Prioritizing Attention in Fast Data

Principles and Promise

Peter Bailis
Edward Gan
Kexin Rong
Sahaana Suri

CIDR 2017

Edward

Kexin

Sahaana

STANFORD INFOLAB

future data

Edward

Kexin

Sahaana

Firas

Deepak

STANFORD INFOLAB

future data

John

Matei

Tony

Edward

Kexin

Sahaana

Firas

Deepak

John

Matei

STANFORD INFOLAB

future data

Ihab

Xu

Sam

Lei

Tony

abundant data, scarce attention

abundant data, scarce attention

data is increasingly too big for manual inspection

abundant data, scarce attention

data is increasingly too big for manual inspection

- Twitter, LinkedIn, Facebook, Google: log 12M+ events/s
- projected 40% year-over-year growth (e.g., via IoT)

abundant data, scarce attention

data is increasingly too big for manual inspection

Twitter, LinkedIn, Facebook, Google: log 12M+ events/s

projected 40% year-over-year growth (e.g., via IoT)

today’s operators say:

< 6% of this data is ever accessed after ingest

abundant data, scarce attention

data is increasingly too big for manual inspection

- Twitter, LinkedIn, Facebook, Google: log 12M+ events/s
- projected 40% year-over-year growth (e.g., via IoT)

today's operators say:

< 6% of this data is ever accessed after ingest

call this trend "fast data"

6%

abundant data, scarce attention

e.g., telemetry and metrics from 100k-MM devices

is the application behaving as expected?

idea: use a classifier to filter data

idea: use a classifier to filter data

too much data?

filter the stream for “interesting”/useful data

basic classifier: static rules

basic classifier: static rules

basic classifier: static rules

example: power drain > 2W?

basic classifier: static rules

example: power drain > 2W?

pros: scalable, simple

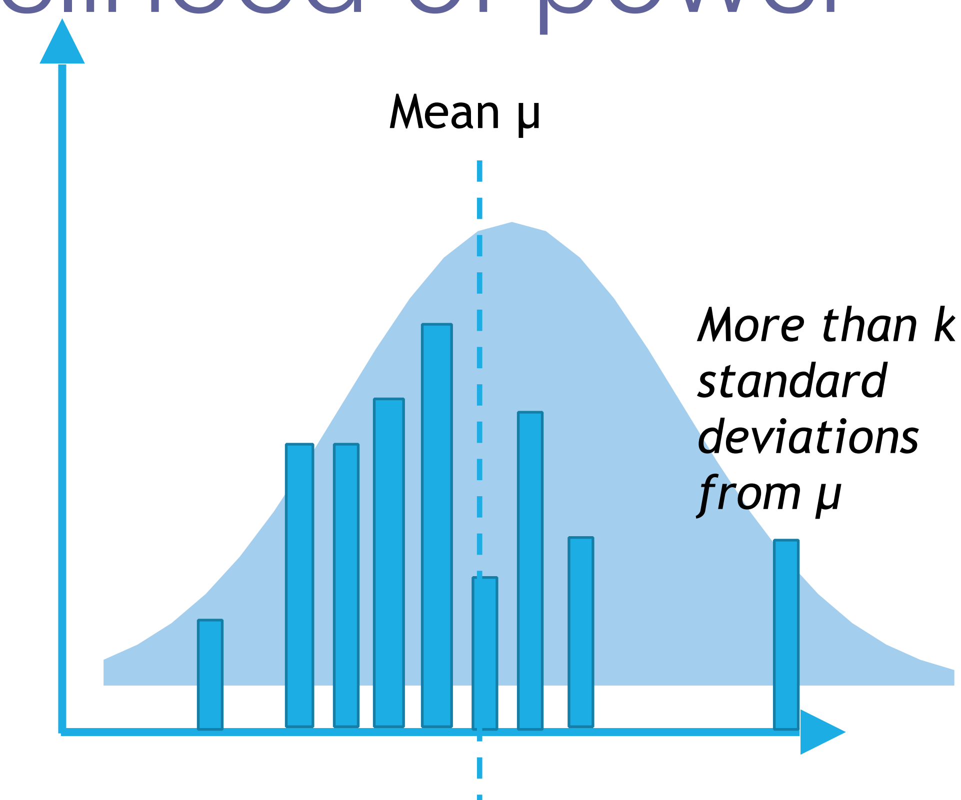
cons: highly brittle, may miss events

better classifier: use ML & statistics

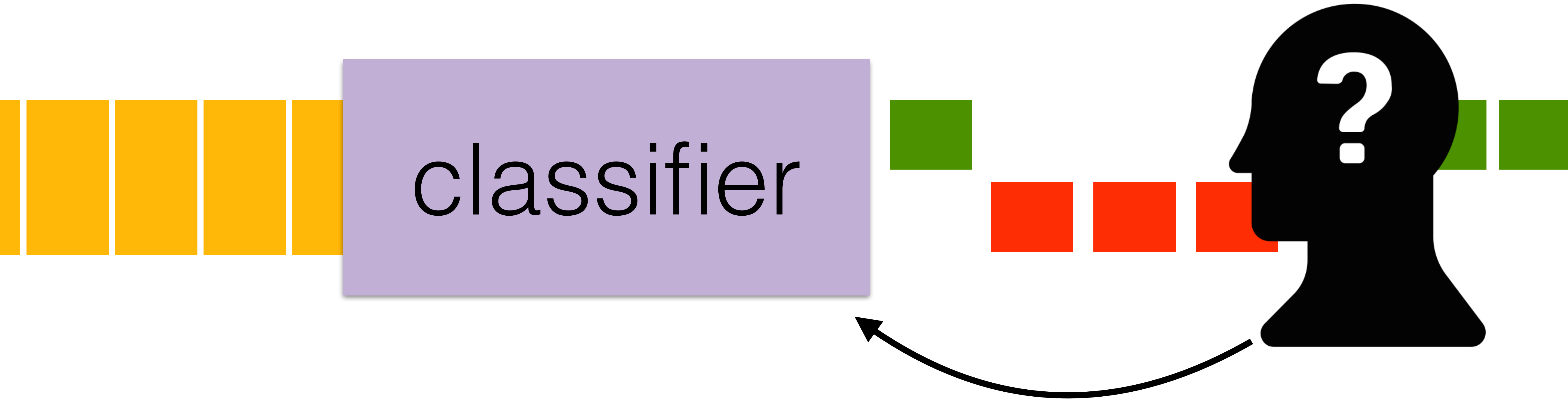
example: compute statistical likelihood of power activity given user population

