Feature Engineering

Creating Features that Make Machine Learning Work

Poul Petersen
CIO, BigML, Inc

Gaming the ML Performance

A Tale of Two Strategies…

- Use ML to improve performance **automatically**
  - OptiML
  - Unsupervised Feature Engineering (PCA, Topic Models, Clustering, Anomaly Detection, etc)
  - Automated feature selection
- Use domain knowledge to improve performance **manually**
  - Bespoke features (requires expertise)
  - Fusions of models
  - Manual feature selection

what is Feature Engineering

Feature Engineering: applying domain knowledge of the data to create new features that allow ML algorithms to ***work better, or to work at all***.

- This is really, really important - *more than algorithm selection*!
  - In fact, so important that BigML often does it *automatically*
- ML Algorithms have no *deeper understanding* of data
  - Numerical: have a natural order, can be scaled, etc
  - Categorical: have discrete values, etc.
- The "magic" is the ability to find patterns quickly and efficiently
- ML Algorithms only *know* what you tell/show it with *data*
  - Medical: **Kg** and **M**, but **BMI = Kg/M²** is better
  - Lending: **Debt** and **Income**, but **DTI** is better
- Intuition can be risky: remember to prove it with an evaluation!

Built-in Transformations

Date-Time Fields

DATE-TIME

2013-09-25 10:02

…	year (NUM)	month (CAT)	day (NUM)	hour (NUM)	minute (NUM)	…
…	2013	Sep	25	10	2	…
…	…	…	…	…	…	…

- Date-Time fields have a lot of information "packed" into them
- Splitting out the time components allows ML algorithms to discover time-based patterns.

Built-in Transformations

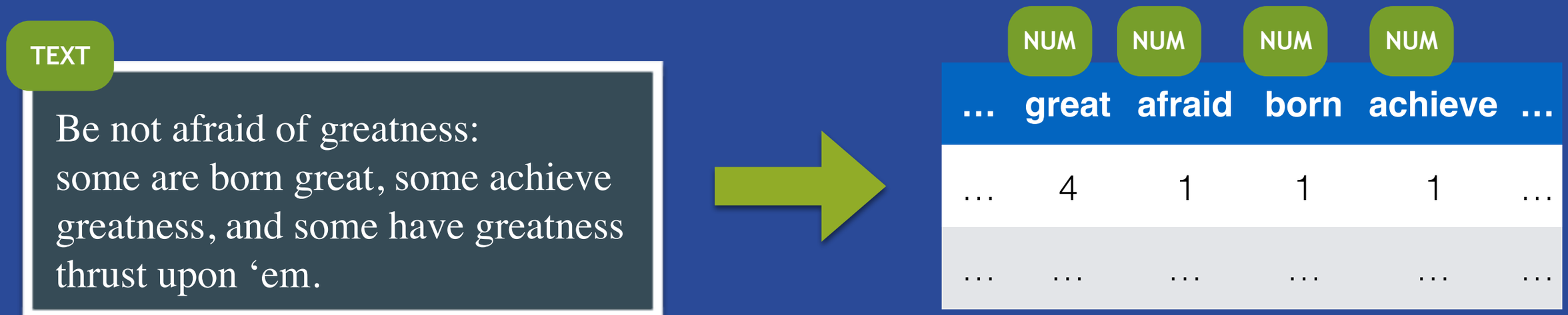
Categorical Fields for Clustering/LR

	CAT		NUM	NUM	NUM	
…	alchemy_category	…	business	health	recreation	…
…	business	…	1	0	0	…
…	recreation	…	0	0	1	…
…	health	…	0	1	0	…
…	…	…	…	…	…	…

- Clustering and Logistic Regression require numeric fields for inputs
- Categorical values are transformed to numeric vectors automatically*
- *Note: In BigML, clustering uses k-prototypes and the encoding used for LR can be configured.

