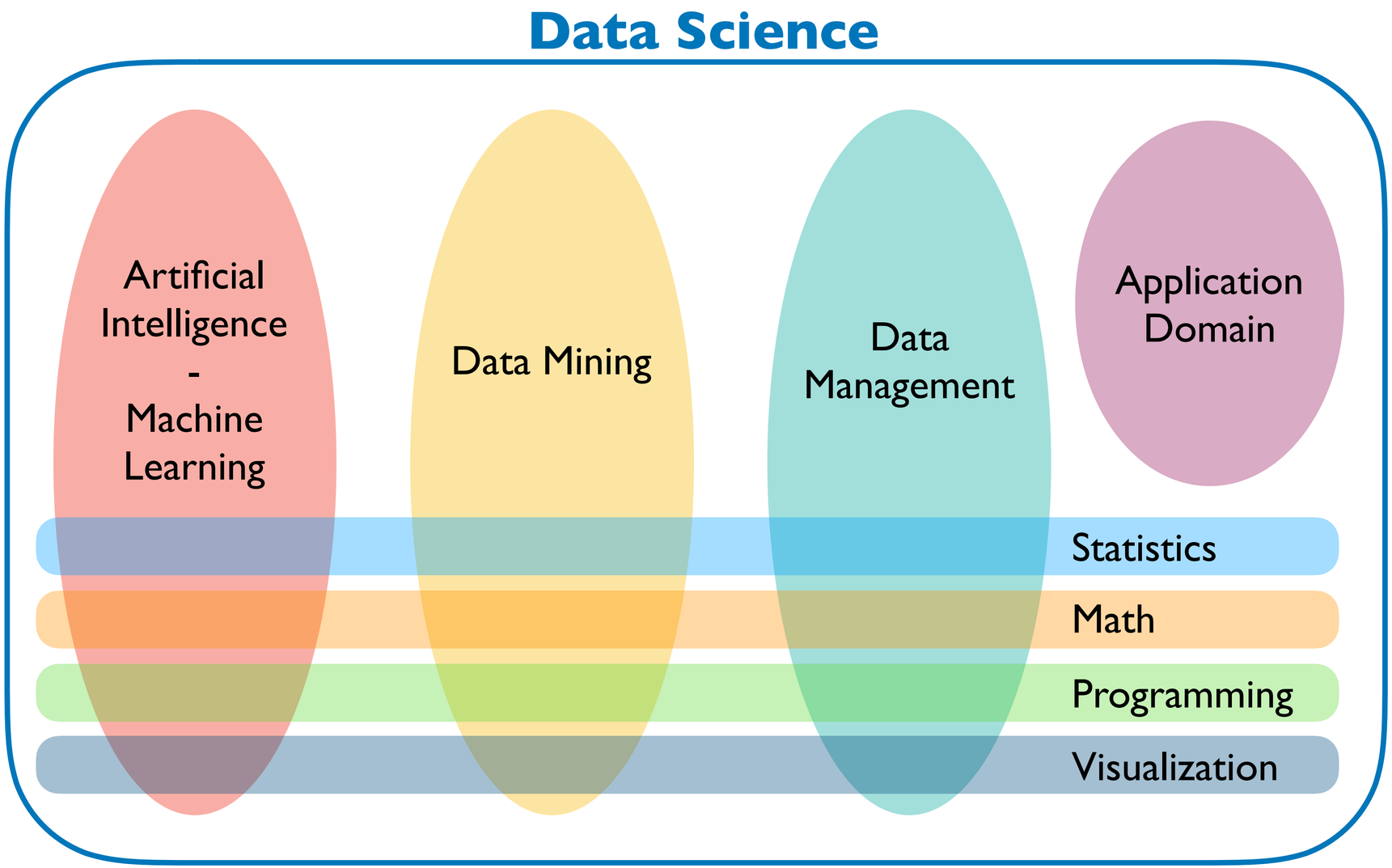

Big Data!
AI!
Data Science!
ML!


Big Data!
AI!
Data Science!
ML!

https://commons.wikimedia.org/wiki/File:House_of_Commons_2010.jpg Open Government Licence v3.0 (OGL v.3).

Data Science
vs
Machine Learning


A.I.
Big Data Management
Application Domain
Data Science
Data Mining
Machine Learning
Statistics

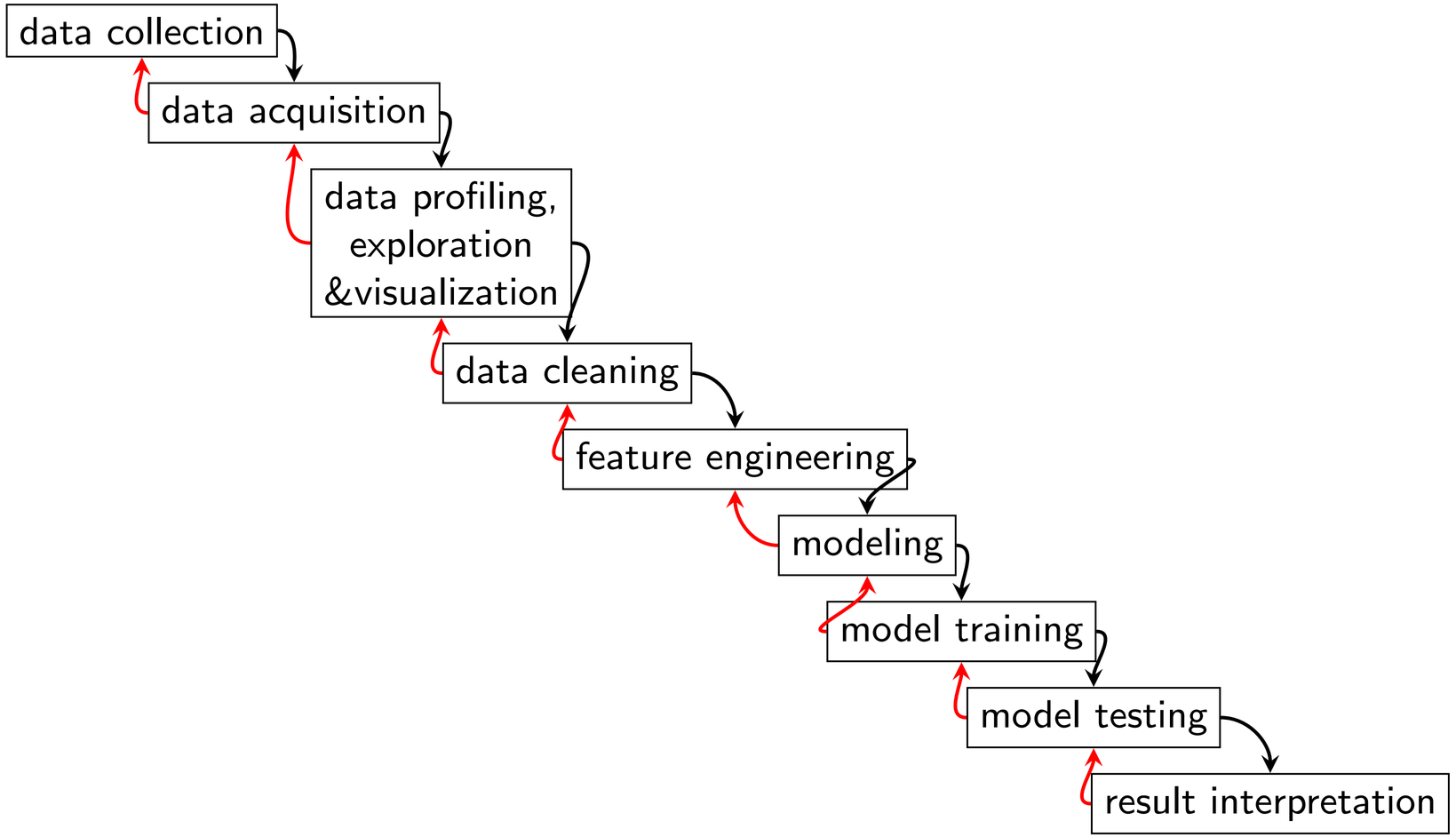

Data Science
Artificial Intelligence - Machine Learning
Data Mining
Data Management
Application Domain
Statistics
Math
Programming
Visualization