better classifier: use ML & statistics

example: compute statistical likelihood of power activity given user population



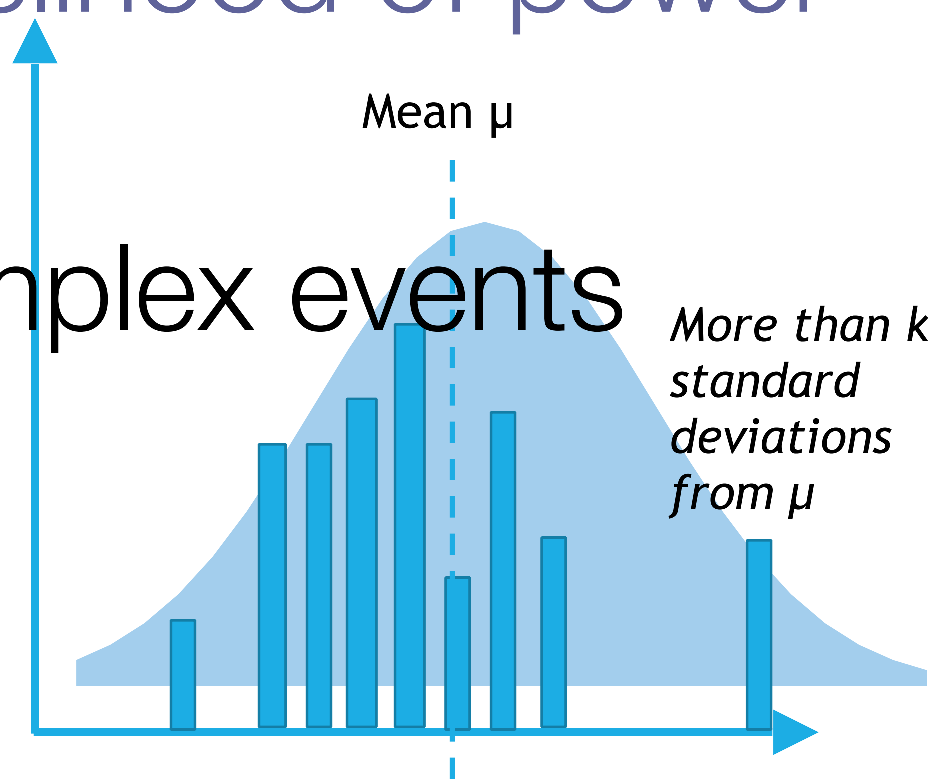
better classifier: use ML & statistics



example: compute statistical likelihood of power activity given user population

pros: can model dynamic & complex events

cons: often slow!



models are expensive to run

e.g., state-of-art CNN: 30fps requires $1200 GPU

models are expensive to run

e.g., state-of-art CNN: 30fps requires $1200 GPU

anecdote: speed vs. quality

engineers at major online service monitoring per-device QoS:

off-the-shelf stats packages too slow, not scalable

solution: manually tune thresholds per-user, per-device!

models are expensive to run

e.g., state-of-art CNN: 30fps requires $1200 GPU

anecdote: speed vs. quality

engineers at major online service monitoring per-device QoS:

off-the-shelf stats packages too slow, not scalable

solution: manually tune thresholds per-user, per-device!

result: brittle, reactive, false negatives

wanted: accurate, scalable classifiers

raw data is still too much

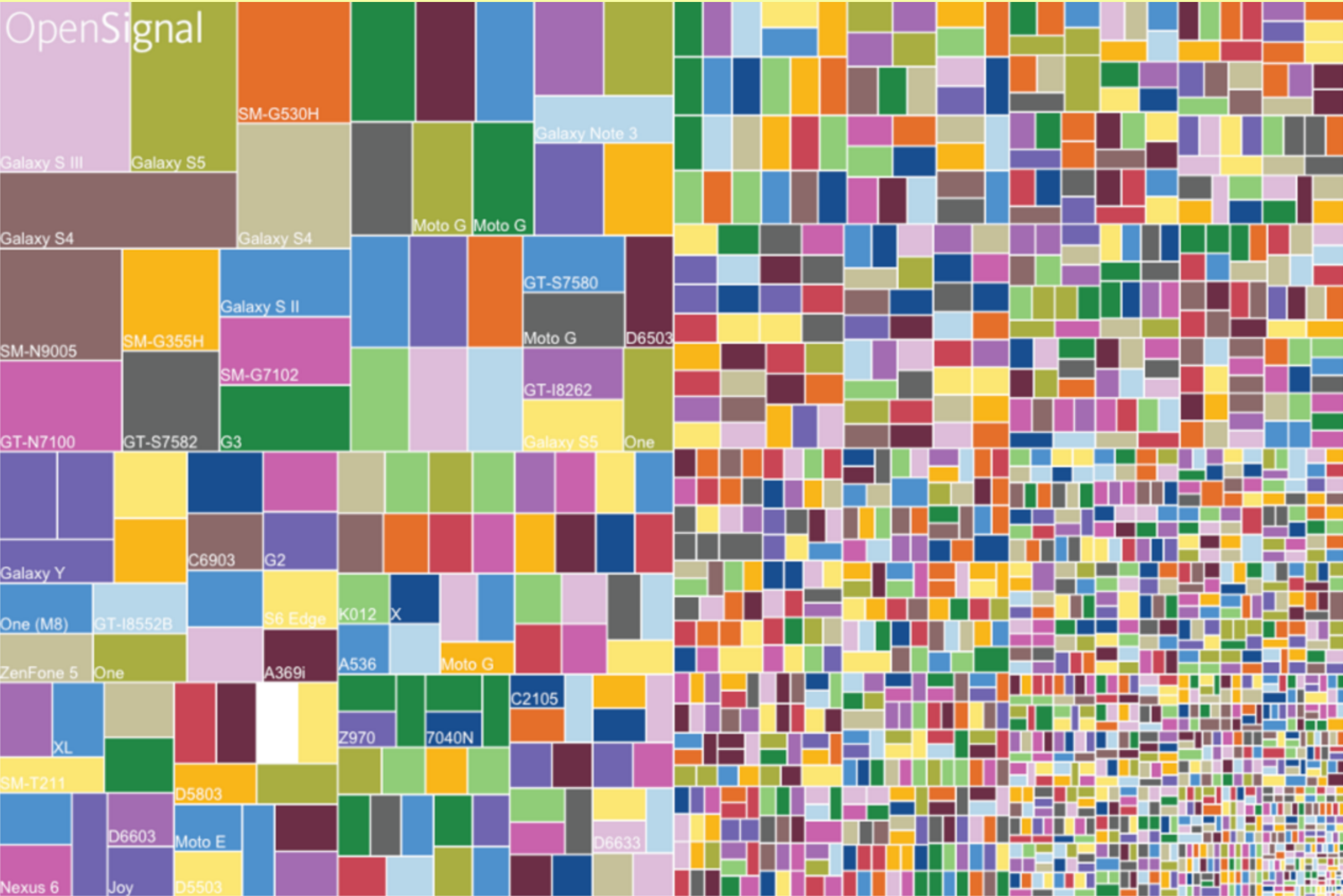
even filtered data is problematic at scale

