Predicting Water Pump Functionality in Tanzania

Isabel Hirama DrivenData “Mining the Waterboard” Challenge December 2017

Challenge:

Predict whether water points in Tanzania are functional, non-functional, or need repair, based on data from 59k points.

Goal:

Score in the top 25% of submissions; requires ~.75 accuracy.

Results:

BEST SCORE	CURRENT RANK	# COMPETITORS
0.7620	904	4083

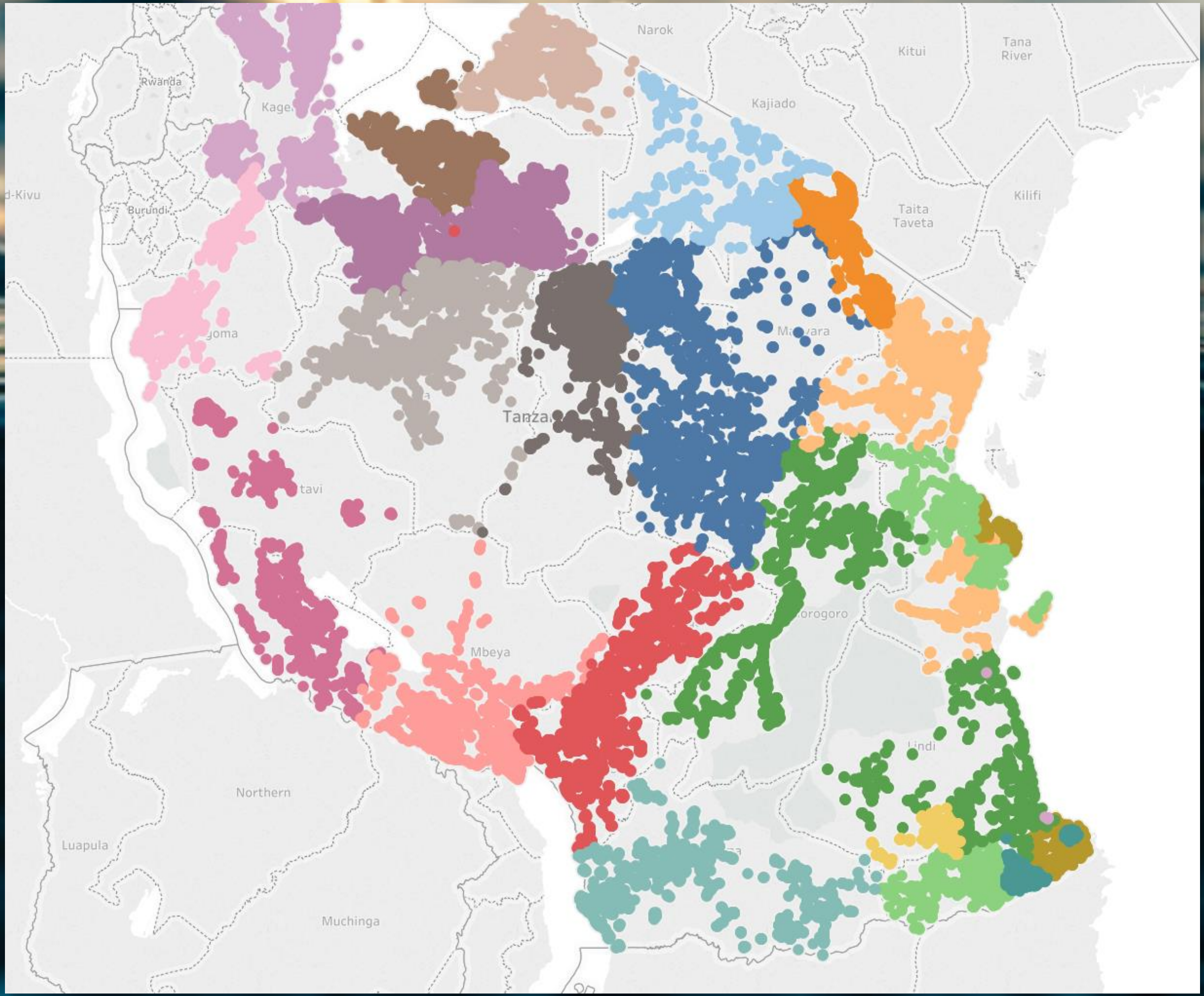
Scored in top 22% using a random forest model, which outperformed gradient boosting & support vector machines.

