Wayfair's Mercury Platform:
Scaling ML Applications via Programmatic Feature Definition, Build, and Maintenance

Alexander Hristov, Staff ML Scientist, Wayfair

FEATURE STORE SUMMIT 2023

Organized by HOPSWORKS

wayfair

Key Idea for Mercury:

- Prioritize **maintainability** and **efficiency** over flexibility in defining new features.
- Improve model performance by increasing access to signals
- **Define**, **build**, **consume**, and **maintain** features as a **group,** rather than individually.

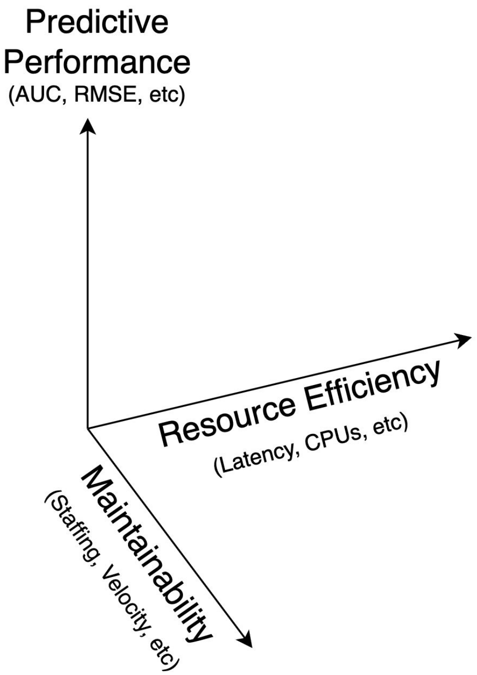
Key Idea for Mercury:

- Prioritize **maintainability** and **efficiency** over flexibility in defining new features.
- Improve model performance by increasing access to signals
- **Define**, **build**, **consume**, and **maintain** features as a **group,** rather than individually.

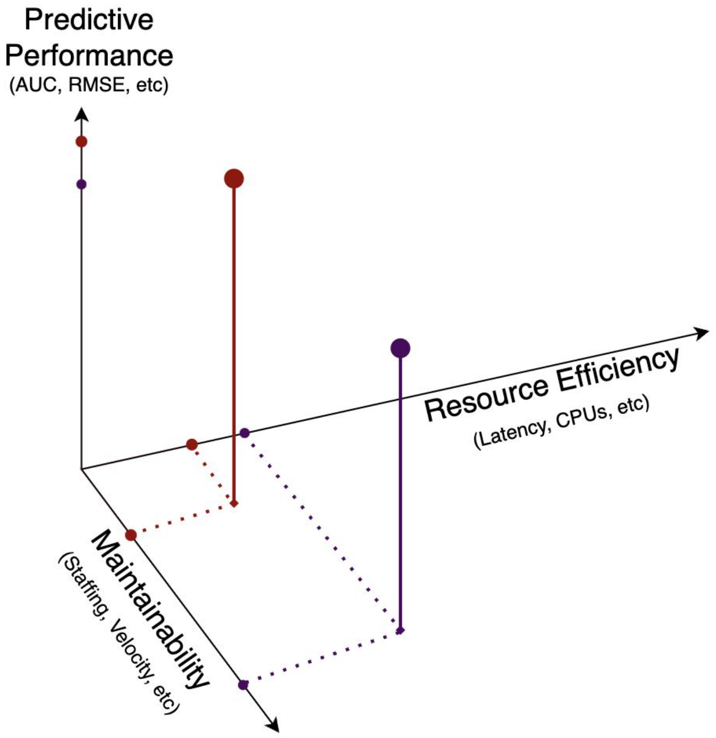
Part 1: Motivation

Why Trade Flexibility for Efficiency in Feature Platform Design

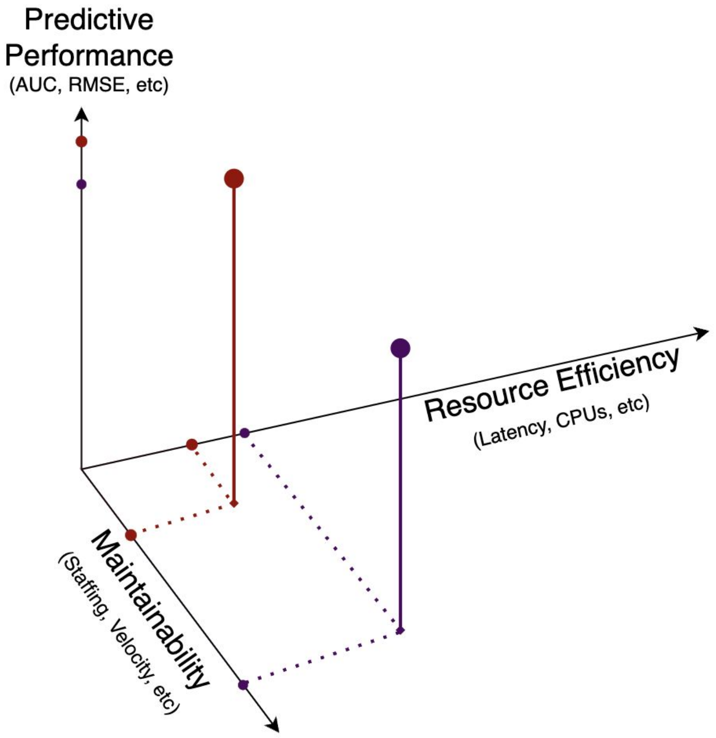
All ML systems make tradeoffs.