high volume still overwhelms human attention

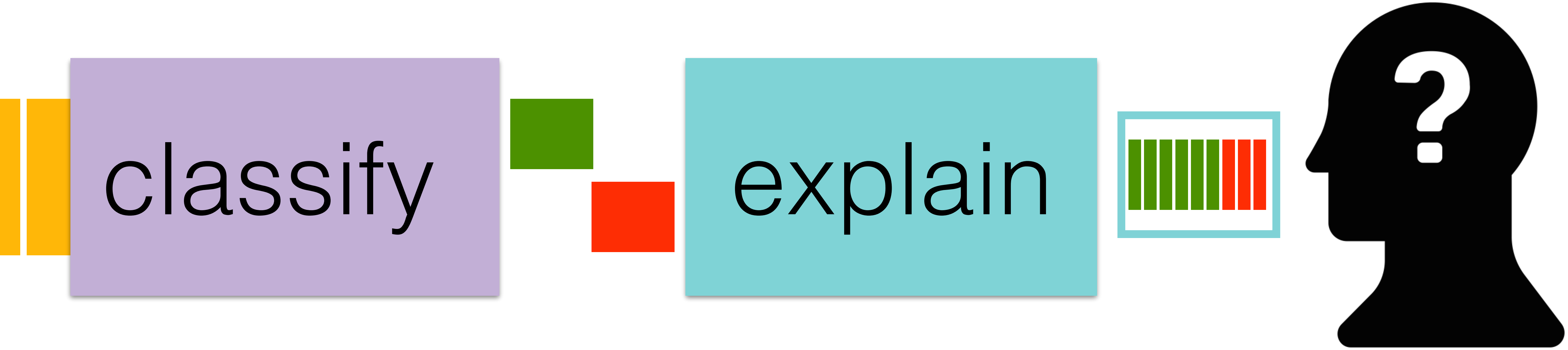
high-dimensional attributes can obscure trends

reading_id	device_id	model	firmware_version	temperature	power_drain
123912	4914	M404	0.4	72.437732	0.350747
123913	4914	M205	0.3.1	72.008312	0.317166
123914	4234	M606	0.3.1	77.035467	0.210455
123915	2144	M104	0.3.1	74.724524	0.215144
123916	7594	M205	0.4	77.459194	0.278835
123917	1958	M404	0.3.1	74.661423	0.336197
123918	3882	M204	0.3.1	70.565394	0.276593
123919	1685	M205	0.3.2	71.564029	0.305529
123920	1810	M205	0.2.4	74.533990	0.396767
123921	2816	M101	0.3.1	79.252845	0.337921
123922	3273	M104	0.3.1	79.040148	0.294952
123923	2526	M205	0.4	72.481287	0.308807
123924	707	M606	0.3.1	75.249177	0.335019
123925	8465	M606	0.2.4	70.153331	0.390715
123926	2816	M104	0.3.1	70.951893	0.222472
123927	663	M606	0.3.2	79.165254	0.316542
123928	3409	M404	0.3.1	75.823737	0.335797