Built-in Transformations

Text Fields



- Unstructured text contains a lot of potentially interesting patterns
- Bag-of-words analysis happens automatically and extracts the "interesting" tokens in the text
- Another option is Topic Modeling to extract thematic meaning

Help ML to Work Better

When text is not actually unstructured

TEXT


```
{
  “url":"cbsnews",
  "title":"Breaking News Headlines
Business Entertainment World News “,
  "body":"  news covering all the latest
breaking national and world news
headlines, including politics, sports,
entertainment, business and more.”
}
```


TEXT TEXT

title	body
Breaking News…	news covering…
…	…

- In this case, the text field has structure (key/value pairs)
- Extracting the structure as new features may allow the ML algorithm to work better

FE Demo #1

Help ML to Work at all

When the pattern does not exist

Highway Number	Direction	Is Long
2	East-West	FALSE
4	East-West	FALSE
5	North-South	TRUE
8	East-West	FALSE
10	East-West	TRUE
…	…	…

Goal: Predict principle direction from highway number

```
( = (mod (field "Highway Number") 2) 0)
```

FE Demo #2

Feature Engineering

Discretization

NUM

Total Spend
7,342.99
304.12
4.56
345.87
8,546.32

"Predict will spend $3,521 with error $1,232"

CAT

Spend Category
Top 33%
Bottom 33%
Bottom 33%
Middle 33%
Top 33%

"Predict customer will be Top 33% in spending"

FE Demo #3

Built-ins for FE

- **Discretize**: Converts a numeric value to categorical
- **Replace missing values**: fixed/max/mean/median/etc
- **Normalize**: Adjust a numeric value to a specific range of values while preserving the distribution
- **Math**: Exponentiation, Logarithms, Squares, Roots, etc
- **Types**: Force a field value to categorical, integer, or real
- **Random**: Create random values for introducing noise
- **Statistics**: Mean, Population
- Refresh Fields:
    - Types: recomputes field types. Ex: **#classes > 1000**
    - Preferred: recomputes preferred status

Flatline Add Fields

Computing with Existing Features

NUM	NUM
Debt	**Income**
10,134	100,000
85,234	134,000
8,112	21,500
0	45,900
17,534	52,000

$$\frac{\text{Debt}}{\text{Income}}$$

NUM
Debt to Income Ratio
0.10
0.64
0.38
0
0.34

(/ (field "Debt") (field "Income"))

What is Flatline?

Flatline: a domain specific language for feature engineering and programmatic filtering

- **DSL**:
  - Invented by BigML - Programmatic / Optimized for speed
  - Transforms datasets into new datasets
    - Adding new fields / Filtering
    - Transformations are written in lisp-style syntax
- **Feature Engineering**
  - Computing new fields: (/ (**field** "Debt") (**field** “Income”))
- **Programmatic Filtering**:
  - Filtering datasets according to functions that evaluate to true/false using the row of data as an input.

Flatline

- Lisp style syntax: Operators come first
  - Correct: (**+** 1 2 3) => **NOT** Correct: (1 **+** 2 **+** 3)
- Dataset Fields are first-class citizens
  - (**field** "diabetes pedigree")
- Limited programming language structures
  - **let**, **cond**, **if**, **map**, list operators, ***/+-**, etc.
- Built-in transformations
  - statistics, strings, timestamps, windows

Flatline s-expressions

Adding Simple Labels to Data

Un-Labelled Data

Name	Month - 3	Month - 2	Month - 1
Joe Schmo	123.23	0	0
Jane Plain	0	0	0
Mary Happy	0	55.22	243.33
Tom Thumb	12.34	8.34	14.56

Define "default" as missing three payments in a row

Labelled data

Name	Month - 3	Month - 2	Month - 1	Default
Joe Schmo	123.23	0	0	FALSE
Jane Plain	0	0	0	TRUE
Mary Happy	0	55.22	243.33	FALSE
Tom Thumb	12.34	8.34	14.56	FALSE

```
(= 0 (+ (abs ( f "Month - 3" ) ) (abs ( f "Month - 2")) (abs ( f "Month - 1") ) ))
```

Your Turn!