Which ML system would you prefer to deploy?



Which ML system would you prefer to deploy?



It depends on ...

- Team: size and maturity
- Resources: budget and elasticity
- Problem value, i.e. d$/dRMSE
- What else could you be doing?
  - Wide teams: 0→ 1, then onto another task
  - Tall teams: Iterate on the same problem

In other words,
The more predictive model is not always better.
Success comes from managing tradeoffs.

Feature platform flexibility is an easily overlooked tradeoff.

Nonetheless, determining this tradeoff <u>at platform level</u> can help teams scale.

Very Flexible:

Make any feature in any way

- More focus on "best case" performance
- Hardest to provide cost, latency guarantees at platform level
- Some changes hard to automate

Very Opinionated:

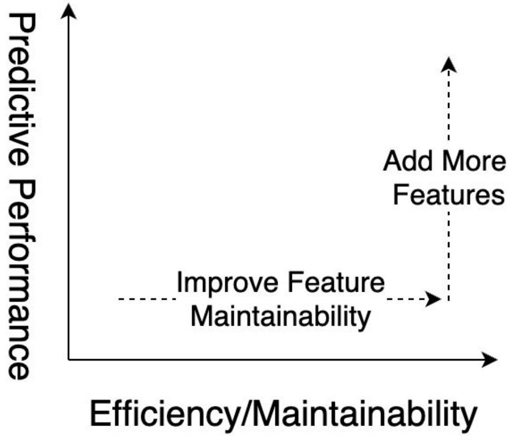
Features must follow specific approach

- More focus on "out of the box" performance
- Stronger platform guarantees for speed, cost
- Easier to automate

Be more opinionated, because ML team time is bounded.

1. Improve predictive performance obtained in bounded time, not the hypothetical best-case performance.

2. Improve not just velocity to deploy new things, but also **carrying capacity**, the ability to support more features and models well.



Mercury is strongly opinionated that users should define, build, consume and maintain features as groups, not one at a time.

Many tabular features can share a pattern, creating a game of “fill in the blanks”

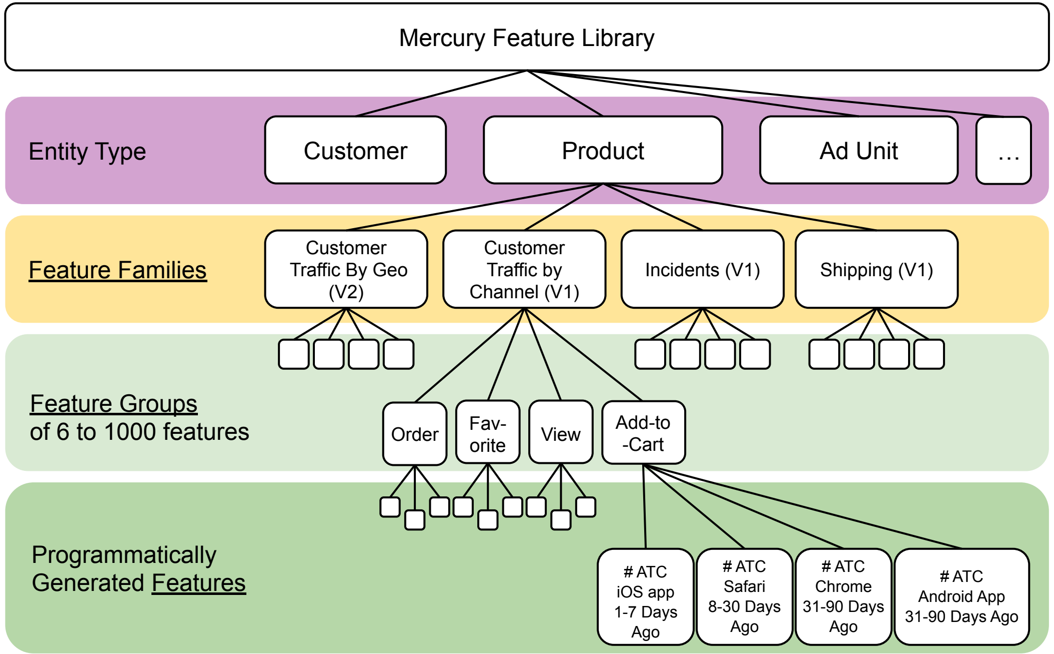
______________ for a product from sessions on	________ between	____________ days ago
# of add-to-cart	iOS app	1 and 7
avg. # per order	desktop chrome	8 and 30
highest review	mobile safari	31 and 90
…	…	91 and 365
		…