android device types by popularity

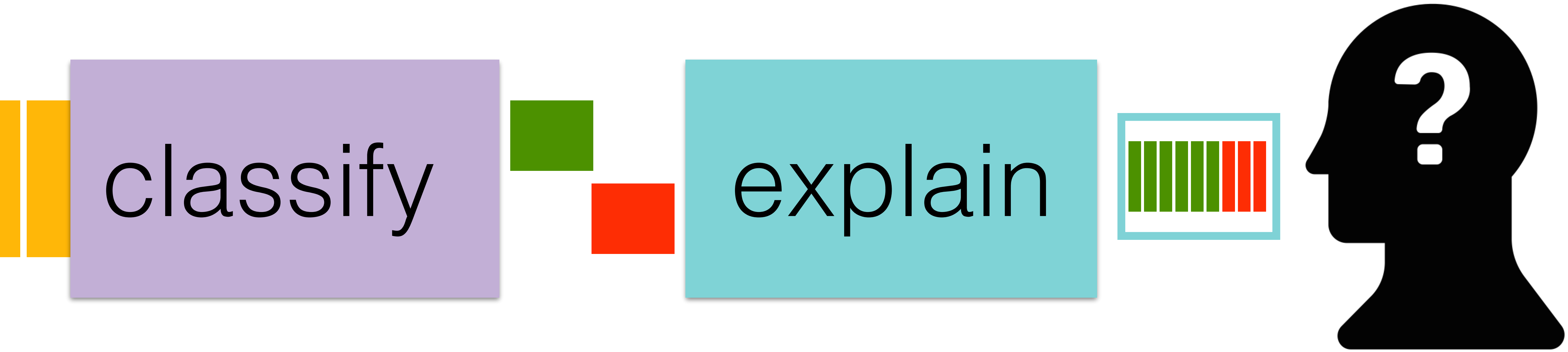


explanations aggregate results



highlight commonalities and trends

explanations aggregate results

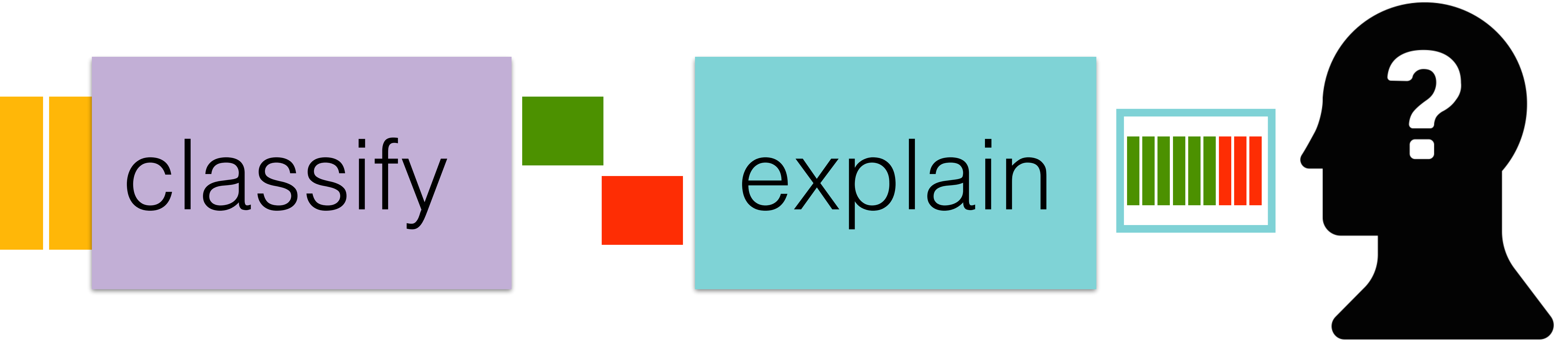


highlight commonalities and trends

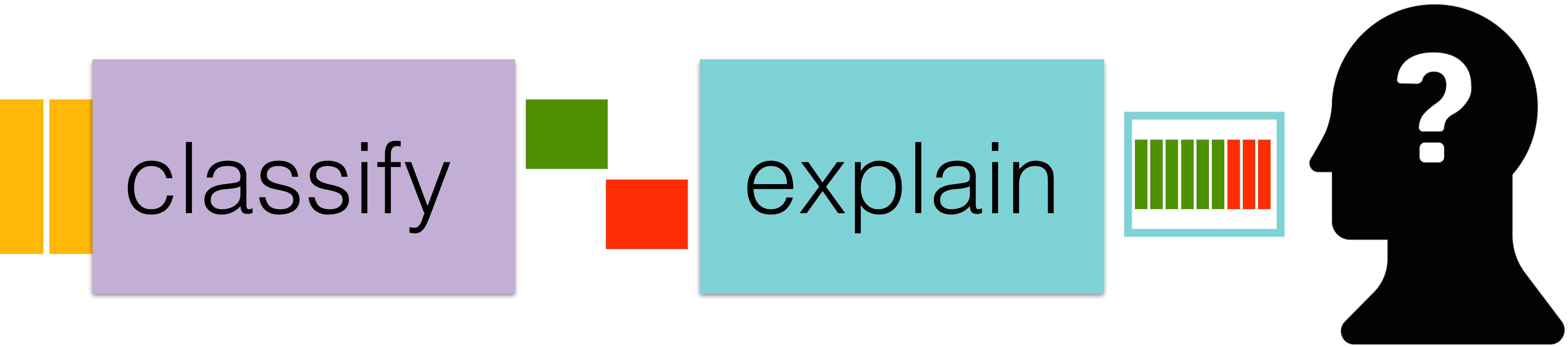
e.g., Android Galaxy S7 devices running app version 2.4.4 are 51x more likely than usual to have extreme power drain

return aggregates and representative events instead of returning raw data

the key to fast data
combine: classify and explain



the key to fast data
combine: classify and explain



how should we do it?

APACHE STORM™ APACHE Spark™ Flink

dataflow (alone) is not enough

dataflow: a substrate, not a complete solution

dataflow (alone) is not enough

dataflow: a substrate, not a complete solution

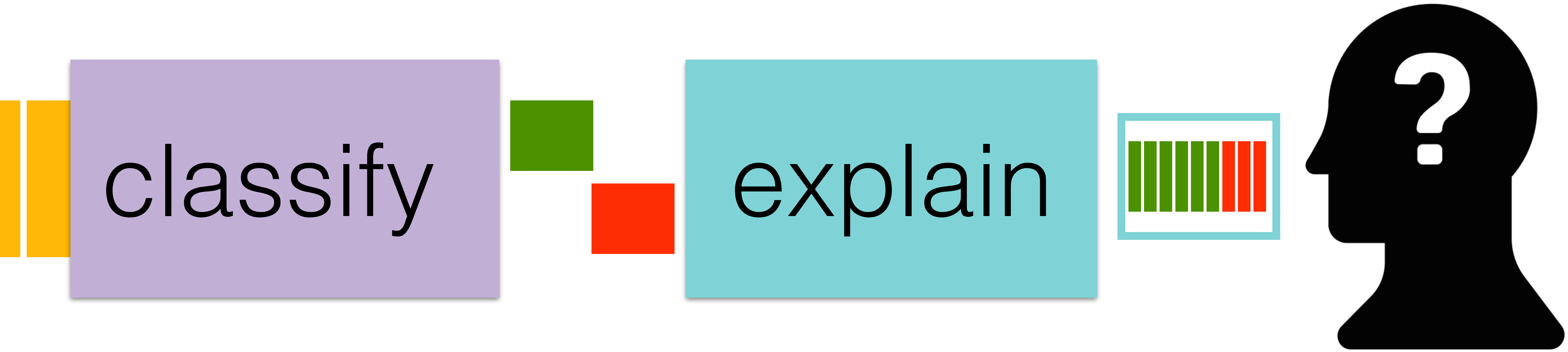
dataflow (alone) is not enough

dataflow: a substrate, not a complete solution

APACHE STORM™ Apache Spark™ Flink

missing:
scalable, modular operators
for prioritizing attention via
classification and explanation

macrobase: a fast data system

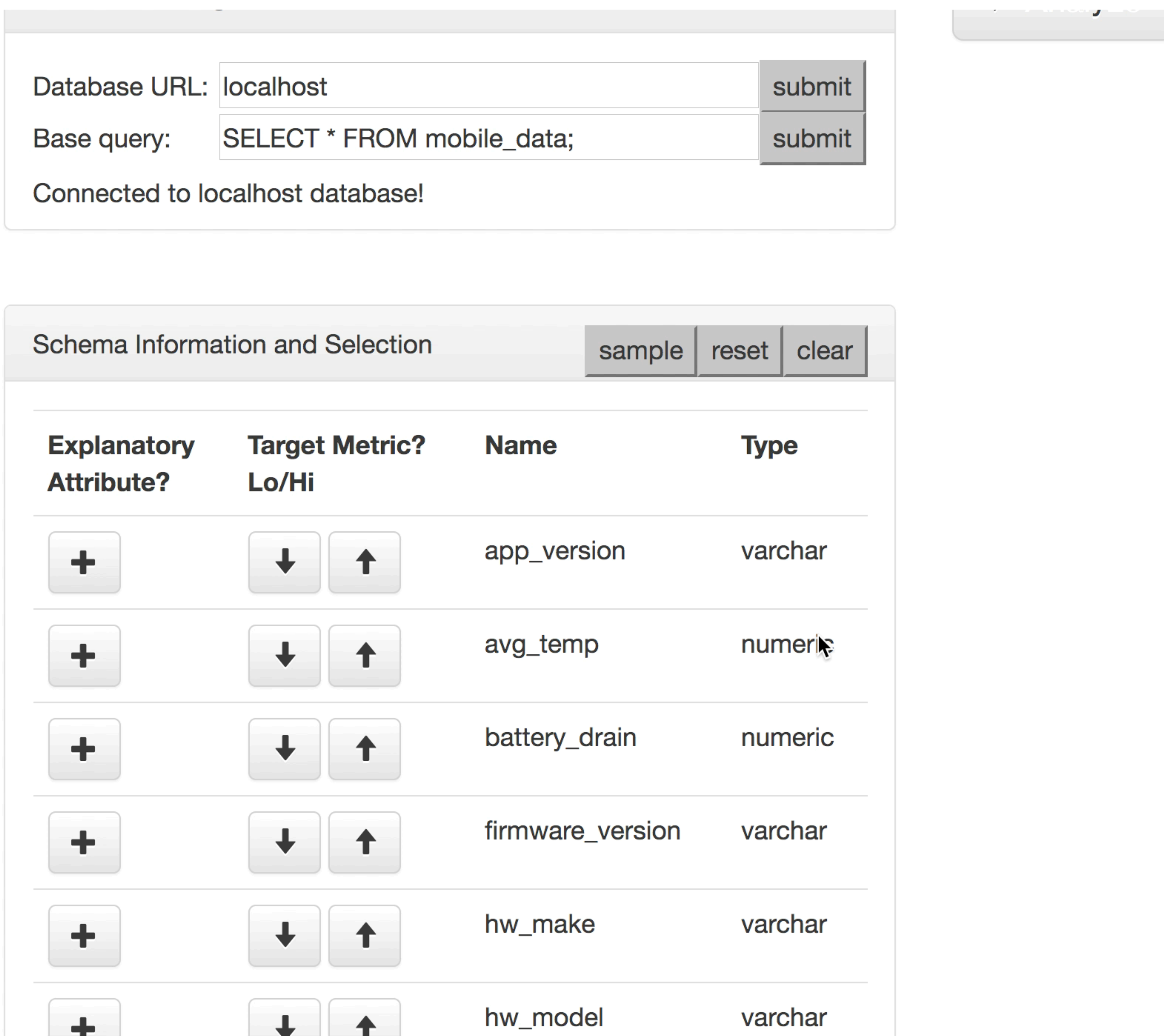


a system providing
fast, reusable, modular operators
for classification and explanation
prioritizing attention in fast data