- Create a Source and Dataset from “Loan Payment”
- Engineer a new feature that is
  - True: If 3 payments in a row are zero
  - False: otherwise

Flatline s-expressions

Shock: Deviations from a Trend

date	volume	price
1	34353	314
2	44455	315
3	22333	315
4	52322	321
5	28000	320
6	31254	319
7	56544	323
8	44331	324
9	81111	287
10	65422	294
11	59999	300
12	45556	302
13	19899	301

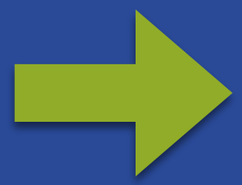
day-4	day-3	day-2	day-1	4davg
				-
			314	-
		314	315	-
	314	315	315	-
314	315	315	321	316.25
315	315	321	320	317.75
315	321	320	319	318.75

$$\frac{\text{Current - (4-day avg)}}{\text{std dev}}$$

Flatline s-expressions

Shock: Deviations from a Trend

$$\frac{\text{Current - (4-day avg)}}{\text{std dev}}$$



Current : (**field** “price”)

4-day avg: (**avg-window** “price” -4 -1)

std dev: (**standard-deviation** “price”)

```
(/  (-  ( f "price") (avg-window "price" -4, -1)) (standard-deviation "price"))
```