Days since a customer last ___________	a ________
browsed for	Sofa
added to cart	Desk
…	…

Features with the same **template** and source **event** can build together

wayfair

Mercury groups features hierarchically for performance and maintainability



For discovery and versioning, group features sharing entity of analysis (Geo, Channel)

For performance, group features together that share actions (atc, clicks, etc)

Programmatic features improve scale and performance in the long run.

1000s	Trillions	20+	Billions	100s
Features Defined	Feature Values Computed Per Day	ML Applications using Mercury	Daily Model Predictions	Model objects deployed use Mercury

User Experience Improvements:

Predictive Performance: 2-9% for 3 preexisting applications that migrated to Mercury
Deploy Speed: 2-5x across multiple teams that used Mercury for new projects
Maintainability: One team reported saving >4 hrs per week per model pipeline

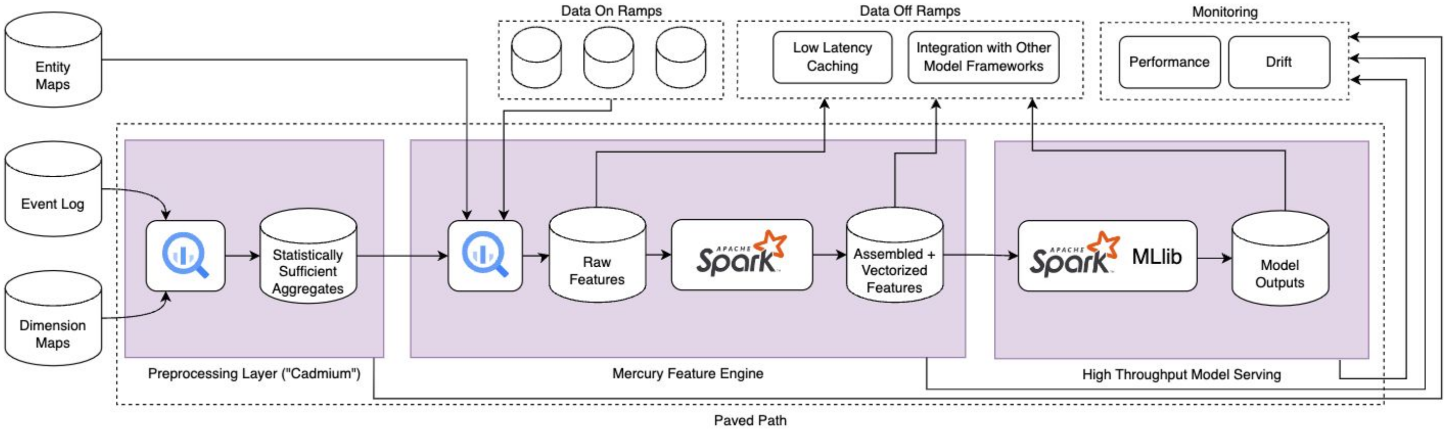
Disclaimer:
Scale isn't free, comes with restrictions on latency and flexibility that may not make sense in all cases.

Part 2: Key Architecture Enablers

Defining and building thousands of features

High Level Logical Architecture

The paved path is highly opinionated, still able to support many applications



Key Lessons:

1. Pre-aggregation is an important step for improving maintainability and performance
2. Build features together and package as vectors
3. Emphasize reproducibility of feature computation
4. Drive reuse in multiple ways

Mercury preprocesses event logs to improve maintainability and performance

Event Log

Customer	Product ID	Timestamp	Geo	Price
Z2Y2X2	ABC123	4/1/2023 6:02:01	CANADA	141.75
QWE747	ABC123	4/1/2023 2:31:48	CANADA	141.75
B3RT13	ABC123	4/1/2023 6:02:01	CANADA	141.75
T1MT1T	ABC123	4/3/2023 11:25:17	US	103.47

↓

Identifier: Product
TimeBin: Day
Dimension: Geo
Aggregates: Sum[Price], Count

↓

Product ID	Time Bin	Geo	Sum Prices	Count
ABC123	4/1/2023	CANADA	425.25	3
ABC123	4/3/2023	US	103.47	1

Try to offload as much compute to this stage: schema standardization, analytic transforms, etc

wayfair

* Some dimension maps are not 1:1, so can only support 1 dimension per table
**All identifiers and numbers for illustrative purposes only and do not reflect real data.