MacroBase default workflow

input: data attributes, key performance metrics

output: attributes that explain deviations in metrics

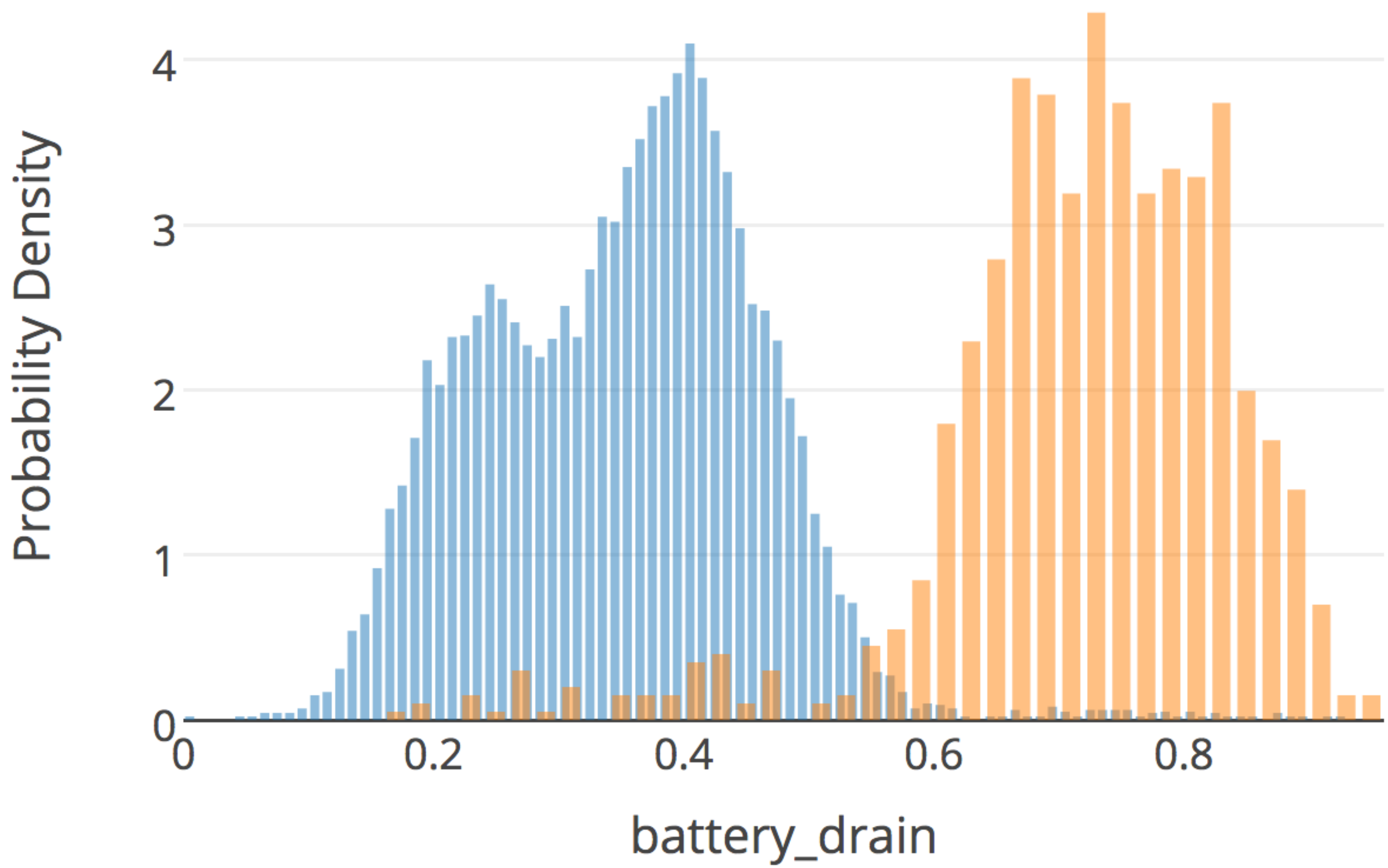


record_id	user_id	state	hw_make	hw_model	firmware_version	app_version	avg_temp	battery_drain	trip_time
131920	49e36c5b031141dd8cf240f7	CO	Lenovo	Lenovo_K910L	4.4.2	v21	79.252124	0.205834	40.910145
131921	a670eab2bc6d4e5991ea4269	WV	TCT (Alcatel)	4009A	7.1.1	v36	72.136380	0.184874	47.253076
131922	247c64e48a8743829c5f7199	UT	TCT (Alcatel)	4009A	7.1.1	v31	77.300103	0.230015	25.342140
131924	6bd9af7242ca480a96d75d0d	OH	HTC	HTC_M10u	6.0.1	v38	70.937014	0.454293	38.661611
131926	d449b12dcb6346d7af1021de	HI	HTC	HTC_Wildfire_S_A510b	6.0	v46	75.436764	0.151338	17.785555
131927	fff8907aa14e4a50ab76bd46	HI	bq	Aquaris_E4.5	4.4.1	v38	70.208187	0.286005	60.443799
131929	8226cd65bb1f4d61a6fcf455	MI	TCT (Alcatel)	ALCATEL_one_touch_97	6.0.1	v35	73.113370	0.249834	16.881133
131930	30e726fadc6744b2ace2d76b	LA	TCT (Alcatel)	ALCATEL_ONE_TOUCH_60	5.0	v40	77.918077	0.405417	51.163642
131931	569f35993da246f4afc83c2e	FL	Lava	S1	6.0.1	v44	76.558080	0.416760	42.252460
131932	9d2db241316c43378b8ec14c	AL	LGE	LG-D724	7.0	v29	76.760340	0.334446	37.922632
131933	484a1ced0a6a46468874861c	LA	Hisense	LED42K680X3DU	4.4.4	v49	77.138769	0.409485	23.345804
131934	d375d5a0e10d46cf9b91e343	MI	Techno	TECNO_P5S	6.0	v31	70.115019	0.179464	45.051123
131936	e4835a64d96e4e89997ce027	WI	ZTE	Z828	6.0.1	v35	71.615570	0.396389	47.662474
131937	cf00ae2105bb4e3cb43b64b2	FL	Spice	Spice_Mi-498H	5.0	v42	72.045184	0.327405	45.099422
131939	c94d264a846149f08f51c28e	RI	Infocus	InFocus_M320u	4.4.1	v49	73.543359	0.224504	19.069803
131940	c3c82947ab5a4d09b52afe21	MI	Hisense	Bouygues_Telecom_Bs_	4.4.2	v41	75.949490	0.409603	38.287879
131943	4e4566143b144be1809ad4d9	RI	Turk Telekom	E4	7.0	v22	76.831963	0.486268	28.306756
131944	0ee8ff83606b496392bedd49	NE	Sony Ericsson	MT11i	4.4.4	v31	78.661817	0.346537	26.749918
131946	0ee8ff83606b496392bedd49	OK	LGE	LS670	4.0.4	v30	78.186519	0.381604	27.601968
131947	7a4bc4c56aa54d20b1119d42	NV	Oppo	F1f	4.3.1	v38	77.434095	0.436364	40.869723
131948	c3c82947ab5a4d09b52afe21	GA	Huawei	HUAWEI_Y320-U151	4.4.3	v42	77.715329	0.281726	23.077248
131949	2b0acf33a91f49b6aa70cbe5	MO	ZTE	KPN_Smart_300	4.4.4	v39	75.368614	0.371224	44.295975
131950	5aab0148ea794d2eacfc9a27	NV	Ketablet	TR10CS1	5.0	v49	79.459844	0.491424	37.653744