Advanced s-expressions

Highway isEven?

```
( = (mod (field "Highway Number") 2) 0)
```

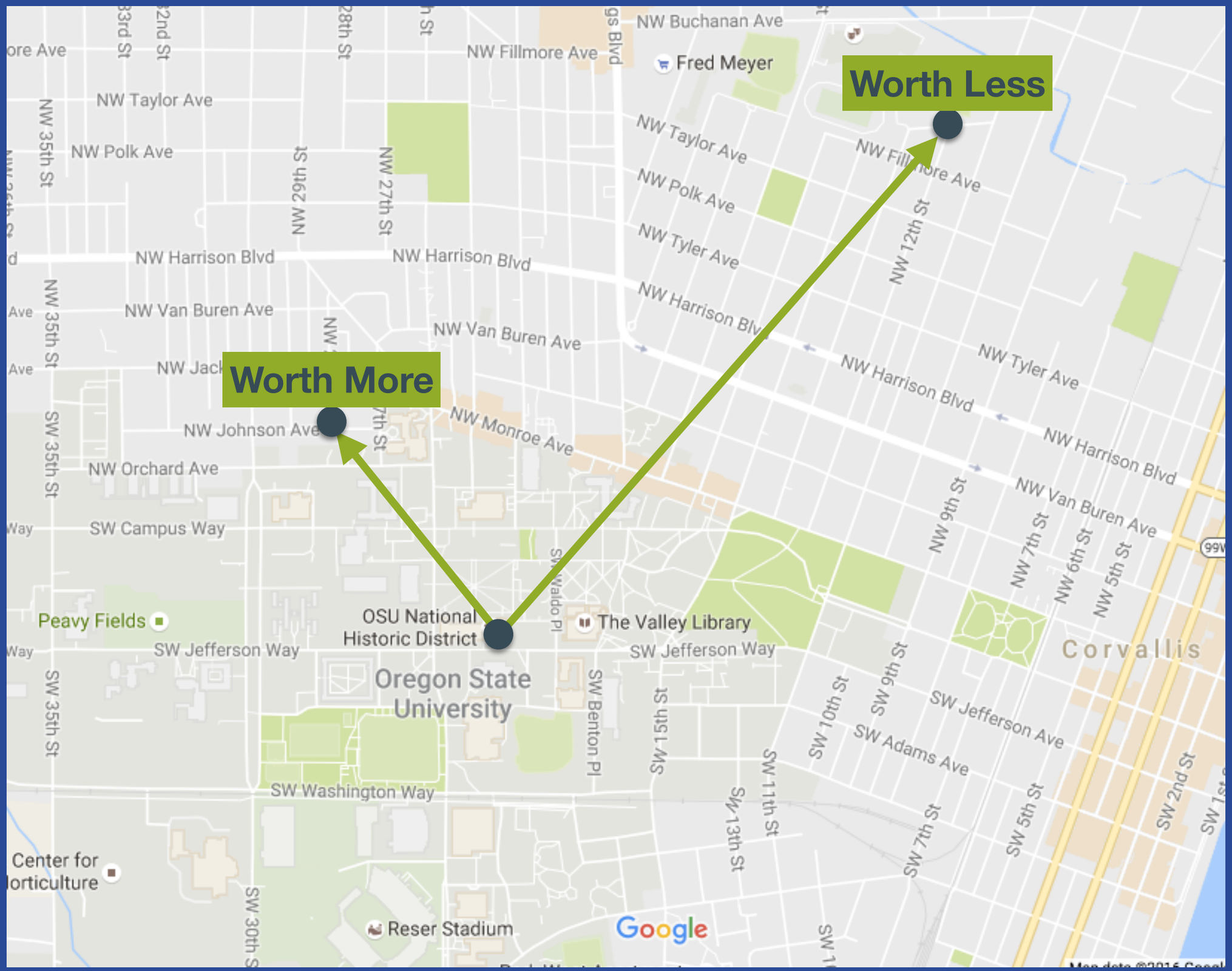
Advanced s-expressions

Moon Phase%

```
( /
   ( mod
      ( -
         ( /
            ( epoch ( field "date-field" ))
            1000
         )
         621300
      )
      2551443
   )
  2551442
)
```

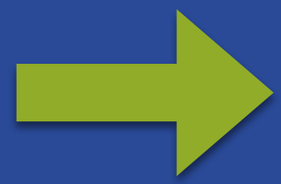
https://gist.github.com/petersen-poul/0cf5022ed1768837fe13af72b2488329

Home Price Feature



Home Price Feature

LATITUDE	LONGITUDE	REFERENCE LATITUDE	REFERENCE LONGITUDE
44.583	-123.296775	44.5638	-123.2794
44.604414	-123.296129	44.5638	-123.2794
44.600108	-123.29707	44.5638	-123.2794
44.603077	-123.295004	44.5638	-123.2794
44.589587	-123.301154	44.5638	-123.2794



Distance (m)
700
30.4
19.38
37.8
23.39

Haversine Formula

The haversine formula [edit]

For any two points on a sphere, the haversine of the central angle betwe

$$\operatorname{hav}\left(\frac{d}{r}\right) = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{ha}$$

where

- hav is the haversine function:

$$\operatorname{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{2}$$

- d is the distance between the two points great circle of the sphere; see sphe ice),
- r is the radius of the sphere,
- φ_1, φ_2: latitude of point 1 and latitude o , in radians
- λ_1, λ_2: longitude of point 1 and longitud t 2, in radians

On the left side of the equals sign $\frac{d}{r}$ is the c ngle, assuming angl sured in radians (note that φ and λ converted from radians to degrees by multiplying by $\frac{180}{\pi}$ as usual).

Solve for d by applying the inverse haversine (i e) or l e arcsine (inverse sine) function:

$$d = r\operatorname{hav}^{-1}(h) = 2r\arcsin\left(\sqrt{h}\right)$$

where h is $\operatorname{hav}(\frac{d}{r})$, or more explicitly:

$$d = 2r\arcsin\left(\sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\,\mathrm{h}}\right)$$