FEATURE STORE SUMMIT 2023

Mercury preprocesses event logs to improve maintainability and performance

Event Log

Customer	Product ID	Timestamp	Geo	Price
Z2Y2X2	ABC123	4/1/2023 6:02:01	CANADA	141.75
QWE747	ABC123	4/1/2023 2:31:48	CANADA	141.75
B3RT13	ABC123	4/1/2023 6:02:01	CANADA	141.75
T1MT1T	ABC123	4/3/2023 11:25:17	US	103.47

Identifier: Product
TimeBin: Day
Dimension: Geo
Aggregates: Sum[Price], Count

Product ID	Time Bin	Geo	Sum Prices	Count
ABC123	4/1/2023	CANADA	425.25	3
ABC123	4/3/2023	US	103.47	1

Dimension Table

Product ID	Style	Started	Ended
ABC123	Contemporary	2/12/2023	5/12/2023
XZY222	Coastal	1/6/2021	-

Identifier: Customer
TimeBin: Month
Dimension: Style (Via Product ID)
Aggregates: Count

Customer	Time Bin	Style	Count
Z2Y2X2	4/2023	Contemporary	2
Z2Y2X2	4/2023	Coastal	6
T1MT1T	4/2023	Scandinavian	2
T1MT1T	4/2023	Contemporary	1
T1MT1T	4/2023	Modern	6

Try to offload as much compute to this stage: schema standardization, analytic transforms, etc

wayfair

* Some dimension maps are not 1:1, so can only support 1 dimension per table
**All identifiers and numbers for illustrative purposes only and do not reflect real data.

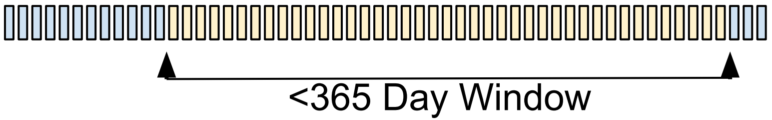
Simple aggregation patterns are flexible enough to support many uses

Supports statistically sufficient aggregates for

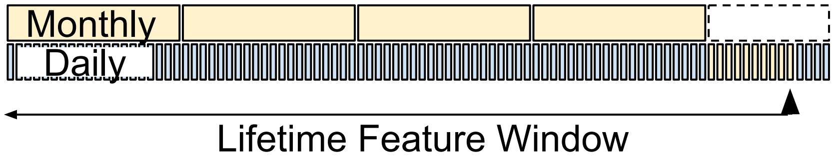
- Counts & Totals:
  store a daily count or sum
- Averages & Quantiles:
  Store daily sum or %-tile and count
- Ratios:
  Store daily sums for the two values
- Min & Max:
  Store daily min and max

… and different windows

Daily - Used for features with up to a 1 year lookback window



Monthly - Used for features with a lifetime-lookback window



Being opinionated does however obstruct some features, like

- distinct counts
- sequence graph transforms

Features build all at once and are packaged into a sparse vector

Example: Avg. price & total count of orders per supplier in __(geo)__ over previous _#_ to _#_ days

- Reduces repeated scans
- All features within the group are “deployed”
- Vectors from multiple groups can be concatenated

Requested Entities

Supplier ID	Date
AA	2023-05-01 00:00:00
YY	2023-05-01 00:00:00

Entity Map Snapshot

Supplier ID	Date	Product ID
AA	2023-05-01 00:00:00	ABC123
YY	2023-05-01 00:00:00	DEF456
YY	2023-05-01 00:00:00	TUV789

Preaggregated Data

Product ID	Time Bin	Geo	Sum Prices	Count
ABC123	2023-04-15	CA	500	3
ABC123	2023-04-12	CA	100	1
DEF456	2023-03-01	DE	160	4
DEF456	2023-03-01	US	181	6
TUV789	2023-04-01	US	144	7

Supplier ID	Date	Time Diff	Geo	Count	Average Price
AA	2023-05-01	(7, 30]	CA	4	150
YY	2023-05-01	(30, 90]	US	4	40
YY	2023-05-01	(30, 90]	DE	13	25