Process & Tools

Tableau	mySQL	R
Data exploration	Data cleaning & feature engineering	Model building and tuning
1	2	3

Data Exploration - Tableau

- Visually explored data to better understand the relationships between variables
- For example, plotted latitude & longitude on a map to determine the hierarchy between regions, districts, LGA's, wards, and subvillages
- Compared target variable trends across different features



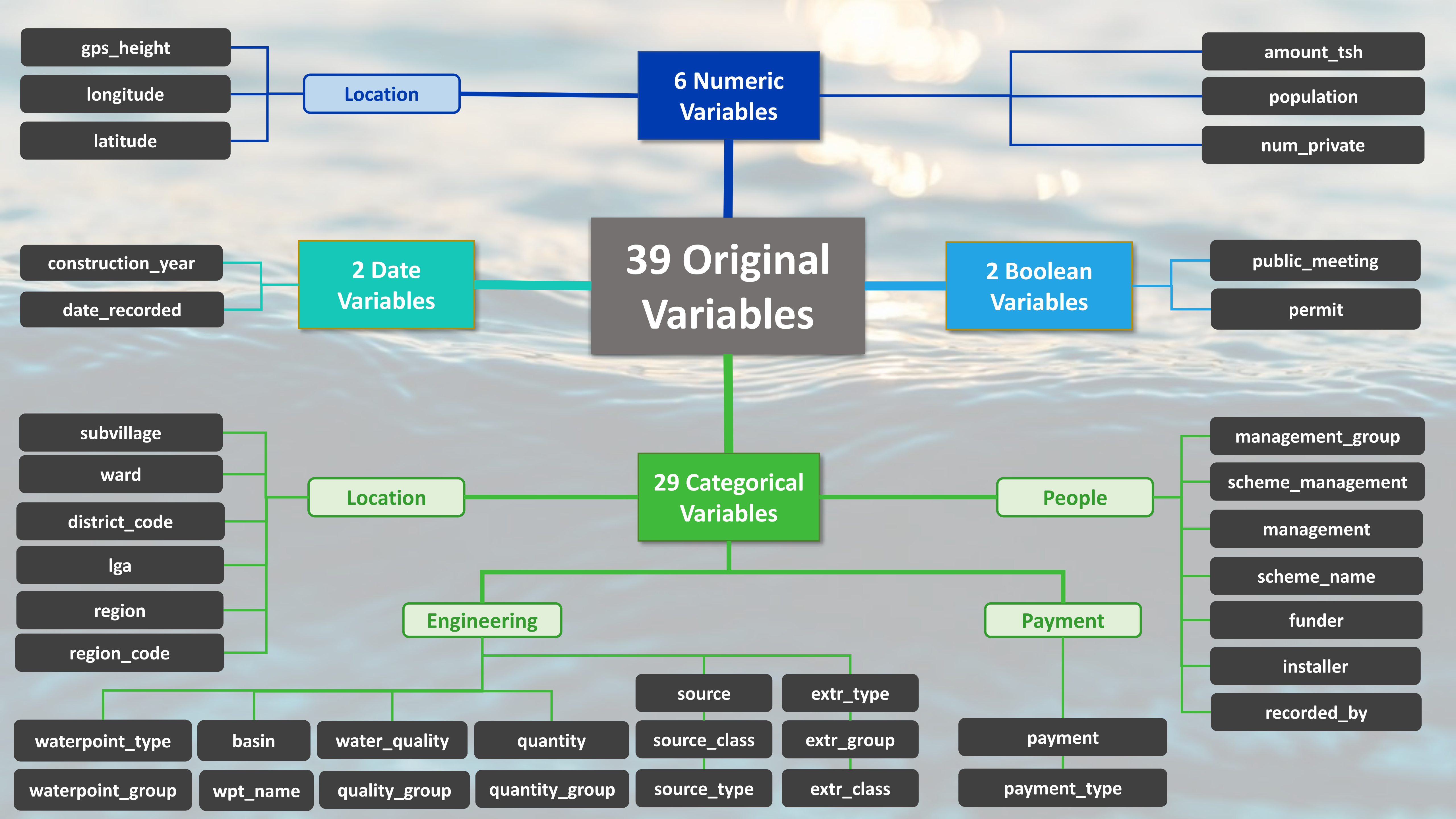
Sample visualization: waterpoints by region