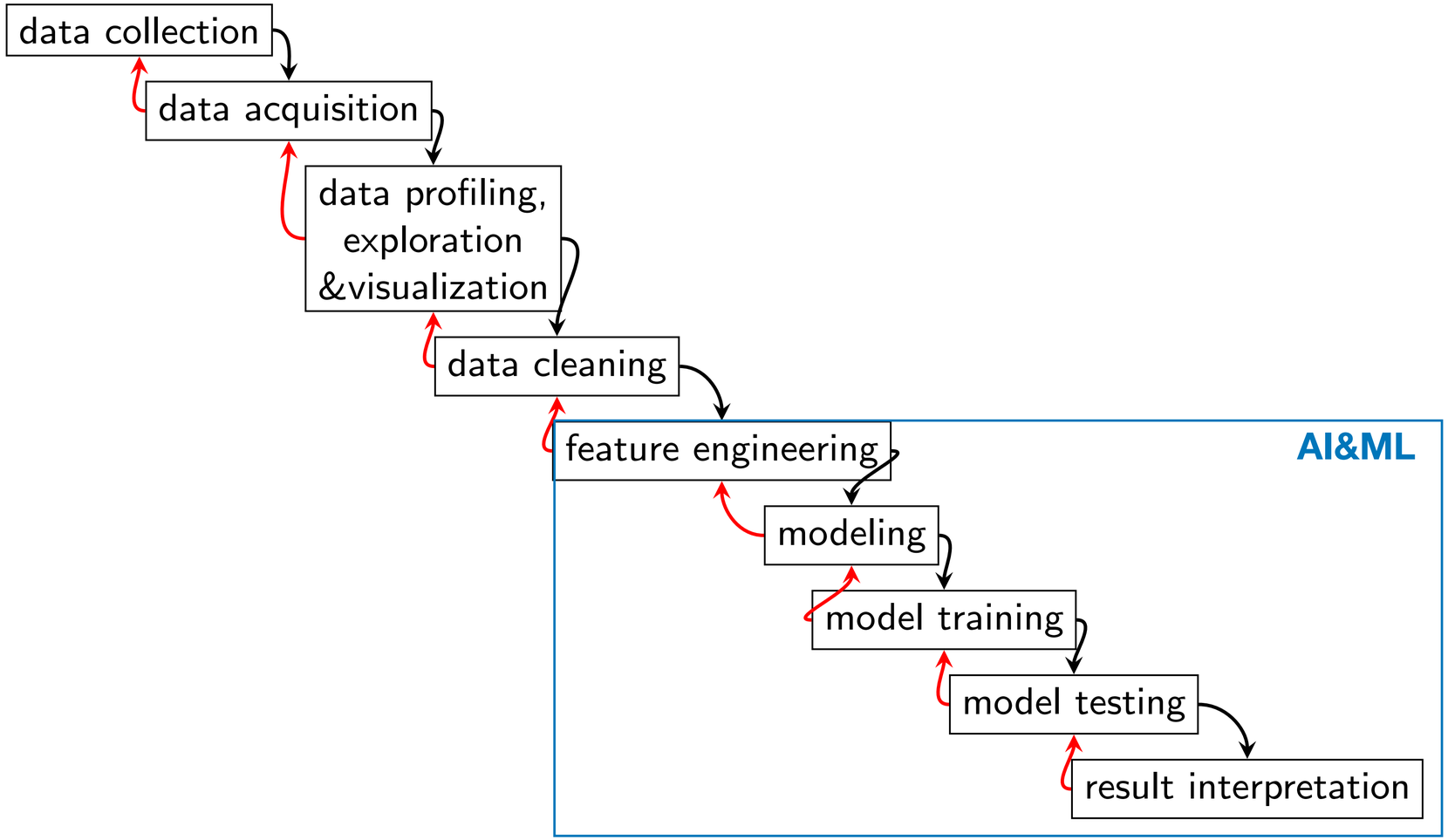
The Data Science Cake

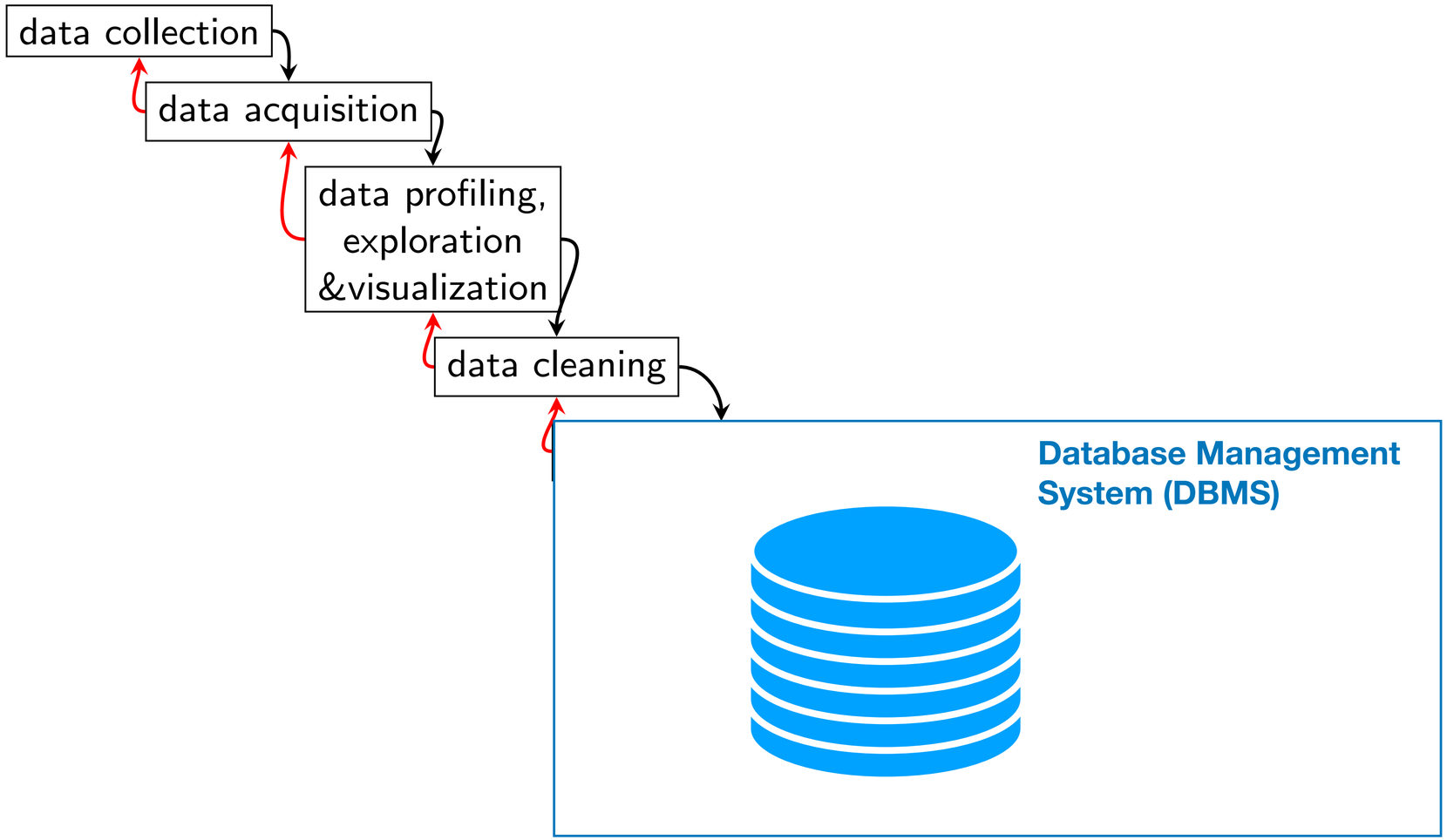
Ingredients:
50g statistics
120g linear algebra
200g programming
1kg visualisation
300g software engineering

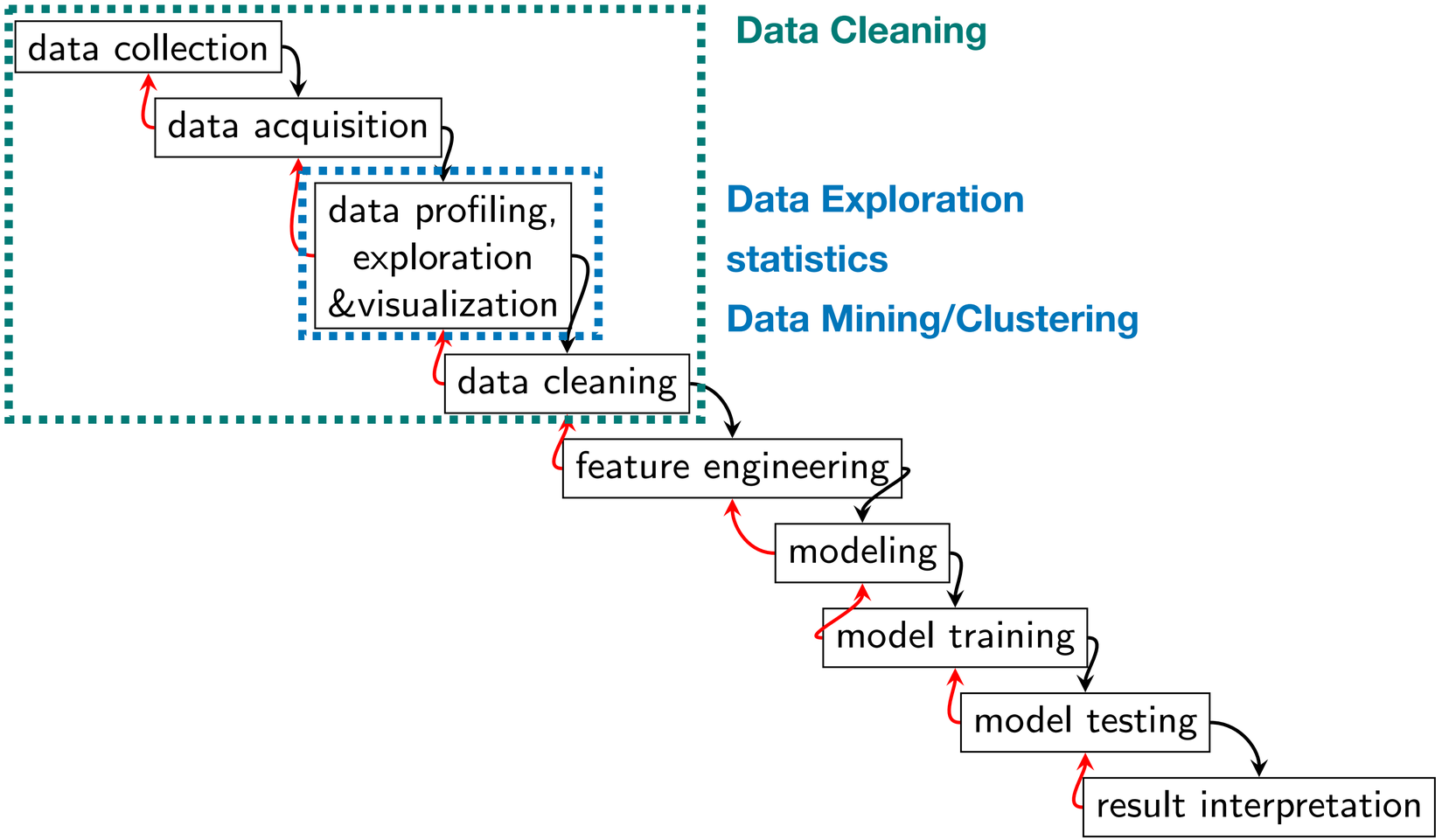
twitter.com/jensdittrich

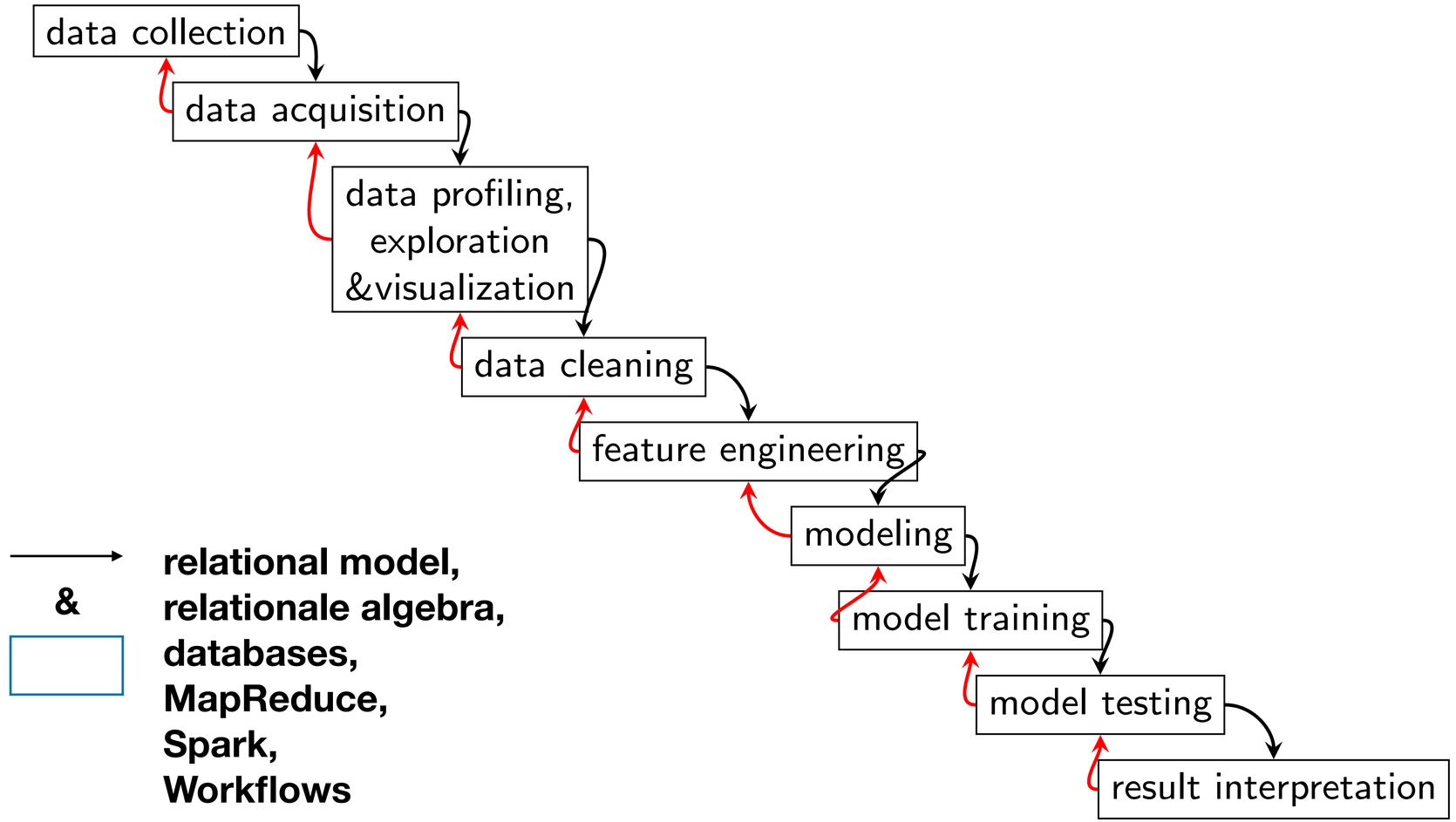
Additional skills:
creativity
out of the box thinking
grit
team spirit