Relative Position

Values

Supplier ID	Date	Features
AA	2023-05-01	[0,....,4,150,....]
YY	2023-05-01	[0,....,13,25,...,4,40,....,]

wayfair

**All identifiers and numbers for illustrative purposes only and do not reflect real data.

Despite building features as large vectors, we are still have flexibility in how models consume the features through SparkML

Product ID	Date	Feature Group 1
ABC123	2023-05-01	[1,0,4]
DEF456	2023-05-01	[0,2,0]

Product ID	Date	Feature Group 2
ABC123	2023-05-01	[3,3,3,4]
DEF456	2023-05-01	[1,0,0,1]

Feature Group Concatenation

Product ID	Date	Combined Features
ABC123	2023-05-01	[1,0,4,3,3,3,4]
DEF456	2023-05-01	[0,2,0,1,0,0,1]

Model Specific Feature Selection via spark `VectorSlicer`

Product ID	Date	Selected Featues
ABC123	2023-05-01	[4,3,3,3]
DEF456	2023-05-01	[0,1,0,0]

Product ID	Date	Selected Featues
ABC123	2023-05-01	[1,0,4]
DEF456	2023-05-01	[0,2,1]

Inference

Product ID	Date	Model A
ABC123	2023-05-01	0.16
DEF456	2023-05-01	0.03

Model A Pipeline

Product ID	Date	Model B
ABC123	2023-05-01	0.82
DEF456	2023-05-01	0.42

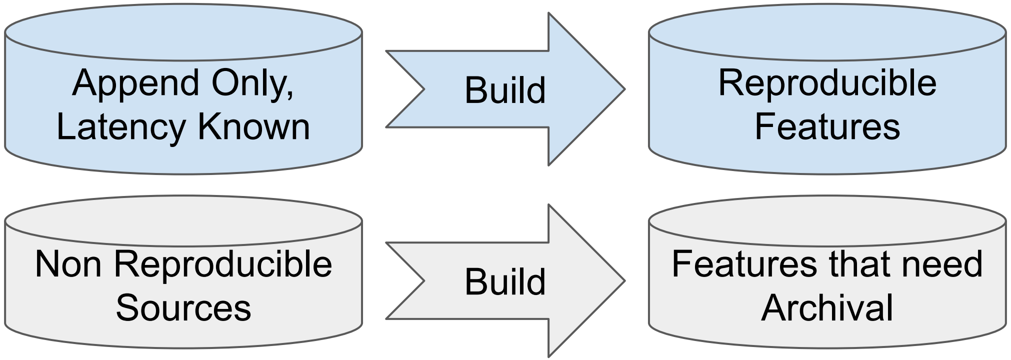
Model B Pipeline

wayfair

**All identifiers and numbers for illustrative purposes only and do not reflect real data.

Write-only data sources help with reproducible compute, tunable retention

Path towards online-offline consistency often depends on the contract with upstream data.



This then determines the requirements and capabilities of a feature platform, such as

- **Library**: Recipes to make features
- **Engine**: Turns recipes from the library into data
- **Archival**: Store feature values for training and offline eval
- **Cache**: Low latency retrieval.

Features can also be reused across entities

The feature compute steps are independent of entity map used

Preaggregated Data

Product ID	Time Bin	Geo	Count
ABC123	2023-04-15	CA	3
ABC123	2023-04-12	CA	1
DEF456	2023-03-01	DE	4
DEF456	2023-03-01	US	6
TUV789	2023-04-01	US	7

Product Class	Date	Product ID
Sofa	2023-05-01	ABC123
Area Rug	2023-05-01	DEF456
Sofa	2023-05-01	TUV789

Product Class	Date	Features
Sofa	2023-05-01	[#####]
Area Rug	2023-05-01	[#####]

Supplier ID	Date	Product ID
AA	2023-05-01	ABC123
HH	2023-05-01	DEF456
HH	2023-05-01	TUV789

Supplier ID	Date	Features
AA	2023-05-01	[#####]
HH	2023-05-01	[#####]
HH	2023-05-01	[#####]

Product ID	Date	Product ID
ABC123	2023-05-01	ABC123
DEF456	2023-05-01	DEF456
TUV789	2023-05-01	TUV789

Product ID	Date	Features
ABC123	2023-05-01	[#####]
DEF456	2023-05-01	[#####]
TUV789	2023-05-01	[#####]

There are many entity hierarchies to which this can apply:

- Household
- Business Account
- UserAccount
- Device

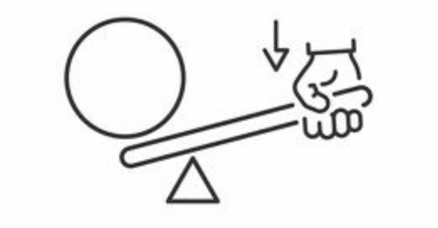
- Category
- Product
- Option Combination

- Channel
- Campaign
- Ad Unit

wayfair

**All identifiers and numbers for illustrative purposes only and do not reflect real data.

Succinctness, automated discovery, and lifecycle management are also important drivers of reuse



300 lines of config can produce thousands of features, compare that to 50 lines of sql for 1 feature.