Column	Value
app_version	v50
hw_model	em_i8180
hw_make	Emdoor

correlated attributes

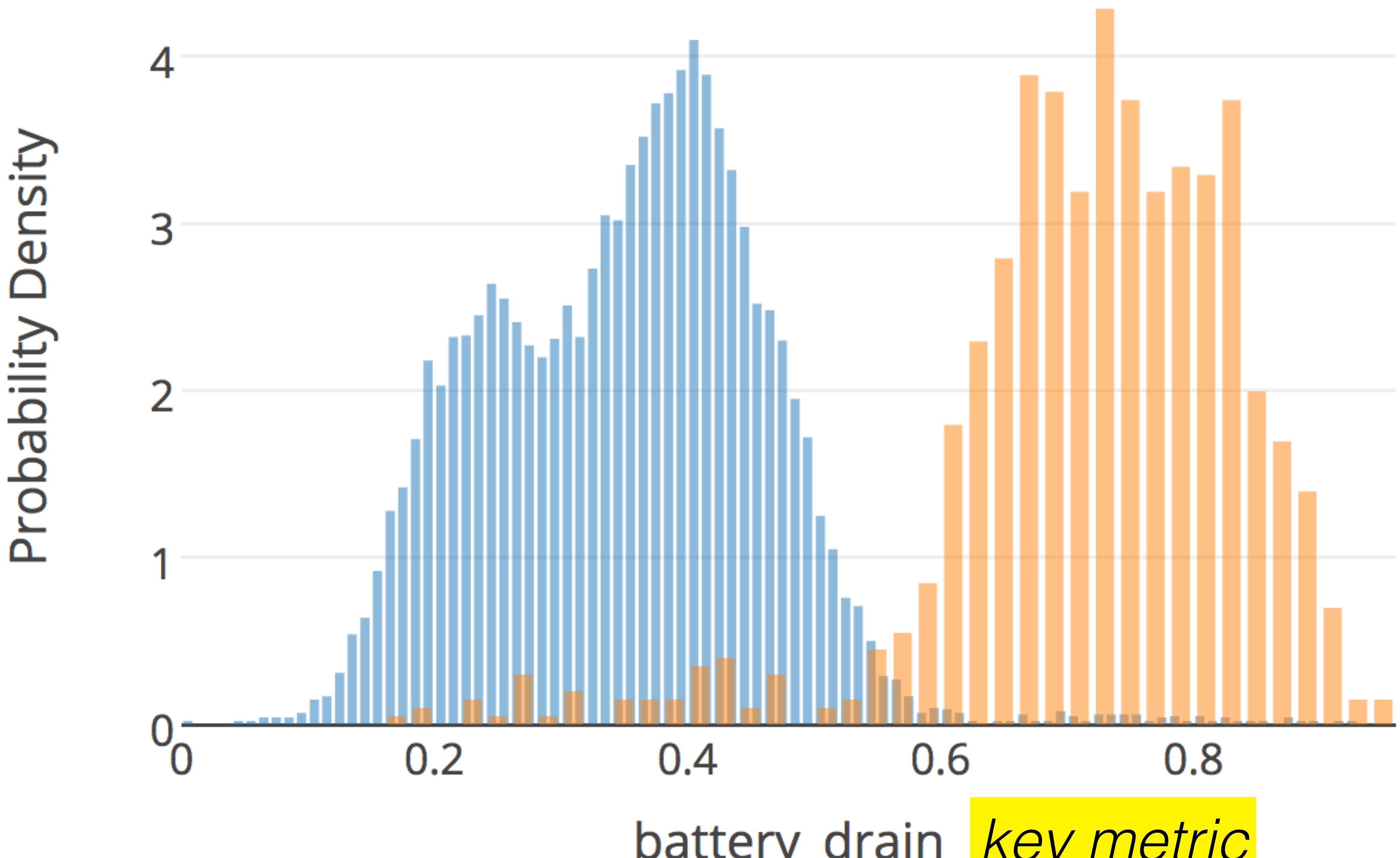
Support: 0.8226
Ratio Out/In: 472.92
Records: 849



Column	Value
app_version	v50
hw_model	em_i8180
hw_make	Emdoor

correlated attributes

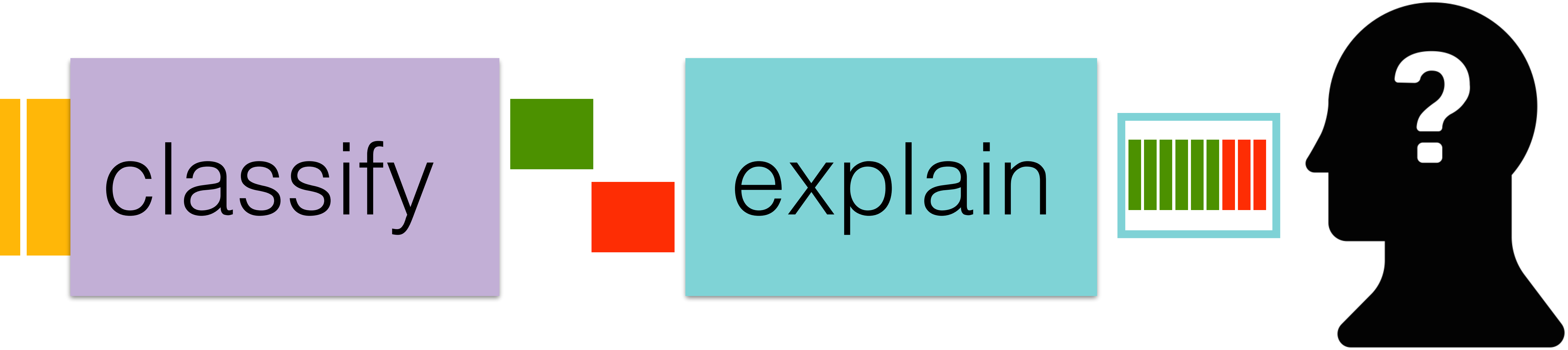
Support: 0.8226
Ratio Out/In: 472.92
Records: 849