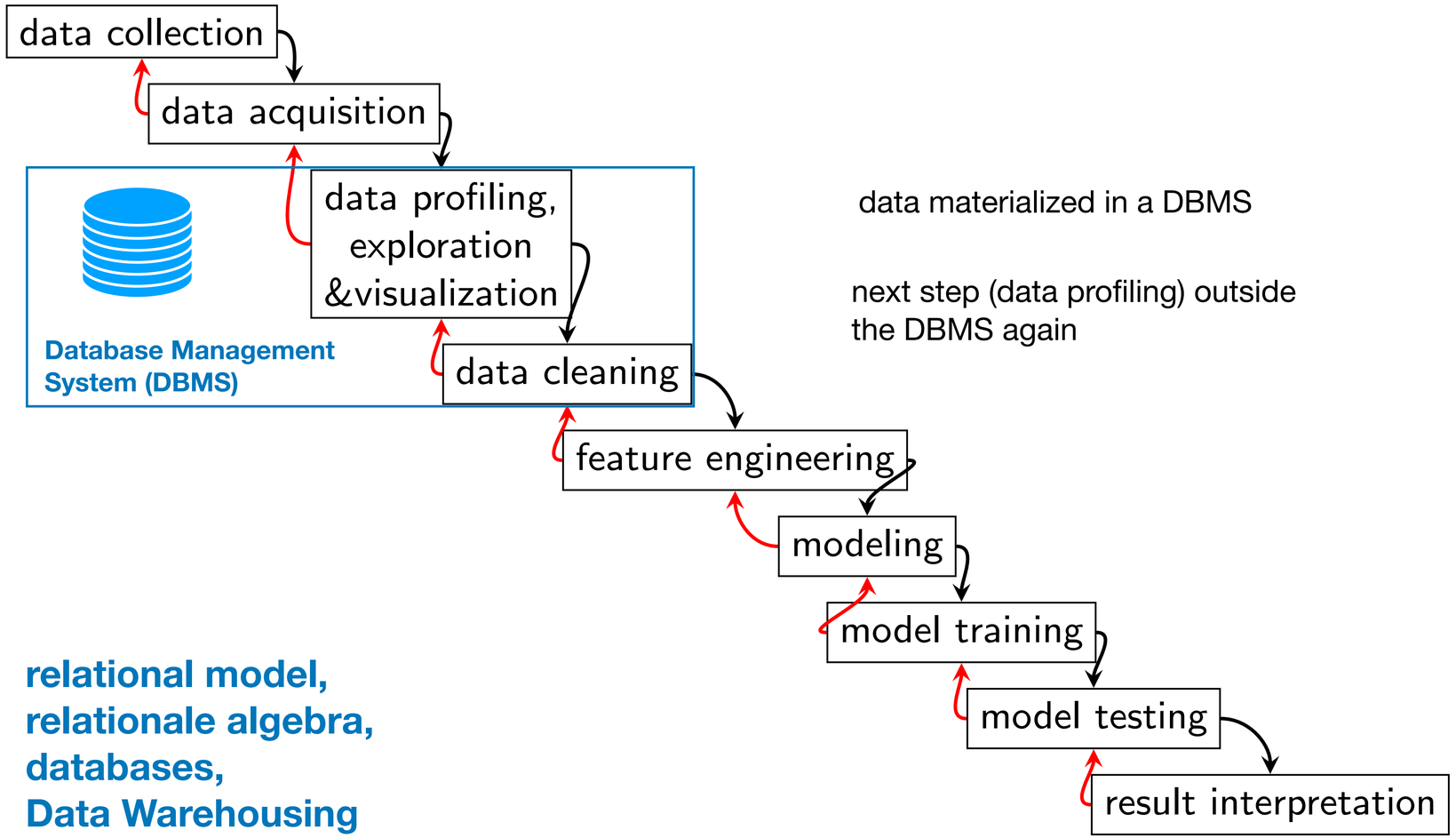

data collection
data acquisition
data profiling,
exploration
&visualization
data cleaning
feature engineering
modeling
model training
model testing
result interpretation


data collection
data acquisition
data profiling, exploration &visualization
data cleaning
feature engineering
modeling
model training
model testing
result interpretation
AI&ML

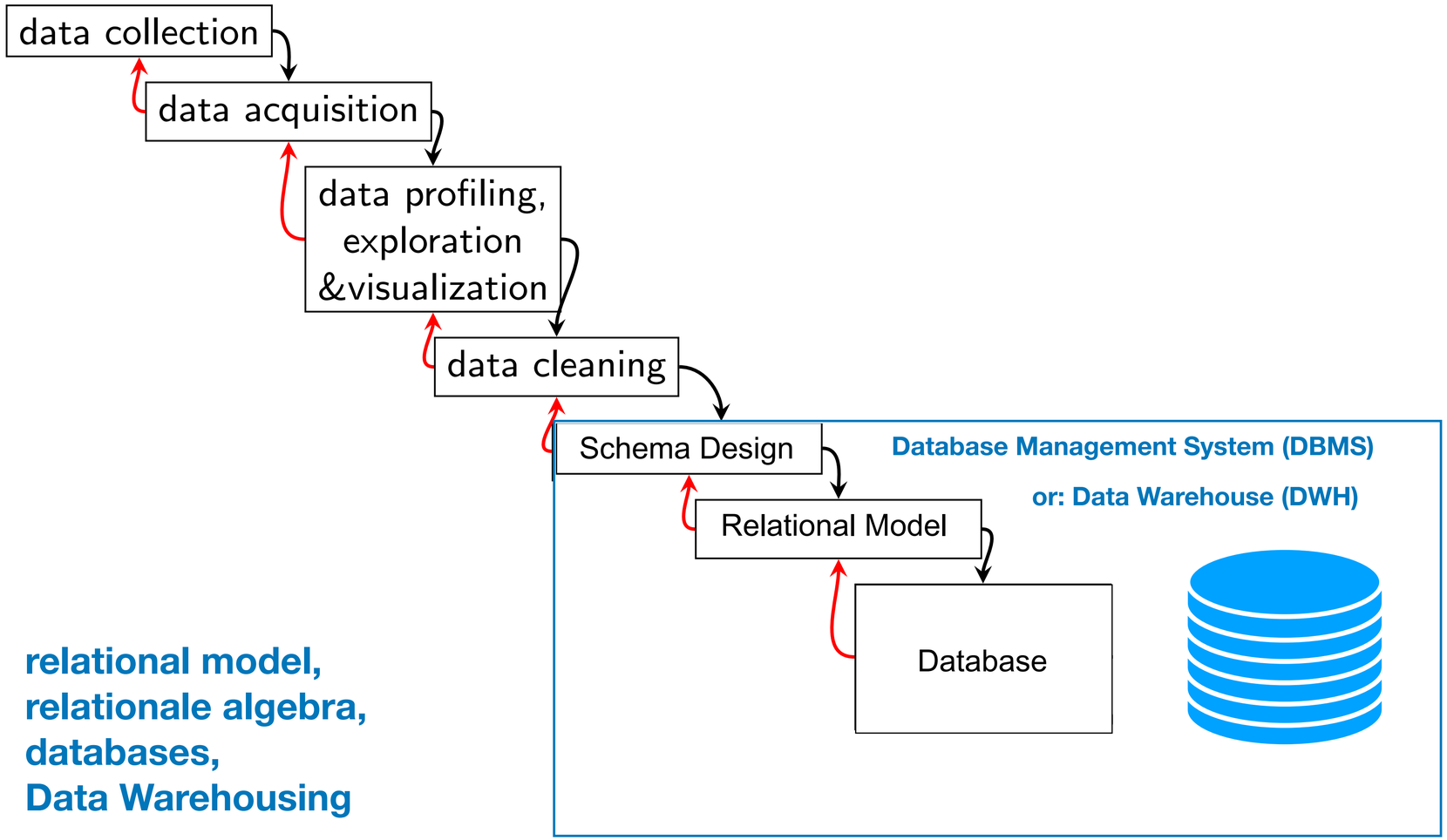

data collection
data acquisition
data profiling,
exploration
&visualization
data cleaning
Database Management
System (DBMS)


data collection
data acquisition
data profiling, exploration &visualization
data cleaning
feature engineering
modeling
model training
model testing
result interpretation
Data Cleaning
Data Exploration
statistics
Data Mining/Clustering

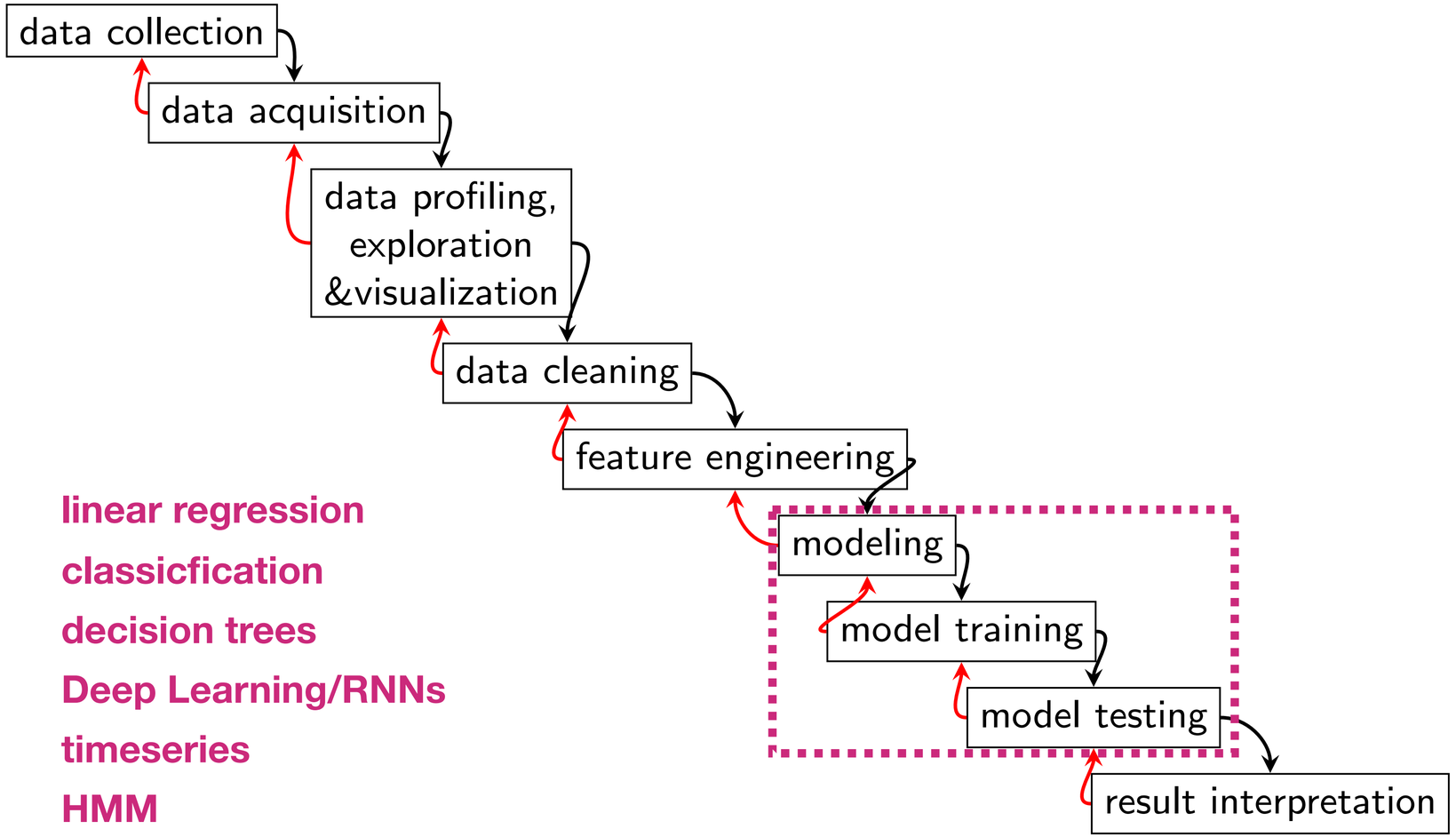

data collection
data acquisition
data profiling,
exploration
&visualization
data cleaning
feature engineering
modeling
model training
model testing
result interpretation
&
relational model,
relationale algebra,
databases,
MapReduce,
Spark,
Workflows



relational model,
relationale algebra,
databases,
Data Warehousing



relational model,
relationale algebra,
databases,
Data Warehousing


data collection
data acquisition
data profiling, exploration &visualization
data cleaning
feature engineering
modeling
model training
model testing
result interpretation
linear regression
classicfication
decision trees
Deep Learning/RNNs
timeseries
HMM