It's actually possible to review the code for features you use.

Not that you should actually have to.

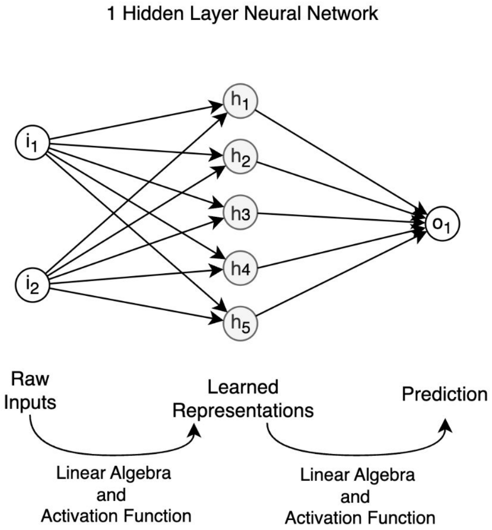
We use scalable feature selection methods to help find predictive signals from the library of thousands.

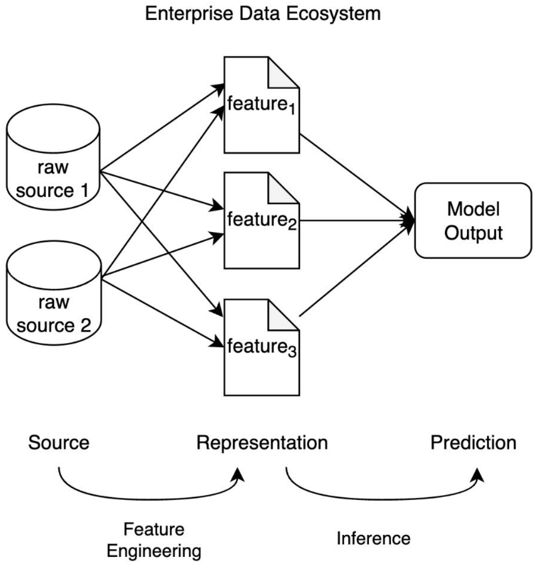
But as helpful as all that is, it's expectations on maintenance of the features that most influence users commitments to reuse features.

Part 3: Intuition

Not-So-Random Kitchen Sinks

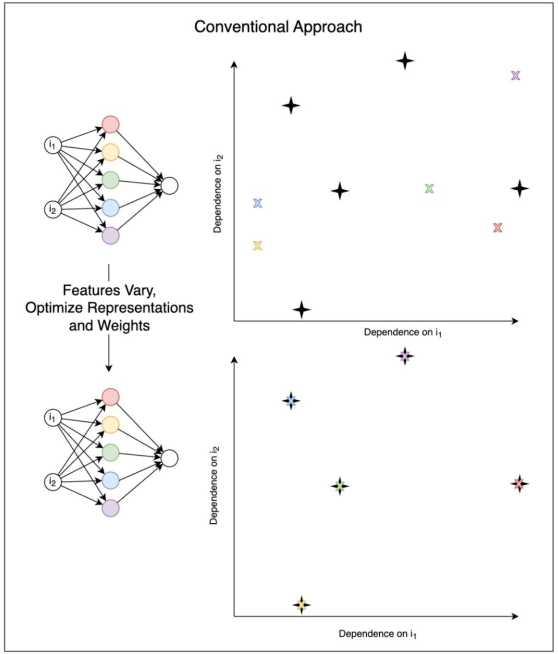
To provide intuition for why Mercury is effective in practice we will rely on a metaphor between a neural network and an enterprise data ecosystem.

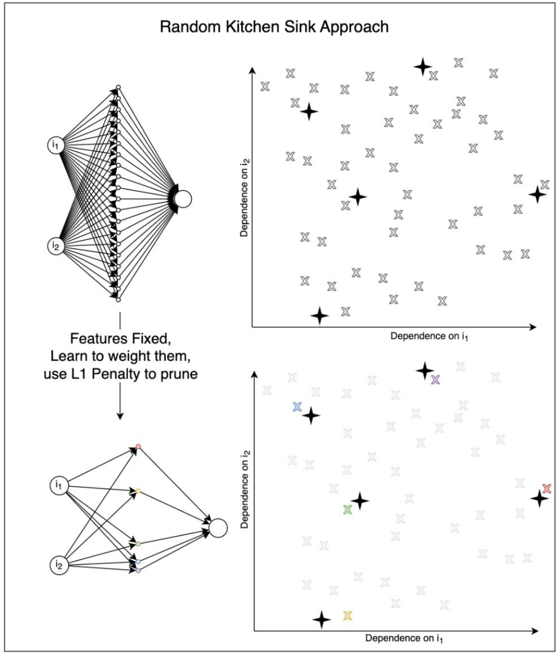




“Random Kitchen Sinks” provide intuition for the effectiveness of programmatic feature definitions.

Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in Learning. A. Rahimi and B. Recht. NIPS, 2008.





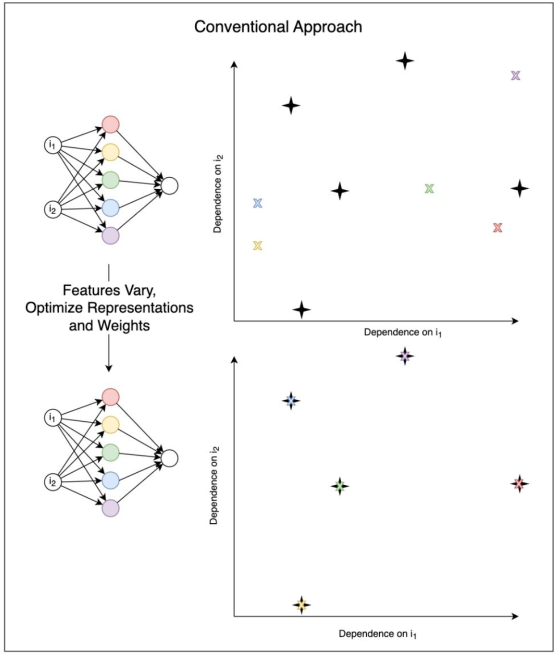
The paper shows random kitchen sinks

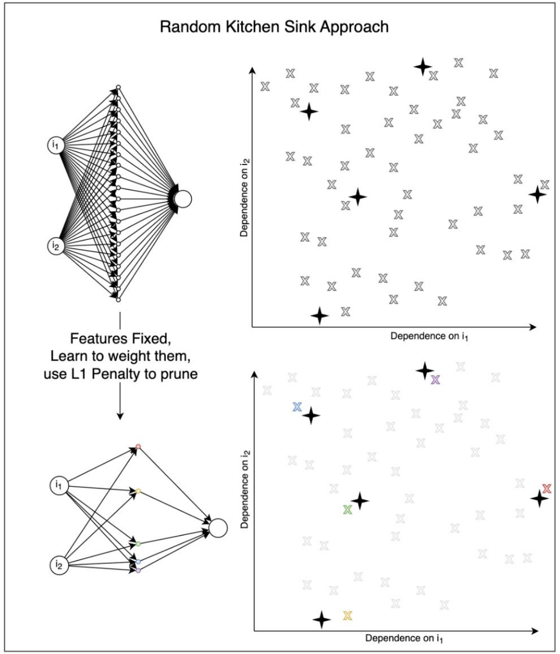
1. Approximate best case performance up to proven bound

2. Train up to ~10x faster

"Random Kitchen Sinks" provide intuition for the effectiveness of programmatic feature definitions.

Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in Learning. A. Rahimi and B. Recht. NIPS, 2008.