key metric

CAMBRIDGE
MOBILE TELEMATICS

"MacroBase discovered a rare issue with the CMT application and a device-specific battery problem. Consultation and investigation with the CMT team confirmed these issues as previously unknown…"



key: make this combo fast

example: end-to-end optimization

example: end-to-end optimization

A	B	streaming explanation
B	C	
A	D	
A	A	
A	B	
	D	
	C	
	B	
	B	
	E	
	B	
	B	

example: end-to-end optimization

A	B
B	C
A	D
A	A
A	B
	D
	C
	B
	B
	E
	B
	B

streaming explanation

standard solution:
find correlations w/in each class

example: end-to-end optimization

A	B
B	C
A	D
A	A
A	B
	D
	C
	B
	B
	E
	B
	B

streaming explanation

standard solution:
find correlations w/in each class

A: 80%
B: 20%

example: end-to-end optimization

A	B
B	C
A	D
A	A
A	B
	D
	C
	B
	B
	E
	B
	B

streaming explanation

standard solution:
find correlations w/in each class

A: 80%
B: 20%

A: 0.1%
B: 46%
C: 31.9%
D: 22%

example: end-to-end optimization

A B A A A

B C D A B D C B B E B B

streaming explanation

standard solution:
find correlations w/in each class

A: 80% B: 20%

A: 0.1% B: 46% C: 31.9% D: 22%

better idea:
exploit *cardinality imbalance*
correlate "outliers", probe "inliers"

example: end-to-end optimization

A	B
B	C
A	D
A	A
A	B
	D
	C
	B
	B
	E
	B
	B

streaming explanation

standard solution:
find correlations w/in each class

A: 80% A: 0.1% C: 31.9%
B: 20% B: 46% D: 22%

better idea:
exploit *cardinality imbalance*
correlate "outliers", probe "inliers"

A: 80%

example: end-to-end optimization

A	B
B	C
A	D
A	A
A	B
	D
	C
	B
	B
	E
	B
	B

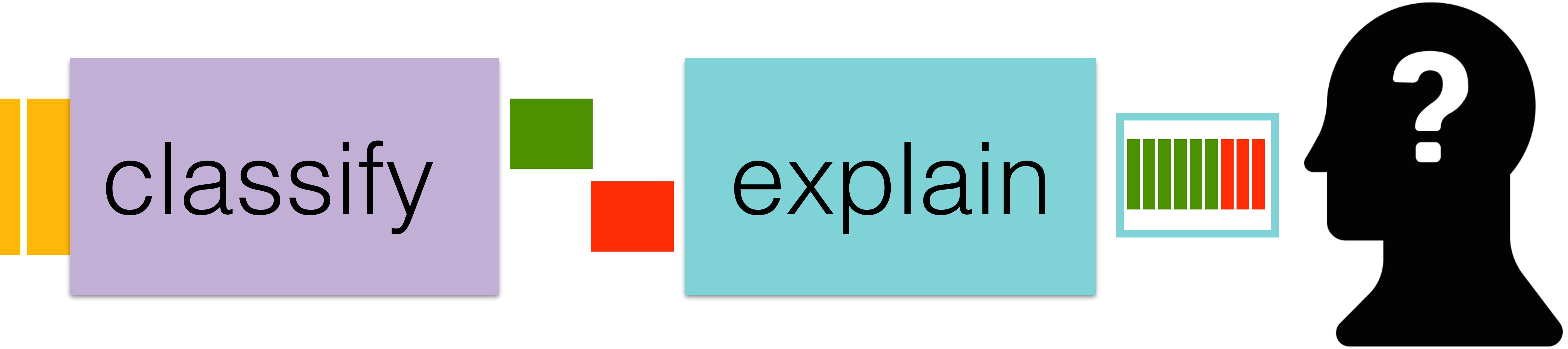
streaming explanation

standard solution:
find correlations w/in each class

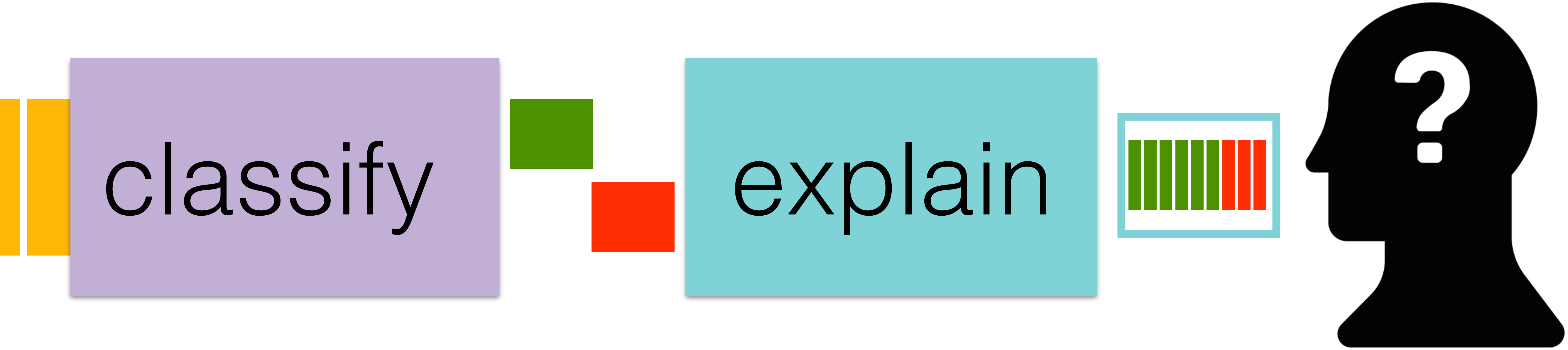
A: 80% A: 0.1% C: 31.9%
B: 20% B: 46% D: 22%

better idea:
exploit *cardinality imbalance*
correlate "outliers", probe "inliers"