2 Opportunities

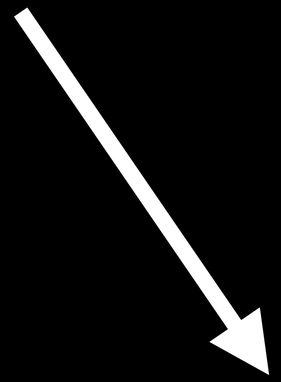
<something> ∩ databases

Software 2.0

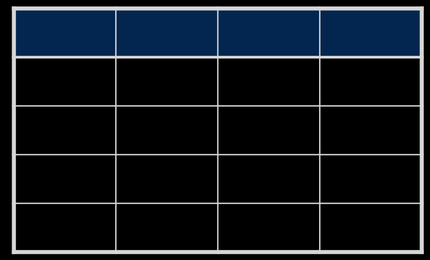
$$f(\quad X \quad) = \quad Y$$

f( ) =

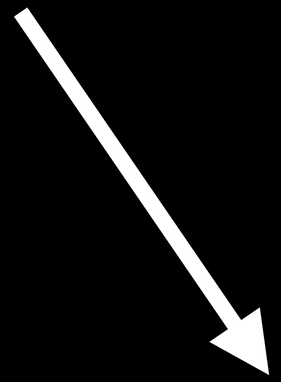
SELECT A FROM…



f( ) =

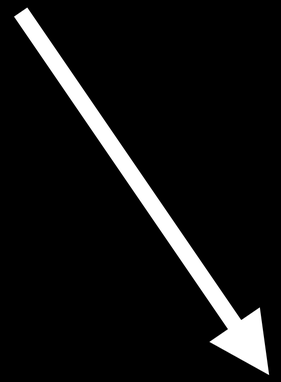


SELECT A FROM…



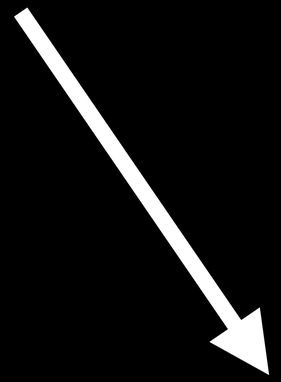
f( ) = LLVM code

SELECT A FROM…

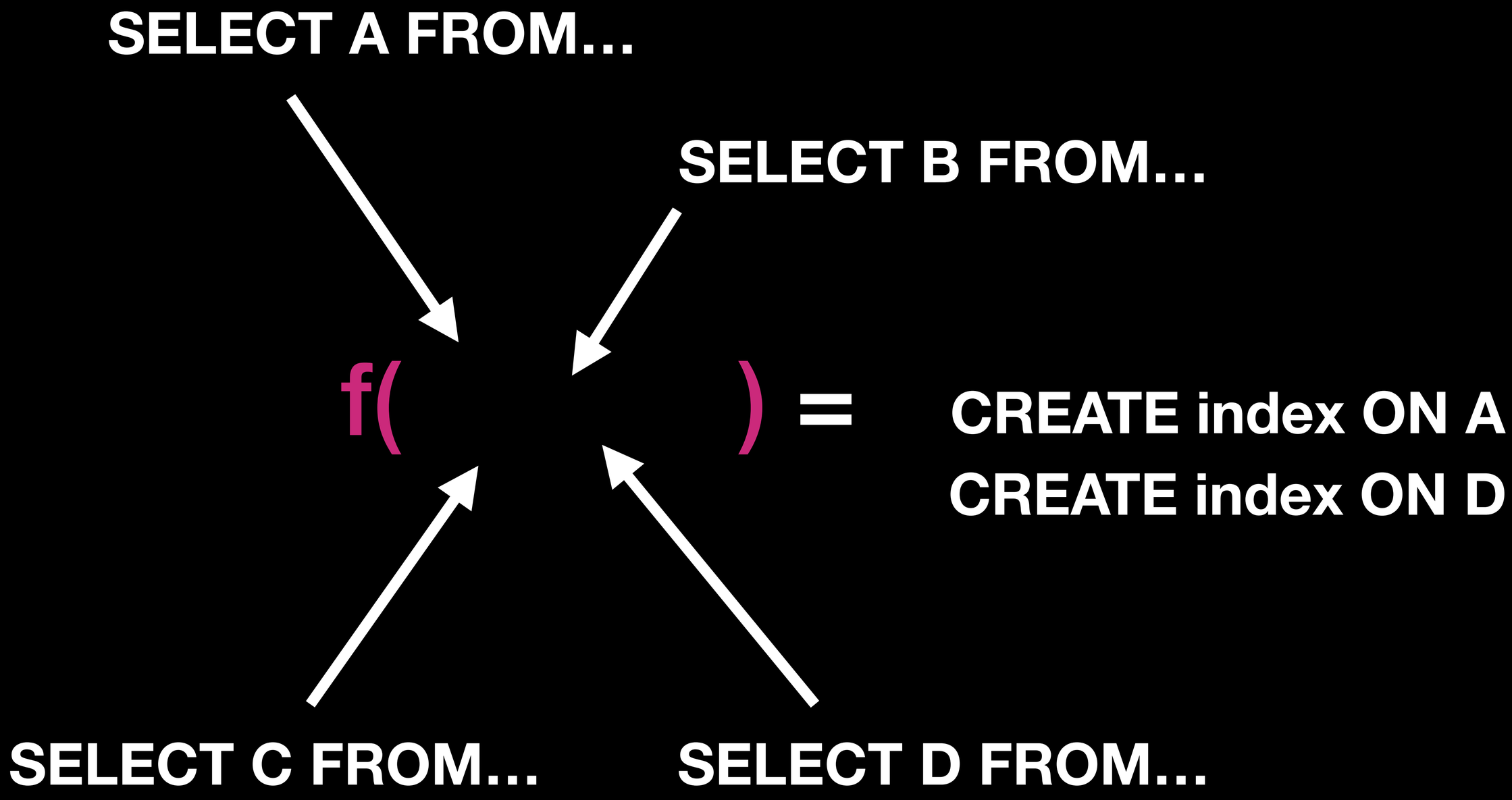


f( ) = ~42 ms

SELECT A FROM...



f( ) = CREATE index ON A


SELECT A FROM…
SELECT B FROM…
f(
) =
CREATE index ON A
CREATE index ON D
SELECT C FROM…
SELECT D FROM…

[VLDB 2017]

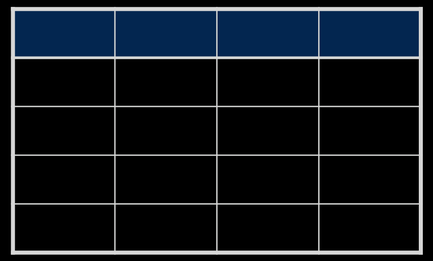
Deep Learning (m)eats Databases

Jens Dittrich

Saarland University
Saarland Informatics Campus
d:AI:mond.ai

$$?(\ X\ )=\ Y$$

(X) = Y



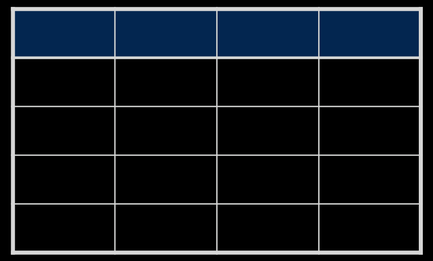
CREATE index ON A

f ( ) =

3

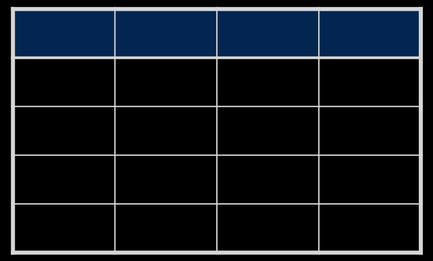
The Case for Learned Indexes

[Kraska et al]



CREATE index ON A

f ( ) =



CREATE index ON A

f ( ) =

{

tensorflow…

code generation…

}

[Kraska et al]

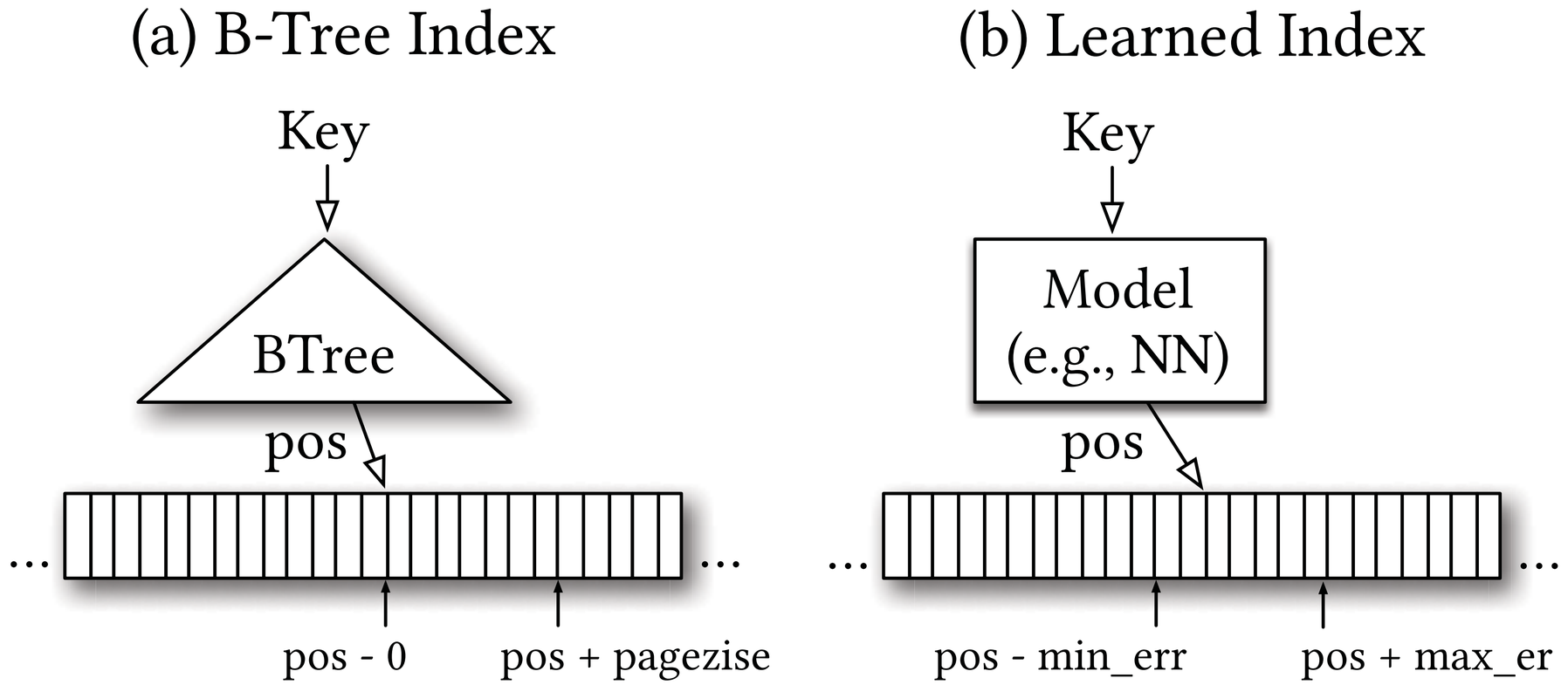


Figure 1: Why B-Trees are models

[Kraska et al]