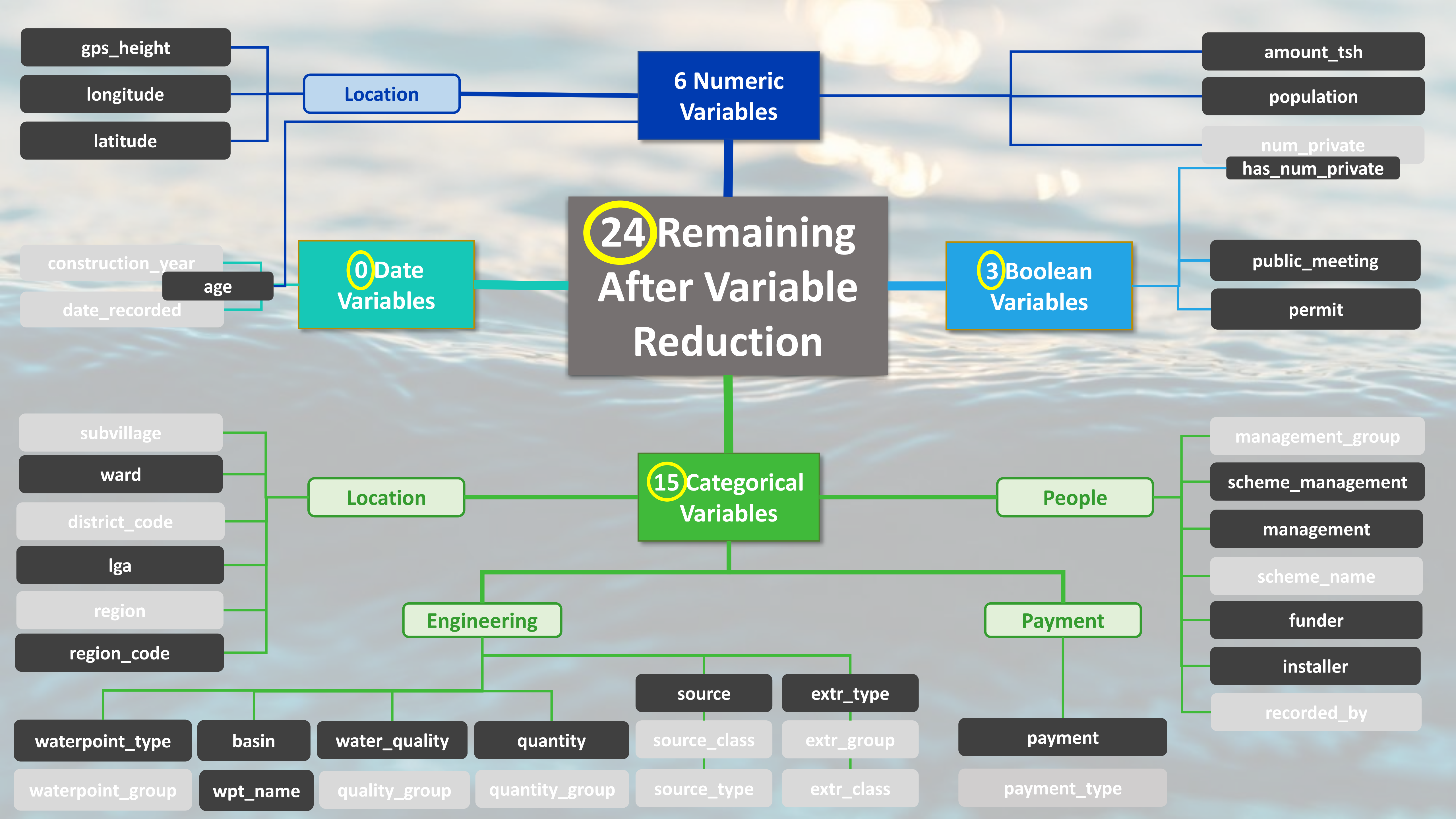
Data Wrangling - mySQL

Main goal: eliminate unhelpful variables

Variable reduction steps included:

- Replacing construction_year and date_recorded with age, the difference between the two
- Replacing num_private with a Boolean indicating whether private_num exists or is null, because 99% of values are null.
- Dropping two columns whose categories and values were exact duplicates of other columns
- Dropping several columns whose categories and values were very similar to others, and provided less information than those others


39 Original Variables
6 Numeric Variables
Location
gps_height
longitude
latitude
amount_tsh
population
num_private
2 Date Variables
construction_year
date_recorded
2 Boolean Variables
public_meeting
permit
29 Categorical Variables
Location
subvillage
ward
district_code
lga
region
region_code
People
management_group
scheme_management
management
scheme_name
funder
installer
recorded_by
Engineering
waterpoint_type
waterpoint_group
basin
wpt_name
water_quality
quality_group
quantity
quantity_group
source
source_class
source_type
extr_type
extr_group
extr_class
Payment
payment
payment_type



24 Remaining After Variable Reduction
6 Numeric Variables
Location
gps_height
longitude
latitude
amount_tsh
population
num_private
has_num_private
0 Date Variables
construction_year
age
date_recorded
3 Boolean Variables
public_meeting
permit
15 Categorical Variables
Location
subvillage
ward
district_code
lga
region
region_code
People
management_group
scheme_management
management
scheme_name
funder
installer
recorded_by
Engineering
waterpoint_type
waterpoint_group
basin
wpt_name
water_quality
quality_group
quantity
quantity_group
source
source_class
source_type
extr_type
extr_group
extr_class
Payment
payment
payment_type

Feature Engineering - mySQL

- Several categorical variables had too many levels to be handled by the R models I would use
- Five of these variables were:

ward	lga	region_code	funder	installer

- Rather than using one-hot encoding, I created new features to replace these categorical variables with meaningful numerical proxies
- Example follows