A: 80% A: 0.1%



key: make this combo fast



key: make this combo fast

surprise:
this combo enables new optimizations

one weird trick for 2017 systems research

1.) read a textbook on statistics/ML
2.) implement the thing that should work
3.) observe it's really slow
4.) make it fast using systems techniques

needed: classic systems techniques

indexing, caching, predicate pushdown, sketching

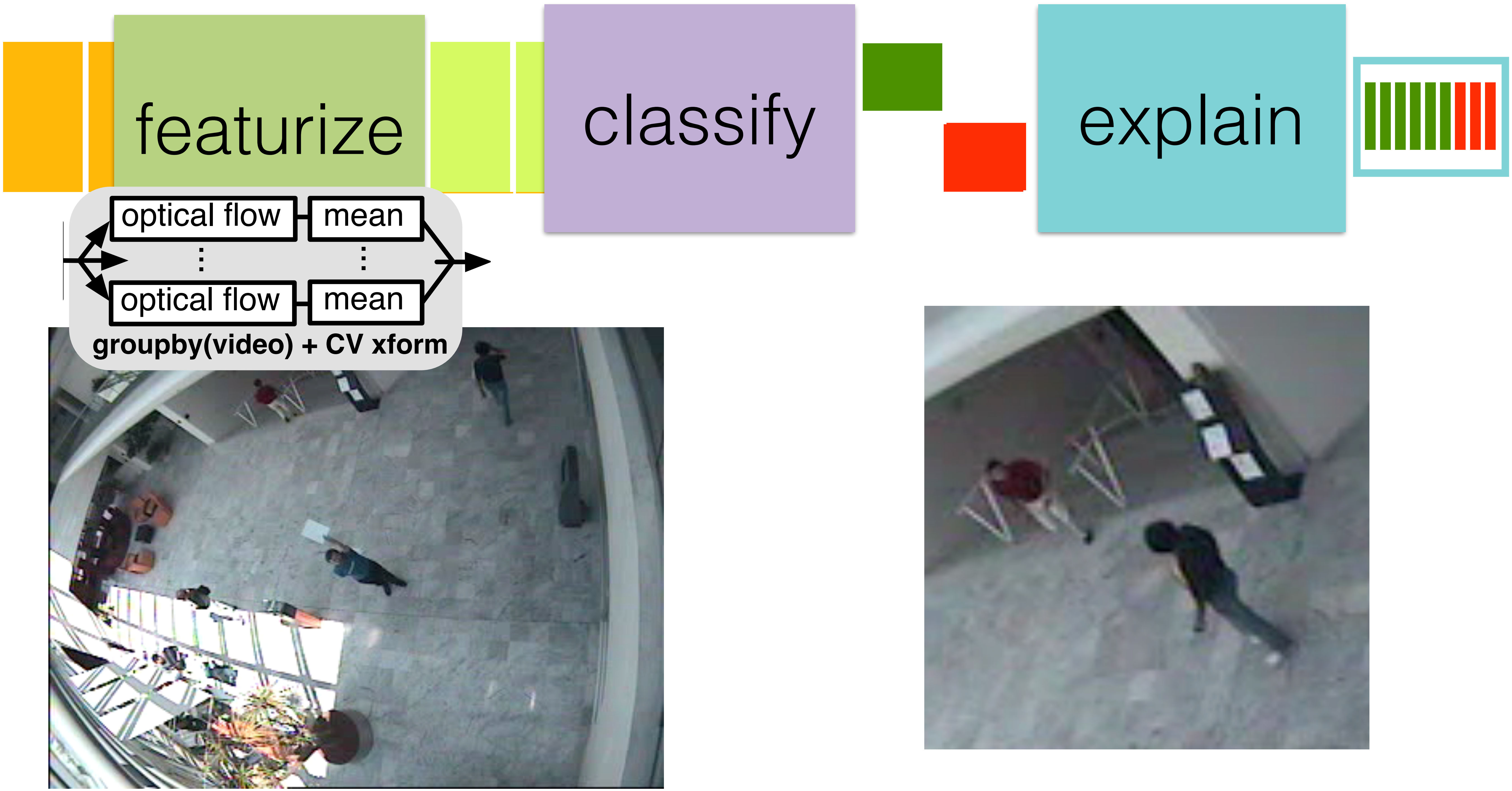
is this system just a bunch of one-off hacks?

is this system just a bunch of one-off hacks?

no! only need a small # of core operators, coupled with domain-specific features

is this system just a bunch of one-off hacks?

no! only need a small # of core operators, coupled with domain-specific features

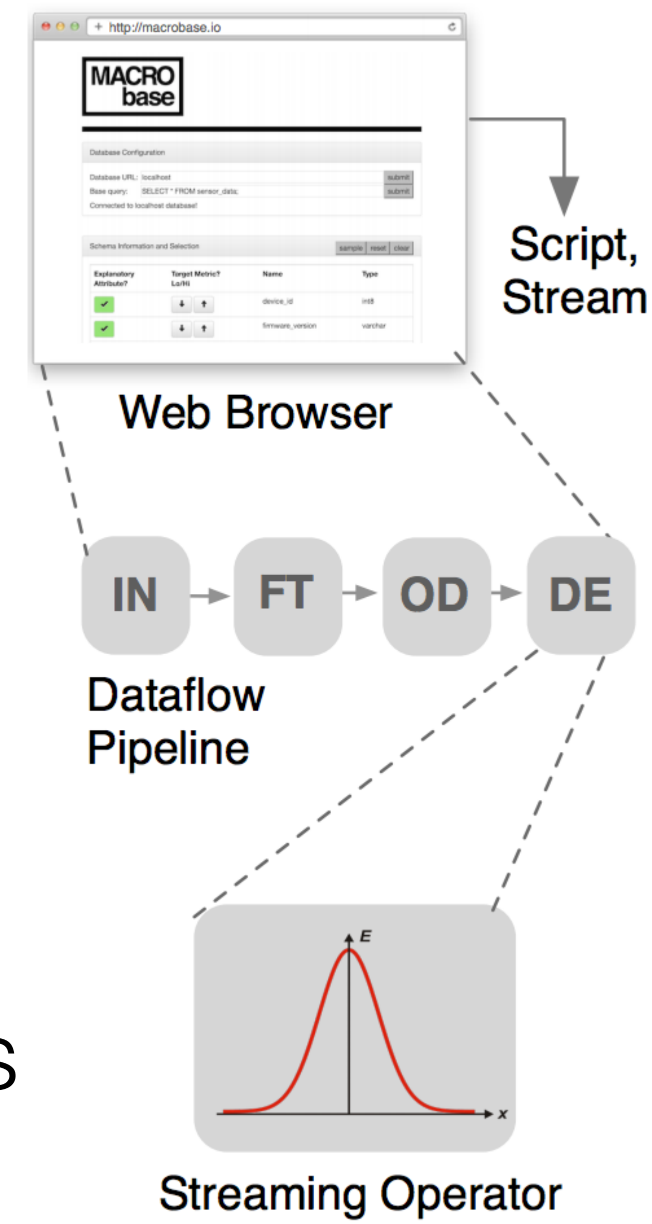


a range of interfaces empowers a range of users:

domain experts: point and click UI

scripters: custom dataflow pipelines

ML and systems ninjas: custom operators



users inform design

automotive
monitoring fleet QoS

online services & datacenters (DevOps / monitoring)
identifying slow containers, exception telemetry

industrial manufacturing
key sources of process variance in product

geophysics
Lunar water ice detection, seismic activity detection

fast data

- overabundant data, scarce human attention
- a major opportunity for systems, w/ real use cases

macrobase

- an open source search engine for fast data
- modular, efficient classification and explanation