$$= 2r\arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

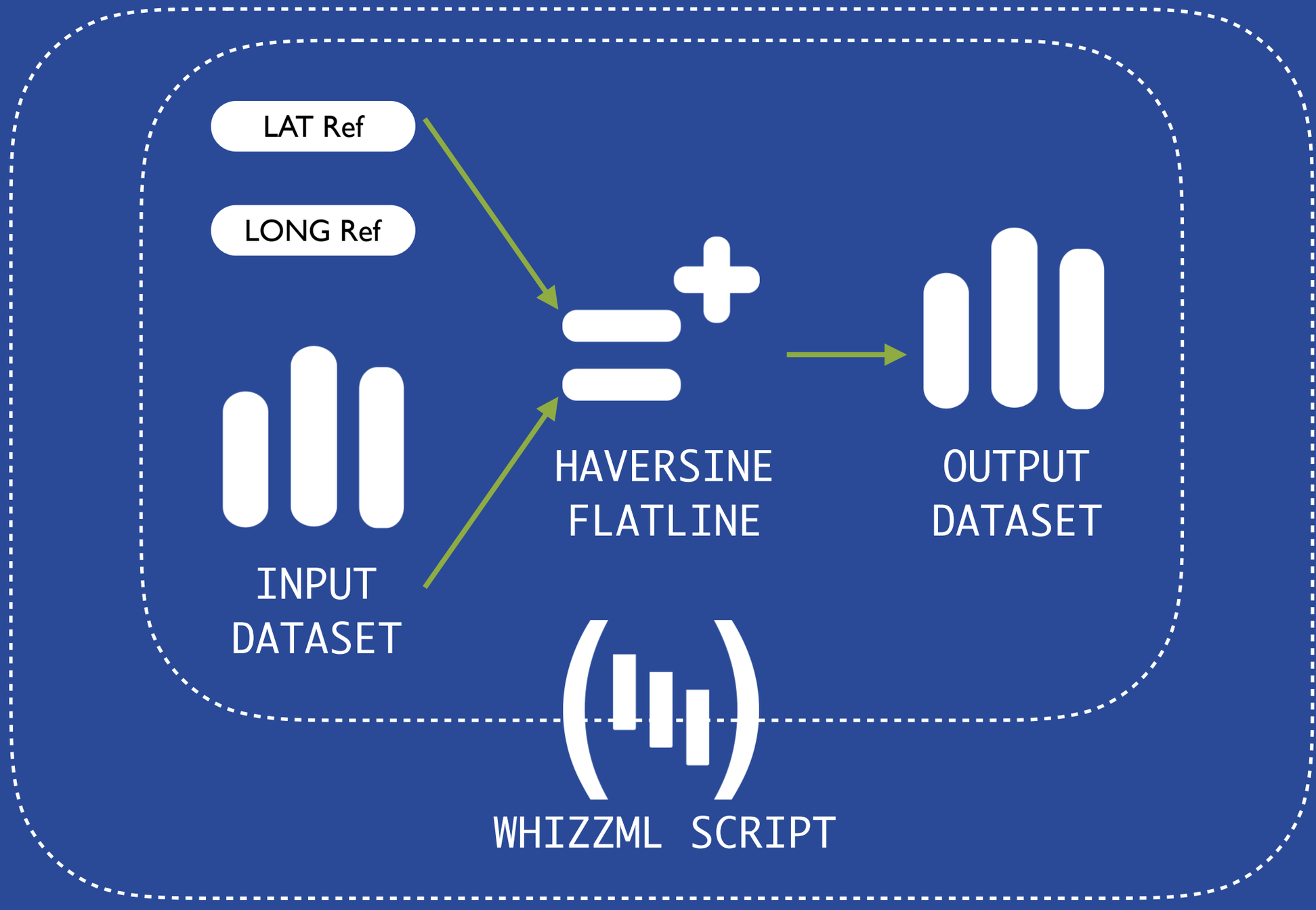
https://en.wikipedia.org/wiki/Haversine_formula

Advanced s-expressions

Distance Lat/Long <=> Ref (Haversine)

```
( let
   (   R 6371000
       latA (to-radians {lat-ref})
       latB (to-radians ( field "LATITUDE" ) )
       latD ( - latB latA )
       longD ( to-radians ( - ( field "LONGITUDE" ) {long-ref} ) )
       a ( +
            ( square ( sin ( / latD 2 ) ) )
            ( *
               (cos latA)
               (cos latB)
               (square ( sin ( / longD 2)))
            )
       )
       c ( * 2 ( asin ( min (list 1 (sqrt a)))))
   )
   ( * R c )
)
```

WhizzML + Flatline



https://bigml.com/gallery/scripts

WhizzML Gallery

Advanced s-expressions

JSON Parser???