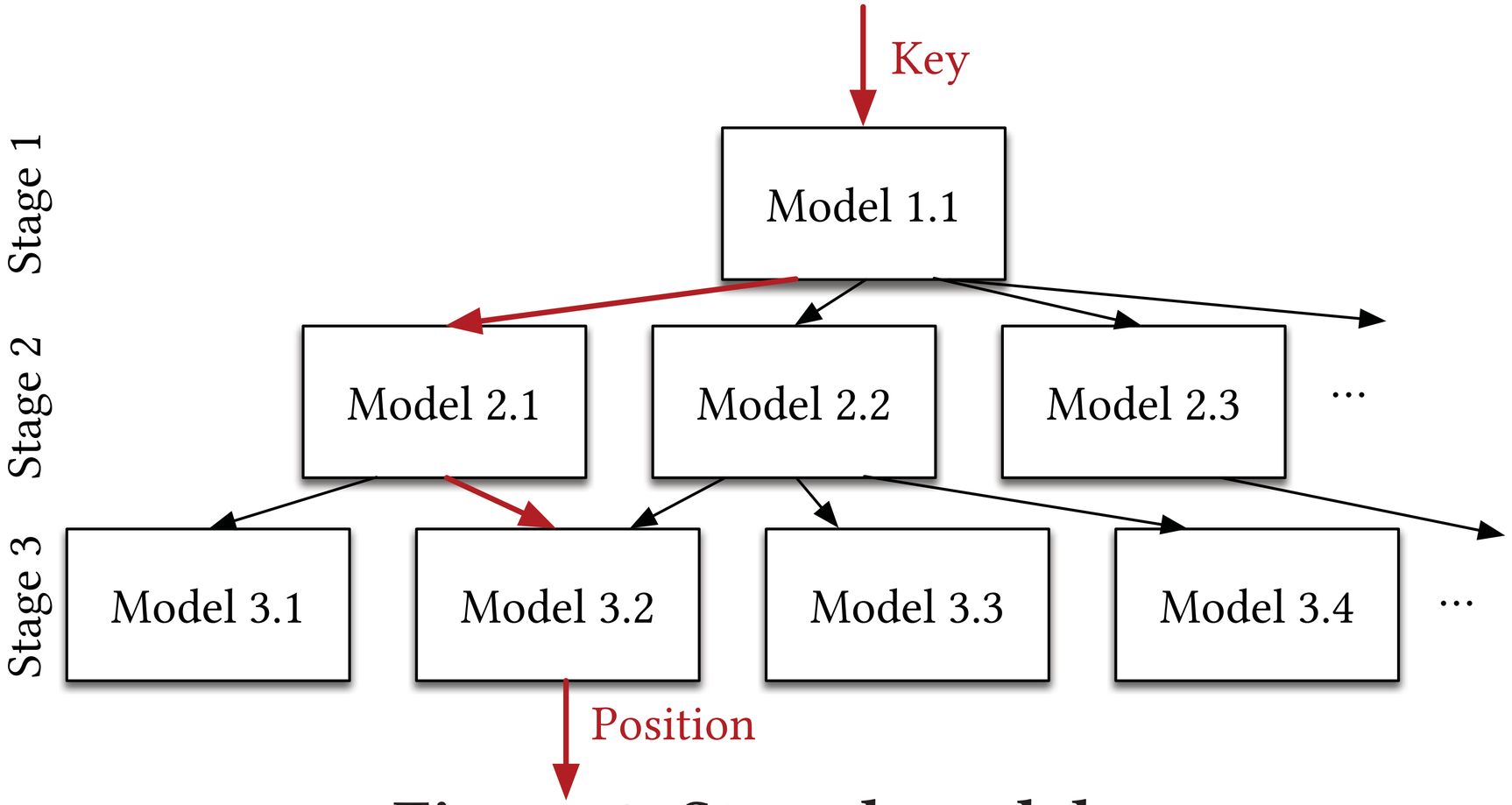


Figure 3: Staged models

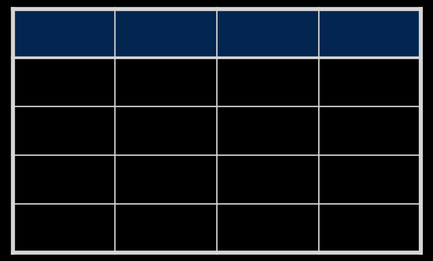
[Kraska et al]

		Map Data			Web Data			Log-Normal Data		
Type	**Config**	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)
Btree	page size: 32	52.45 (4.00x)	274 (0.97x)	198 (72.3%)	51.93 (4.00x)	276 (0.94x)	201 (72.7%)	49.83 (4.00x)	274 (0.96x)	198 (72.1%)
	page size: 64	26.23 (2.00x)	277 (0.96x)	172 (62.0%)	25.97 (2.00x)	274 (0.95x)	171 (62.4%)	24.92 (2.00x)	274 (0.96x)	169 (61.7%)
	page size: 128	13.11 (1.00x)	265 (1.00x)	134 (50.8%)	12.98 (1.00x)	260 (1.00x)	132 (50.8%)	12.46 (1.00x)	263 (1.00x)	131 (50.0%)
	page size: 256	6.56 (0.50x)	267 (0.99x)	114 (42.7%)	6.49 (0.50x)	266 (0.98x)	114 (42.9%)	6.23 (0.50x)	271 (0.97x)	117 (43.2%)
	page size: 512	3.28 (0.25x)	286 (0.93x)	101 (35.3%)	3.25 (0.25x)	291 (0.89x)	100 (34.3%)	3.11 (0.25x)	293 (0.90x)	101 (34.5%)
Learned Index	2nd stage models: 10k	0.15 (0.01x)	98 (2.70x)	31 (31.6%)	0.15 (0.01x)	222 (1.17x)	29 (13.1%)	0.15 (0.01x)	178 (1.47x)	26 (14.6%)
	2nd stage models: 50k	0.76 (0.06x)	85 (3.11x)	39 (45.9%)	0.76 (0.06x)	162 (1.60x)	36 (22.2%)	0.76 (0.06x)	162 (1.62x)	35 (21.6%)
	2nd stage models: 100k	1.53 (0.12x)	82 (3.21x)	41 (50.2%)	1.53 (0.12x)	144 (1.81x)	39 (26.9%)	1.53 (0.12x)	152 (1.73x)	36 (23.7%)
	2nd stage models: 200k	3.05 (0.23x)	86 (3.08x)	50 (58.1%)	3.05 (0.24x)	126 (2.07x)	41 (32.5%)	3.05 (0.24x)	146 (1.79x)	40 (27.6%)