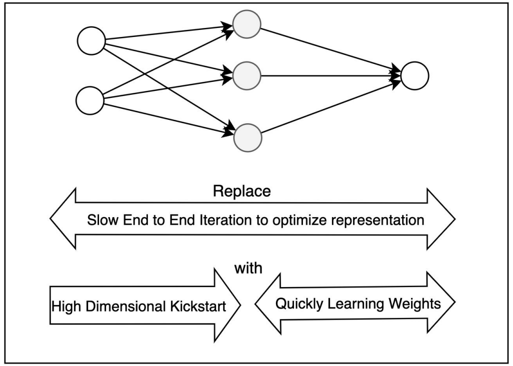




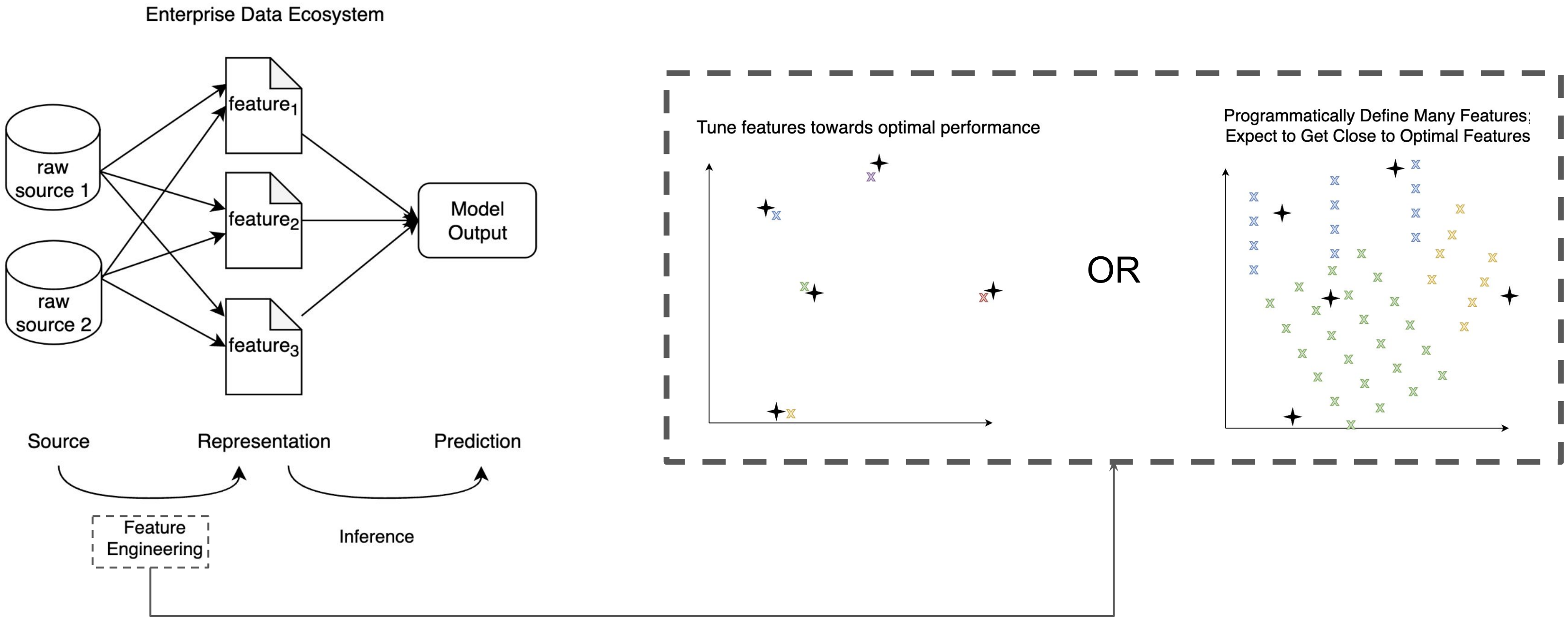
The paper shows random kitchen sinks

1. Approximate best case performance up to proven bound

2. Train up to ~10x faster

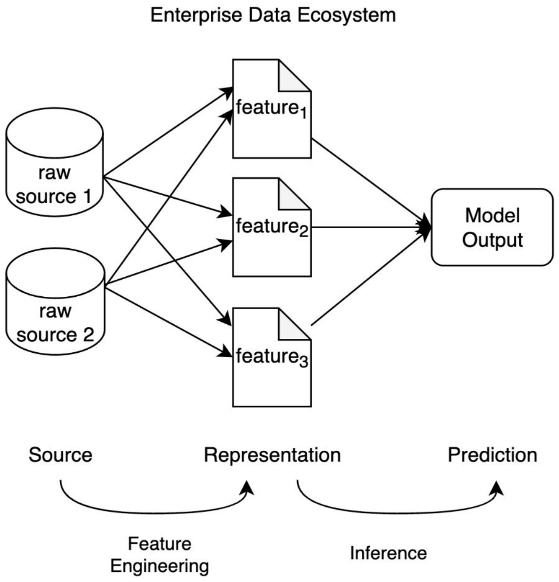


Programmatic feature definitions can be thought of as a deterministic analog of the random kitchen sink in the context of an enterprise data warehouse



FEATURE STORE SUMMIT 2023

Currently excited to explore this approach as a complement to AutoML



"AutoML" frameworks have been powerful ways of helping ML practitioners be more productive.

They provide tools to accelerate:

- Architecture / Hyperparameter Tuning
- Ensembling
- Feature Postprocessing
  - Winsorization
  - Imputation
  - Dimensionality Reduction
  - Polynomial Transformations

… but need to be given a dataframe.

How do we make more of the data ecosystem available?

wayfair

Thank you for your attention.
Questions?

And thank you to Mercury contributors:

- Jacob Baron
- Jesse Fredrickson
- Patrick Andruszkiewicz
- Vipul Dalsukrai
- Kurt Zimmer
- Minjoo Kim
- Masoum Mosmer