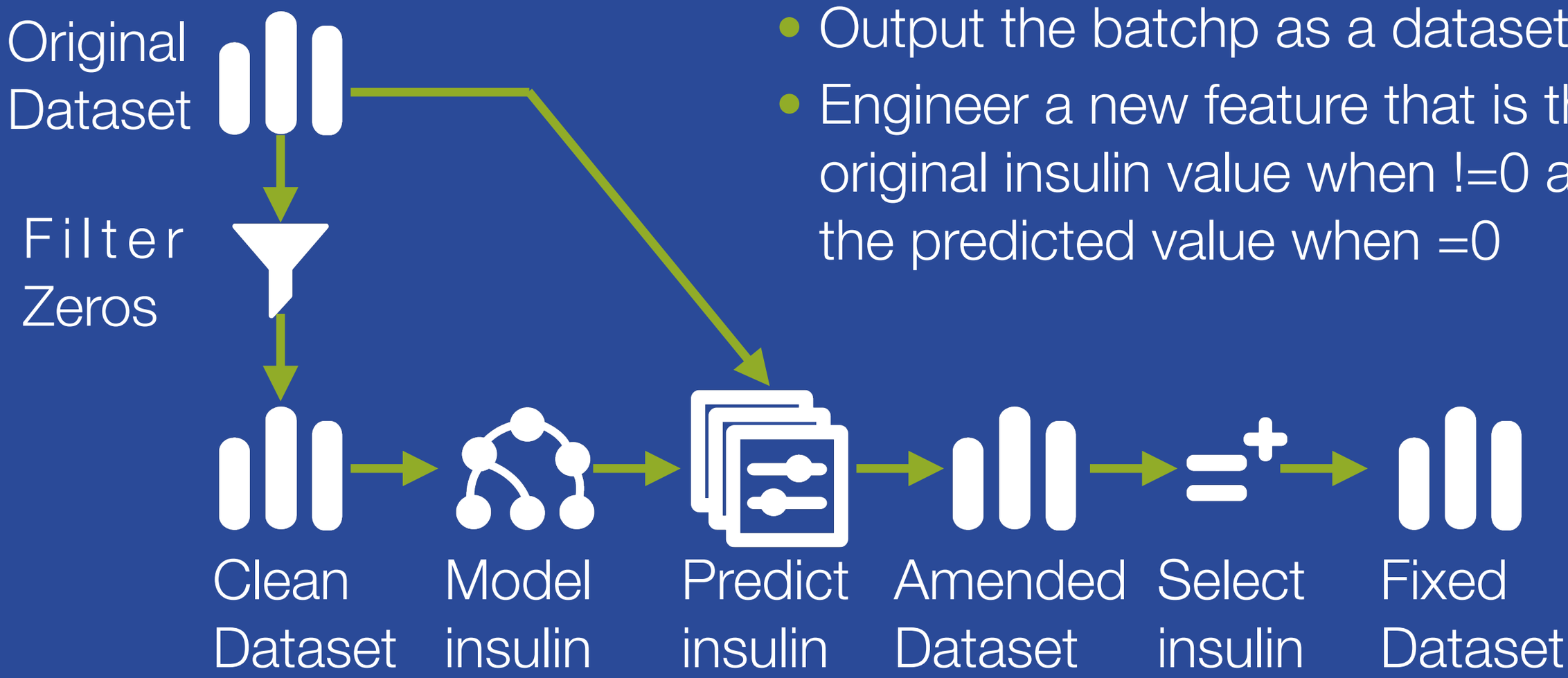
- Remember, Flatline is **not** a full programming language
- No loops
- No accumulated values
- Code executes on one row at a time and has a limited view into other rows

https://gist.github.com/petersen-poul/504c62ceaace76227cc6d8e0c5f1704b

Your Turn!

Fix Missing Values in a "Meaningful" Way

- Start with Diabetes (full dataset)
- Filter to remove insulin=0
- Build a "clean" model to predict insulin
- Batchpredict Insulin (full dataset)
- Output the batchp as a dataset
- Engineer a new feature that is the original insulin value when !=0 and the predicted value when =0




```
( if ( = (field "insulin") 0) (field "predicted insulin") (field "insulin"))
```

Feature Selection

Feature Selection

Care must be taken when creating features!

- Model Summary
    - Field Importance
- Algorithmic
    - Best-First Feature Selection
    - Boruta
- Leakage
    - Tight Correlations (AD, Plot, Correlations)
    - Test Data
    - Perfect future knowledge

Feature Selection

Leakage

- sales pipeline where step n-1 has no other outcome then step n.
- stock close predicts stock open
- churn retention: the worst rep is actually the best (correlation != causation)
- cancer prediction where one input is a doctor ordered test for the condition
- account ID predicts fraud (because only new accounts are fraudsters)


bigml