Figure 4: Learned Index vs B-Tree

3(a) A Criticism of The Case for Learned Indexes

3(a) A positive Criticism of The Case for Learned Indexes

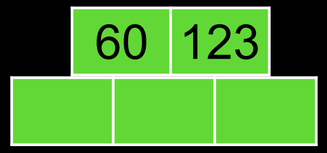


CREATE index ON A

f ( ) =

3(b) A constructive Criticism of The Case for Learned Indexes

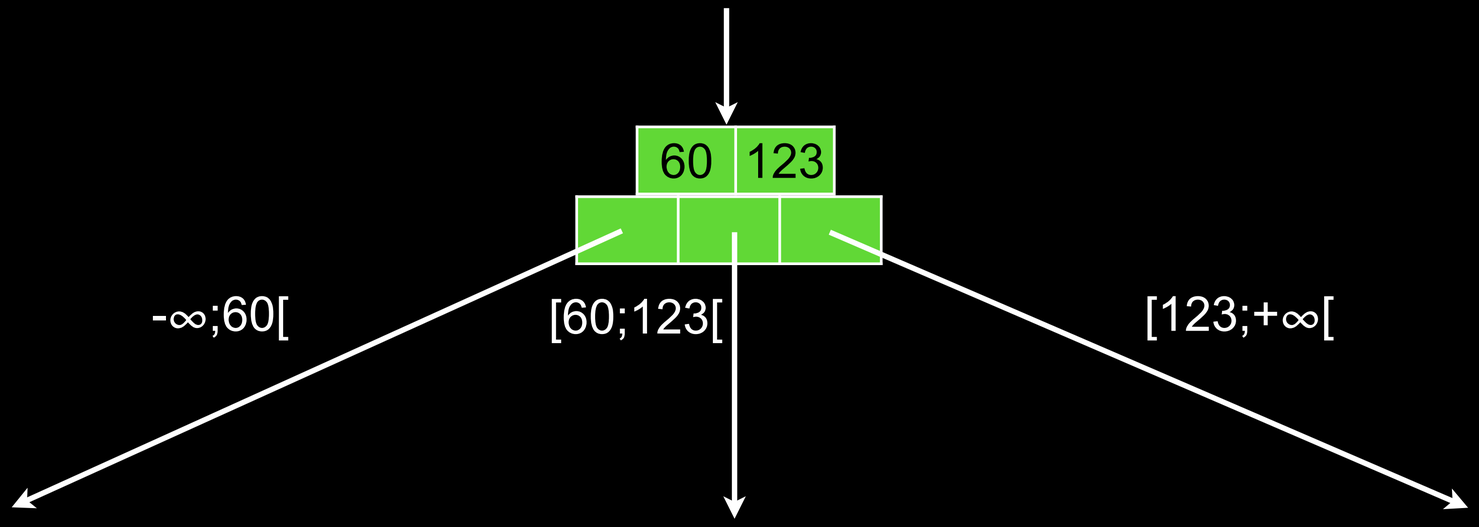
Coarse-granular and Sparse B-Trees



k = 1

k* = 1

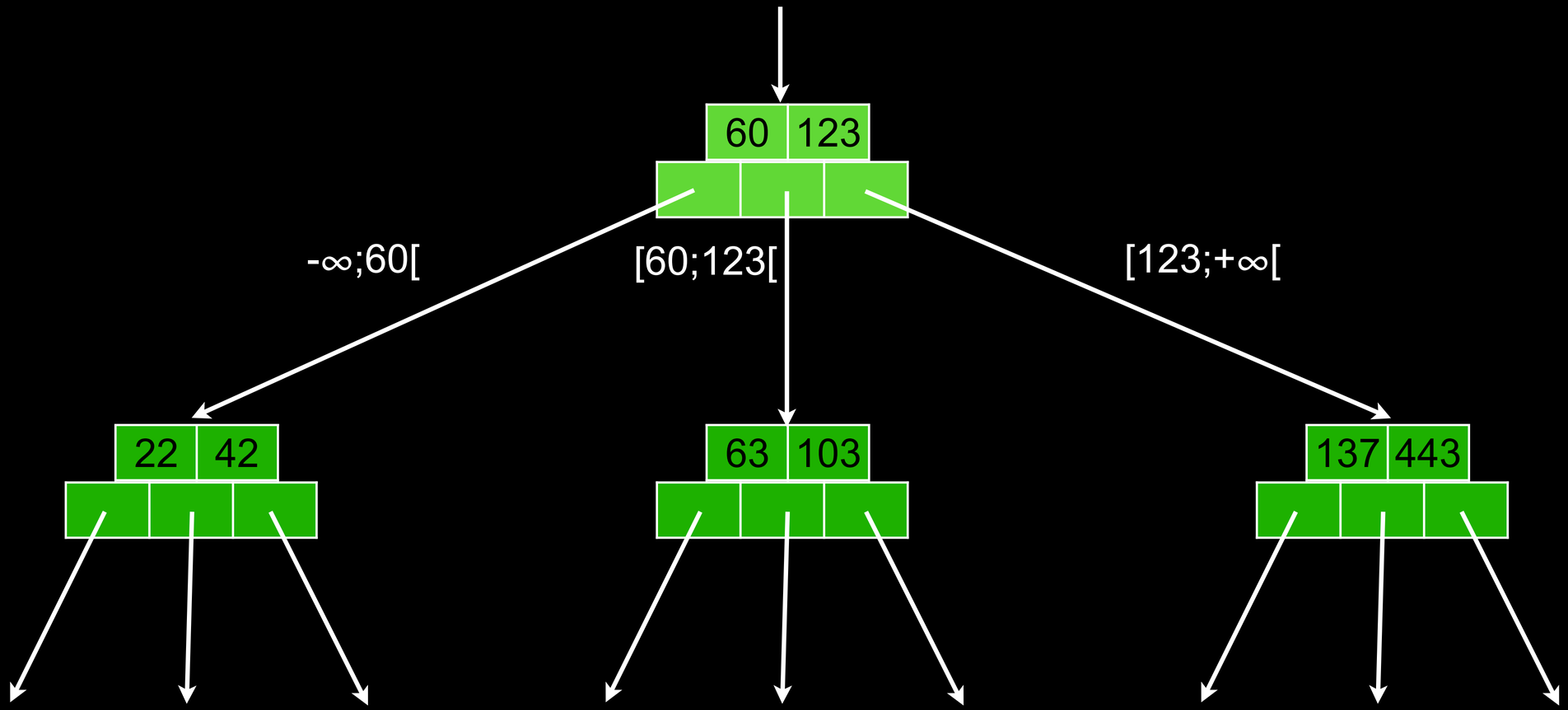
Coarse-granular and Sparse B-Trees



k = 1

k* = 1

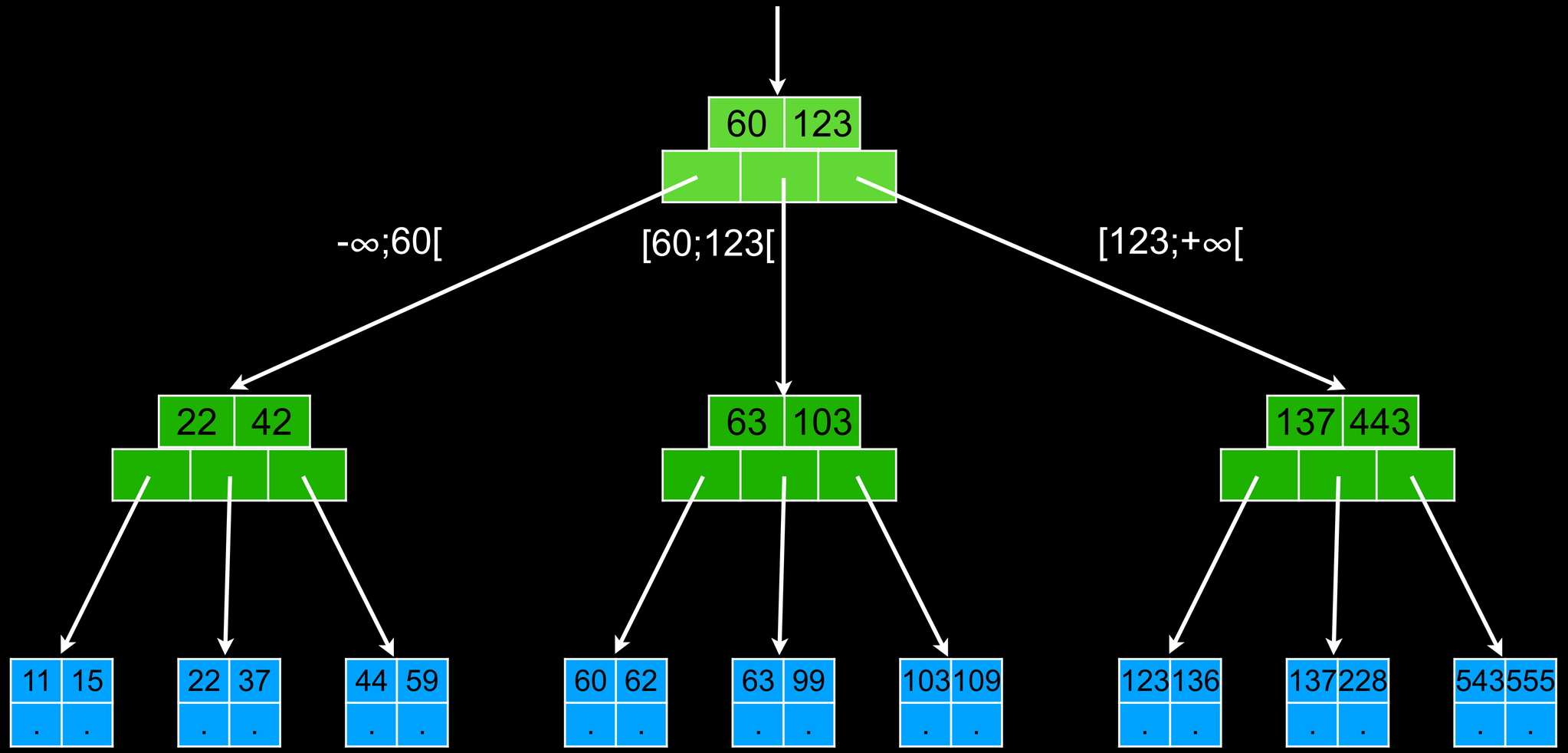
Coarse-granular and Sparse B-Trees



k = 1

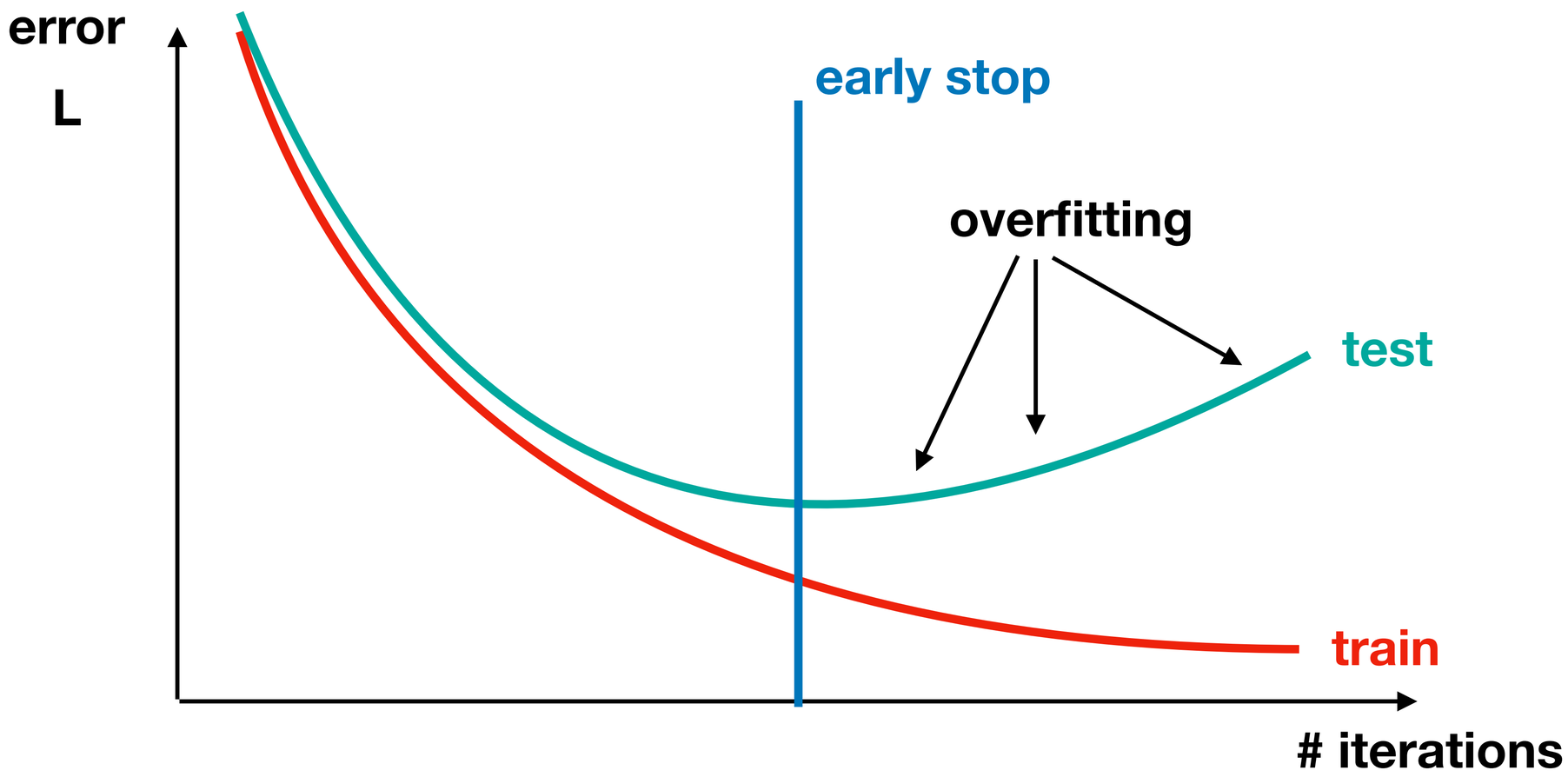
k* = 1

Coarse-granular and Sparse B-Trees



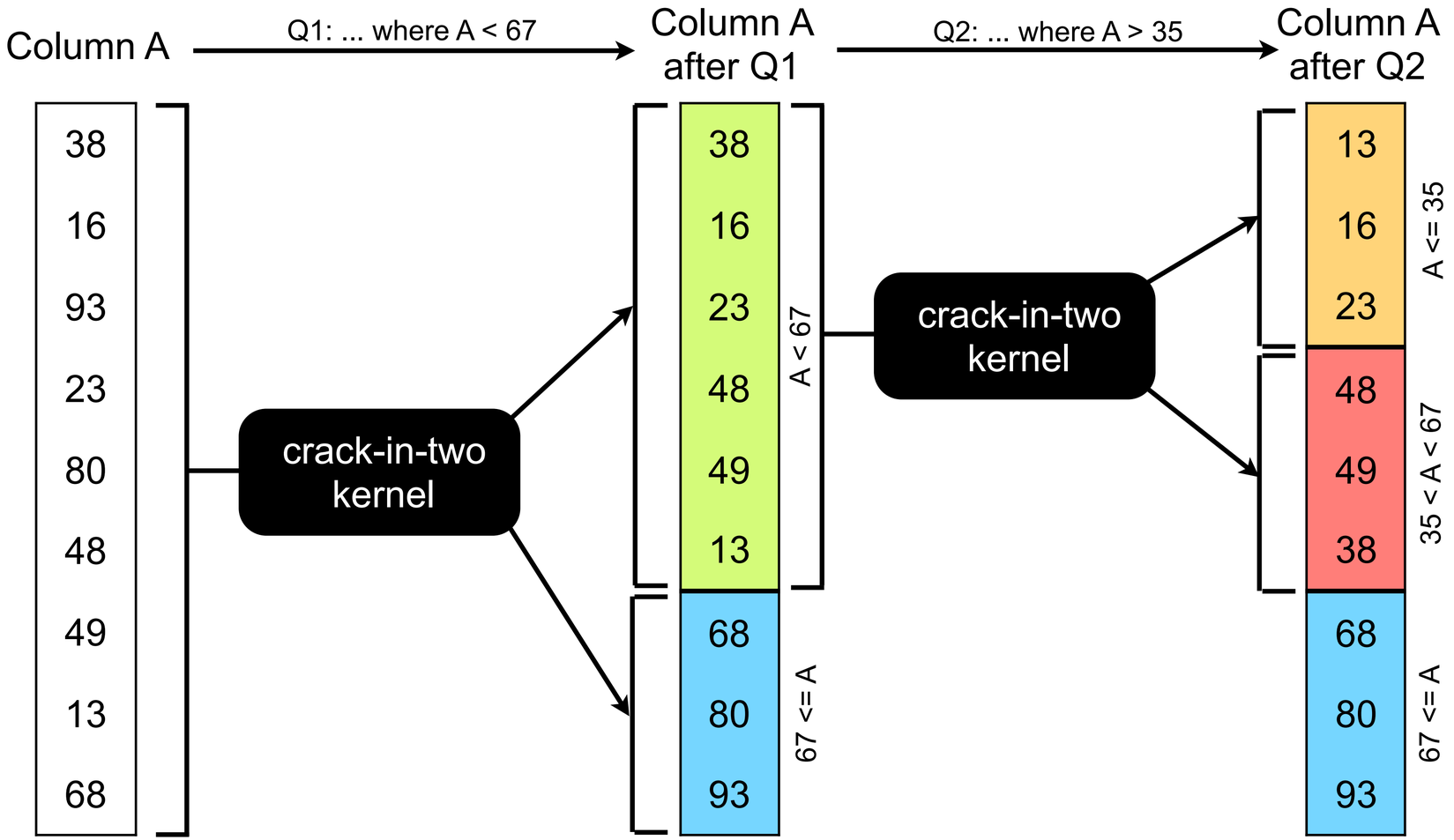
k = 1

k* = 1

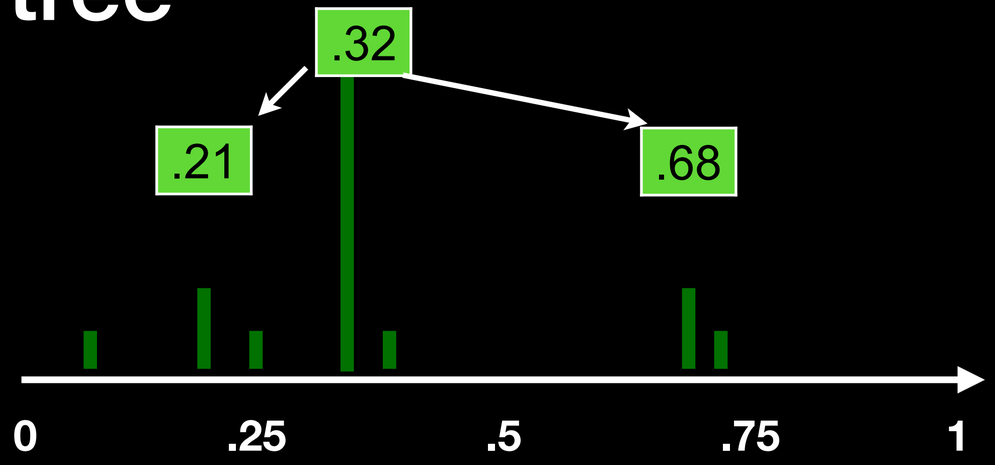

error
L
early stop
overfitting
test
train
iterations

AI

Adaptive Indexing

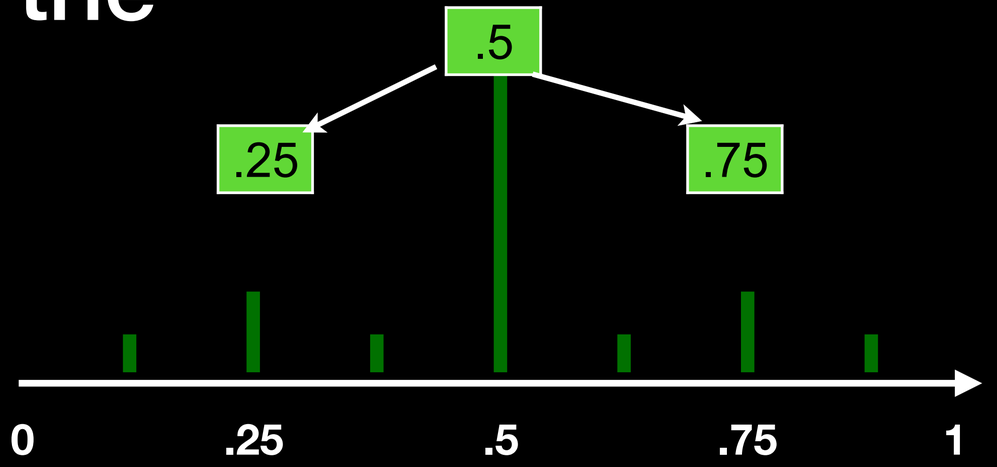

Column A
Q1: ... where A < 67
Column A after Q1
Q2: ... where A > 35
Column A after Q2
38
16
93
23
80
48
49
13
68
crack-in-two kernel
38
16
23
48
49
13
A < 67
68
80
93
67 <= A
crack-in-two kernel
13
16
23
A <= 35
48
49
38
35 < A < 67
68
80
93
67 <= A

tree



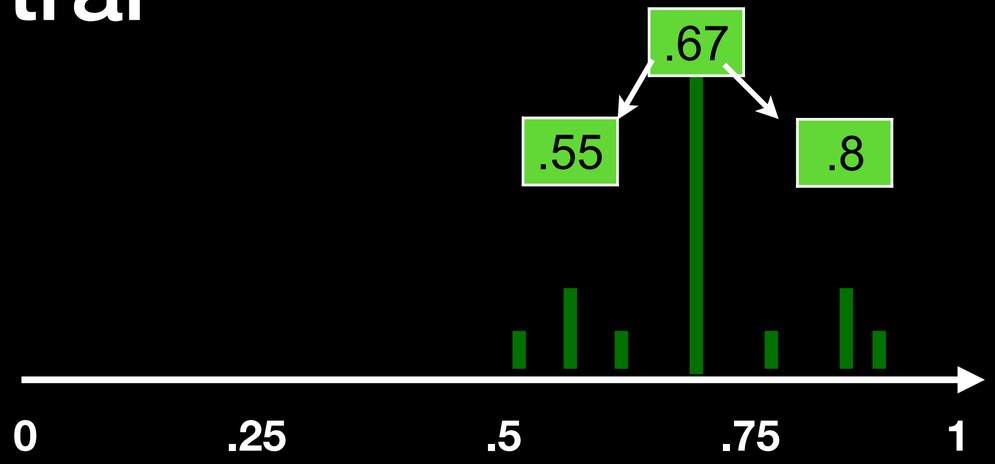
partition the data

trie



partition the dataspace

trai



partition the predicates

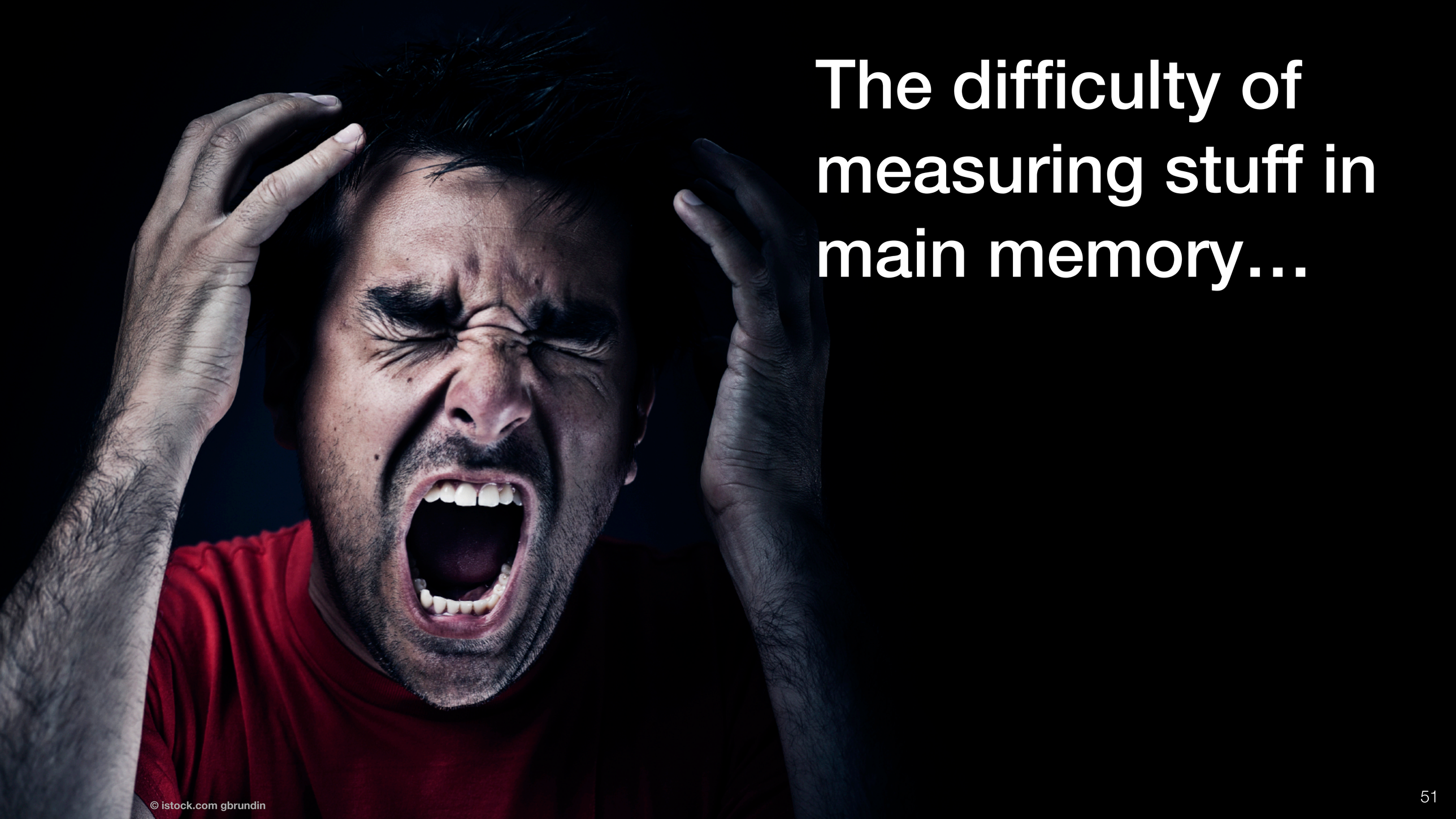
tria

~same as trie

partition the predicate space

Adaptive Indexing

learns the search predicates



The difficulty of measuring stuff in main memory…

© istock.com Maksud_kr

(still) some doubts on practicality of this approach, and diff to related work.

a nice fresh view on indexing, nice observations, food for thought

4 experience from teaching Data Science

Materialien

1. Vorlesungen

- 01a Einführung (1.9 MB)
- 01b Einführung (464 KB)
- 02 Bug Data Analytics (1.4 MB)
- 03a Relationales Modell und Algebra (1.9 MB)
- 03b Zusätzliche Beispiele für abgeleitete Operatoren (159 KB)
- 04 Datenbanken (276 KB)
- 05 Data Exploration (2.3 MB)
- 06 Statistik (26 MB)
- 07 MapReduce & Apache Spark (1.1 MB)
- 08 Lineare Regression (2.7 MB)
- 09 Klassifizierung (4.9 MB)
- 10 Entscheidungsbäume (672 KB)
- 11 Neural Networks (5.2 MB)
- 12 Clustering (4.5 MB)

2. Notebooks

- 02 Data Cleaning (137 KB)
- 03 Relationale Algebra (12 KB)
- 04 SQL (19 KB)
- 07 Spark (13 KB)
- 10 Entscheidungsbäume (6.0 KB)
- 11 Neural Networks (2.1 MB)

Materialien | Home | 02 Data Cleaning

localhost:8888/notebooks/notebooks/02 Data Cleaning.ipynb

jupyter 02 Data Cleaning Last Checkpoint: 12.04.2018 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

In [30]:
```
# bedingte Einfärbung der Zellen des DataFrames:
linksPivot.style.apply(lambda x: ["background: orange" if v >= 42 else "background: lightgreen" if v > 0 else "" for v
```

Out[30]:

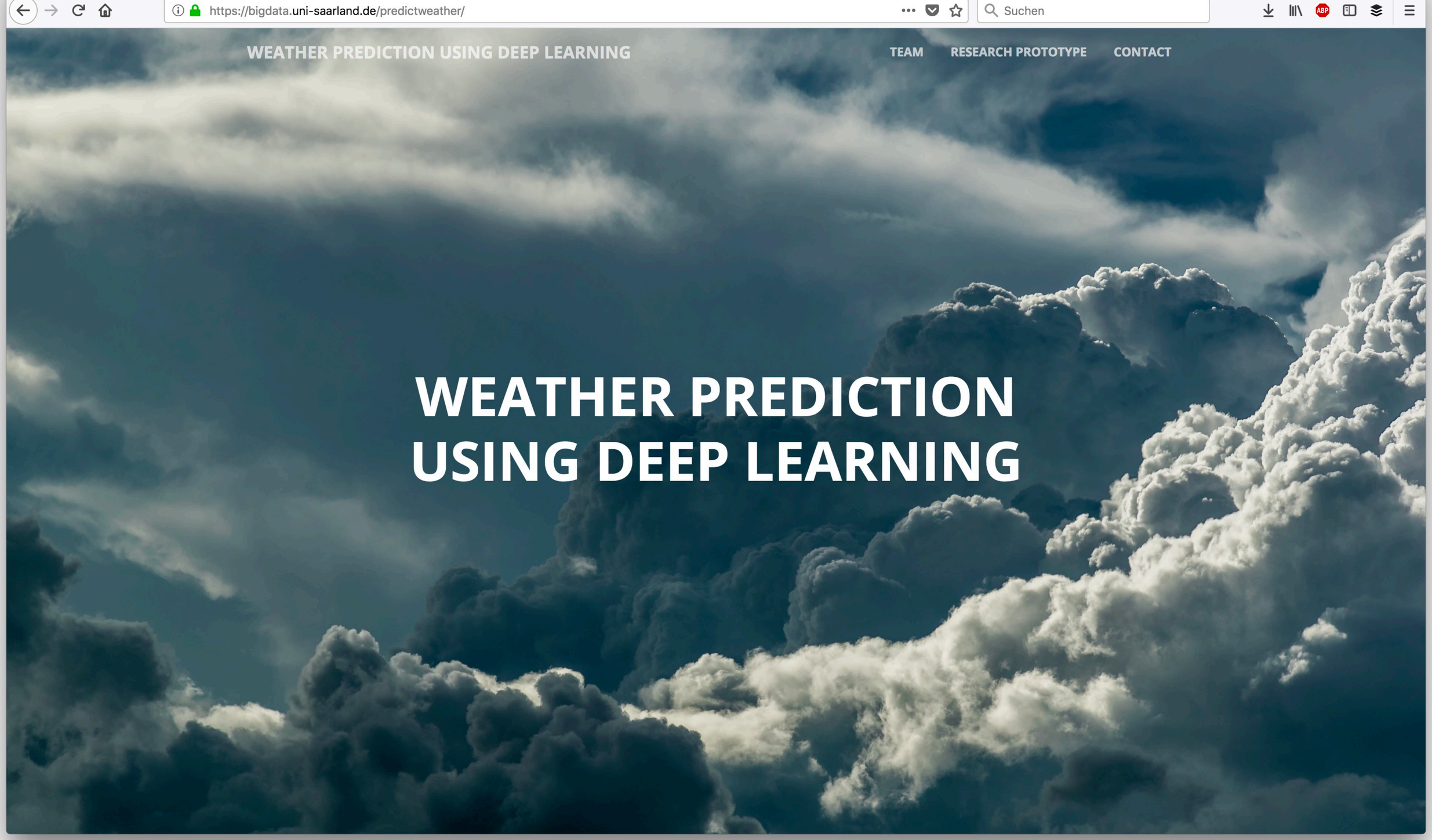
target	Action	Adventure	Animation	Biography	Comedy	Crime	Drama	Family	Fantasy	History	Horror	Music	Musical	Mystery	Romance	Sci-Fi	Sport
source																	
Action	2	155	10	11	45	54	72	7	40	4	19	0	0	18	7	74	4
Adventure	0	1	38	8	59	8	52	27	52	1	7	0	0	10	6	59	0
Animation	0	0	0	0	34	0	3	8	3	0	0	0	0	0	1	0	0
Biography	0	0	0	0	8	12	74	1	0	15	0	0	0	1	5	0	8
Comedy	0	0	0	0	32	26	100	19	13	0	14	8	2	2	71	5	2
Crime	0	0	0	0	0	0	97	0	2	2	7	1	0	29	0	1	1
Drama	0	0	0	0	0	0	48	13	32	28	35	13	4	52	98	28	15
Family	0	0	0	0	0	0	0	0	17	0	0	1	1	0	1	0	0
Fantasy	0	0	0	0	0	0	0	0	0	0	15	1	1	3	6	0	0
History	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Horror	0	0	0	0	0	0	0	0	0	0	11	0	1	22	2	16	0
Music	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0
Musical	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Mystery	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	0
Romance	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	1
Sci-Fi	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
Thriller	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In [31]:
```
# die obige Visualisierung führt auf natürliche Weise zu einer Heatmap:
plt.subplots(figsize=(15,15))
sns.heatmap(linksPivot, cmap=sns.color_palette("Greens", 42),annot=True)
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1128ddb38>

5

experiences from solving real data science problems with domain experts

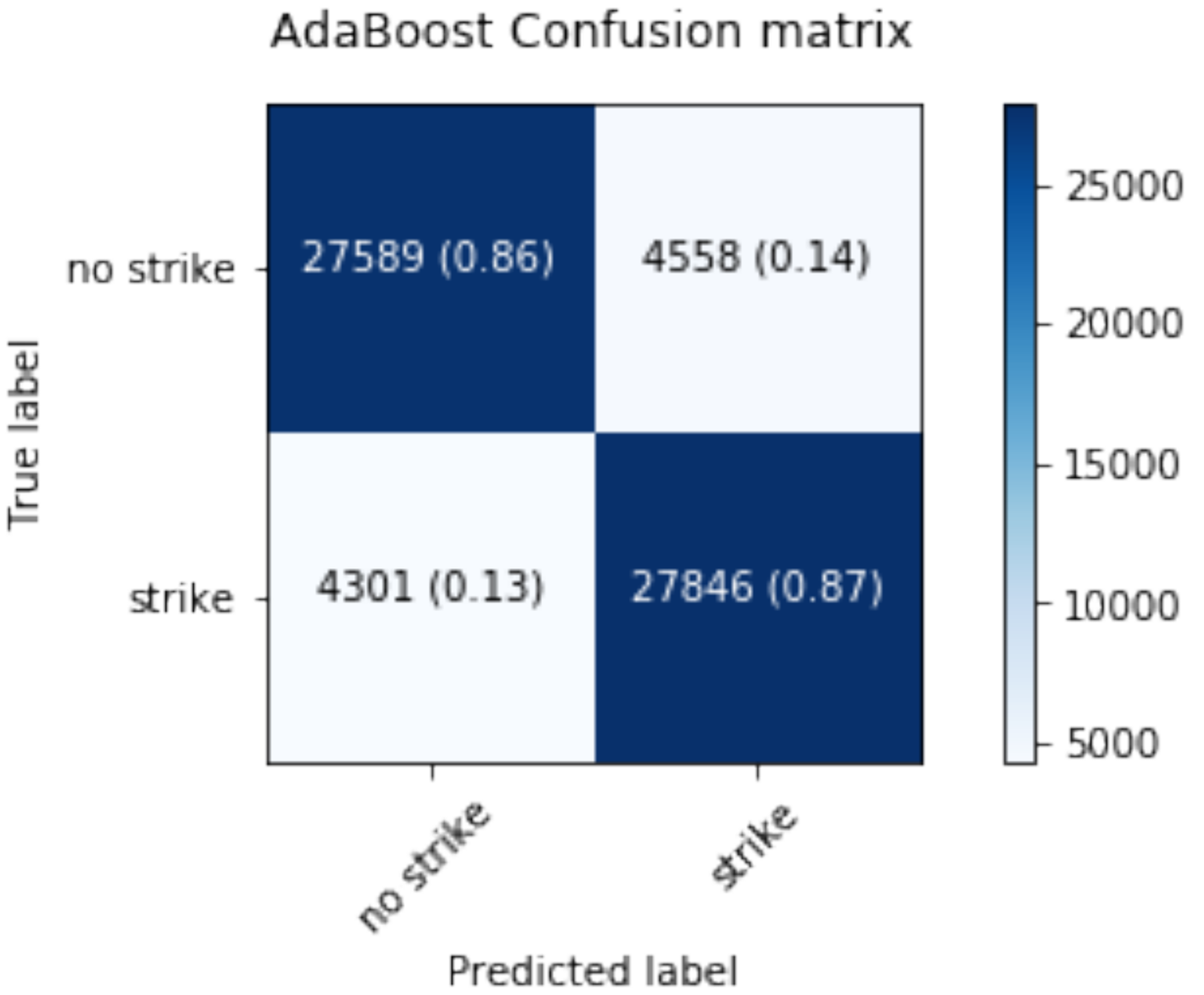

https://bigdata.uni-saarland.de/predictweather/
Suchen
WEATHER PREDICTION USING DEEP LEARNING
TEAM
RESEARCH PROTOTYPE
CONTACT
WEATHER PREDICTION
USING DEEP LEARNING

idea: the bug is the feature

2D nowcasting error

~

lightning prediction


AdaBoost Confusion matrix
27589 (0.86)
4558 (0.14)
4301 (0.13)
27846 (0.87)
no strike
strike
True label
no strike
strike
Predicted label
25000
20000
15000
10000
5000

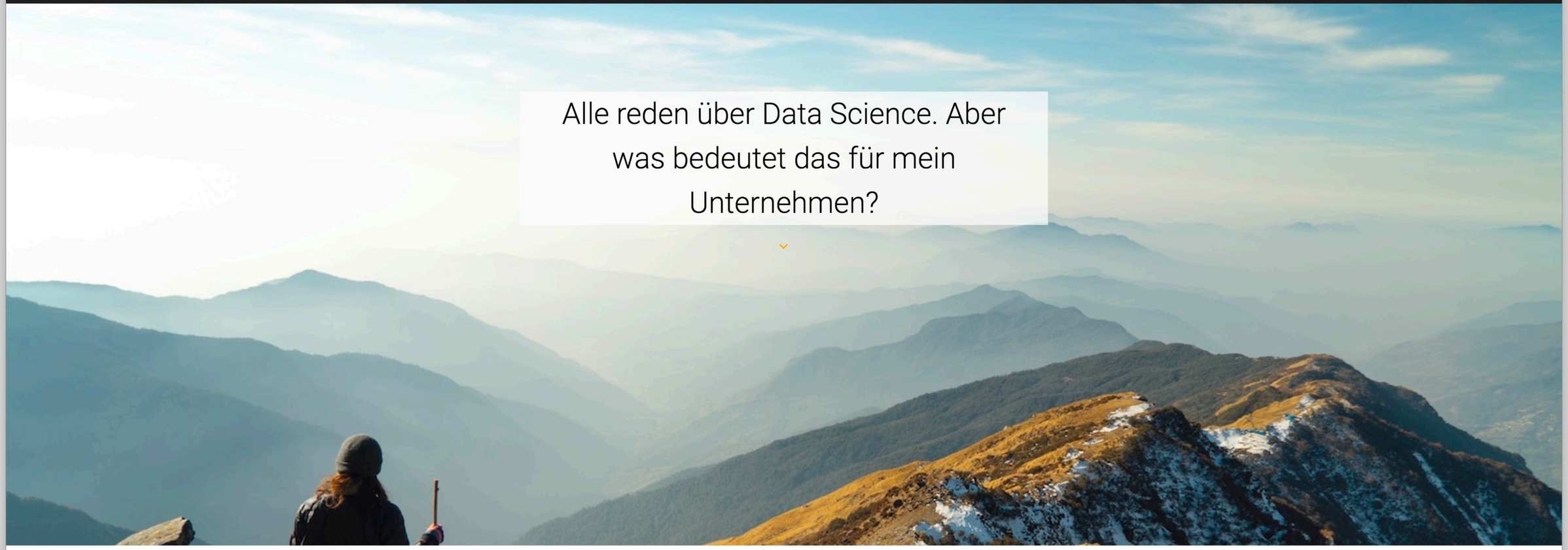
d:AI:mond - data science consulting

daimond.ai

Suchen

D:AI:MOND DATA SCIENCE CONSULTING

Potenzialanalyse Projektrealisierung Team Kontakt Blog



BRINGEN SIE DATA SCIENCE IN IHR UNTERNEHMEN

Als Data Science Consulting helfen wir Ihnen Mehrwerte aus ihren Daten zu schöpfen und dadurch mehr Effizienz, Transparenz und Struktur in ihr Unternehmen zu bringen. Dadurch können Sie sich Wettbewerbsvorteile sichern, Produktionsketten effizienter gestalten, neue

executive summary

1.
remind people that we are one third of what people mean when they say “data science”

2.
We should investigate Software 2.0.
In particular, when f(X) is a function currently implemented by humans.

3.
We need to support and push programs and lectures in data science

4.
get out of your LAB and solve some real problems