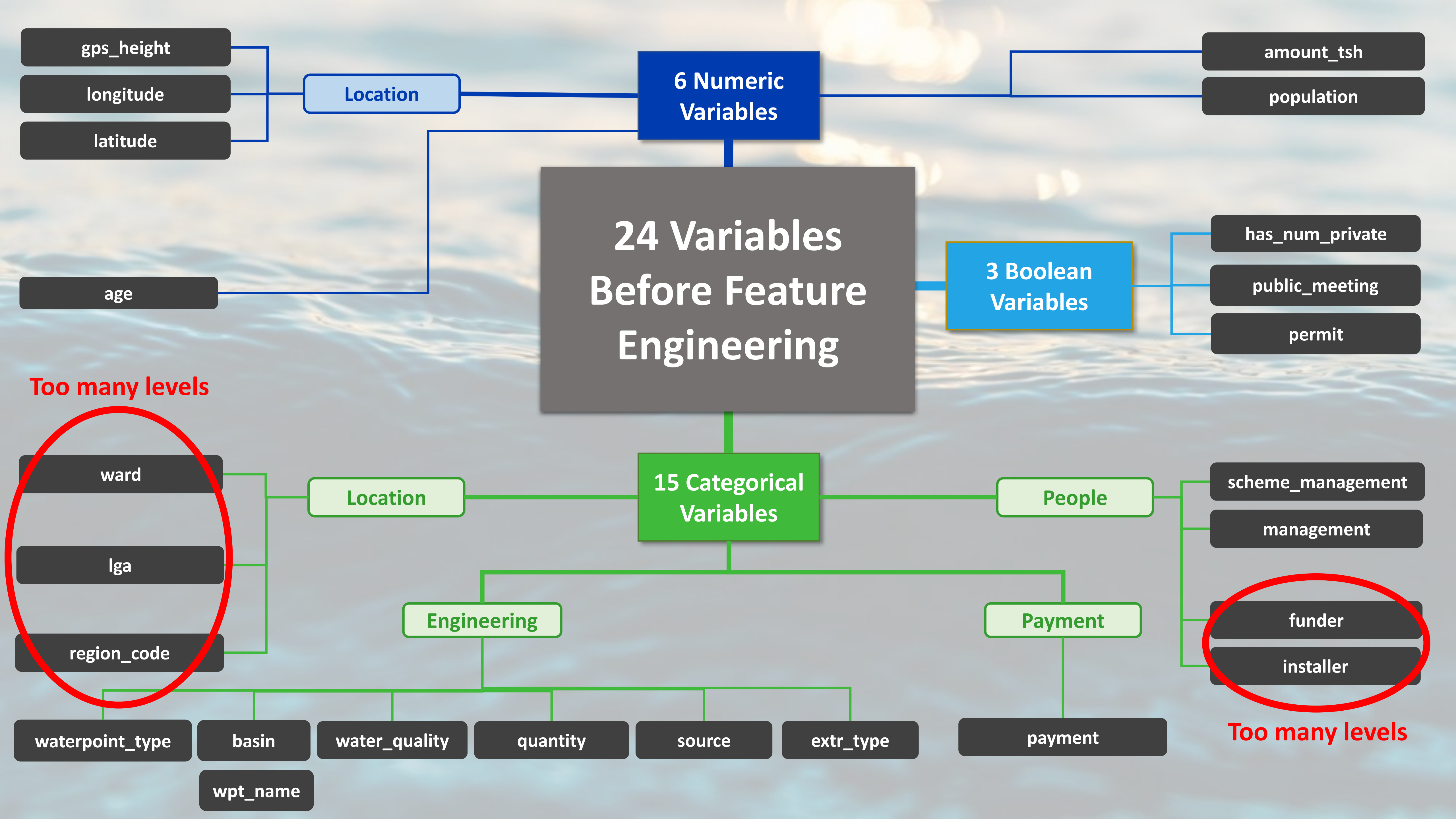
Feature Engineering - mySQL

E.g., for the "funder" variable:

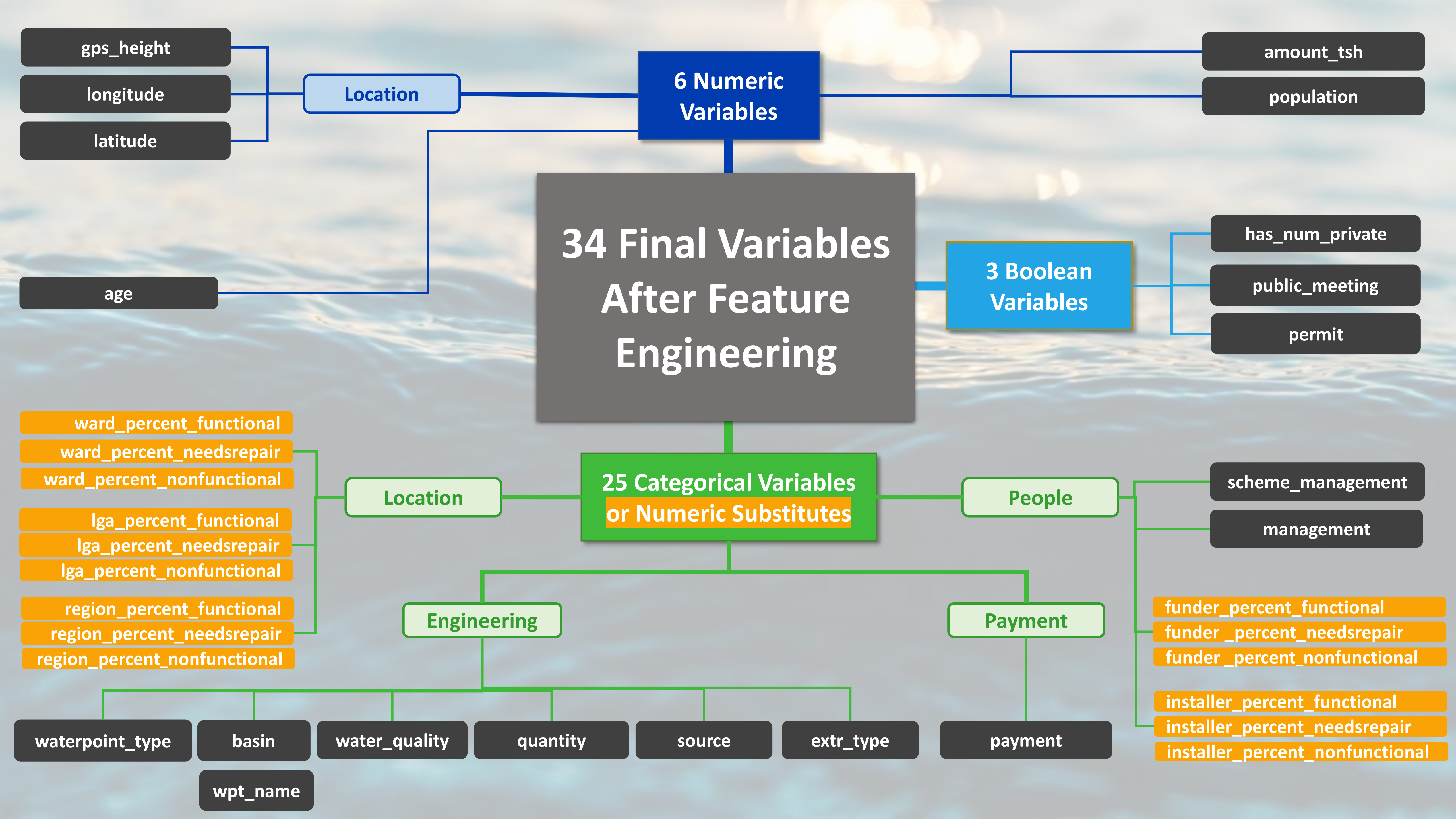
- For each funder, I calculated the % of waterpoints at each of the three functionality levels
- These became three new numeric variables:
  - funder_percent_functional
  - funder _percent_needsrepair
  - funder _percent_nonfunctional
- This was repeated for each of the five variables in question.

```
## Calculate averages
DROP TABLE IF EXISTS funder_averages;
CREATE TEMPORARY TABLE funder_averages
SELECT
    1 dummy,
    avg(funder_functional) as funder_functional,
    avg(funder_needsrepair) as funder_needsrepair,
    avg(funder_nonfunctional) as funder_nonfunctional
FROM funder_stats;

## Impute mean column score for funders with fewer than ten waterpoints
DROP TABLE IF EXISTS funder_imputed;
CREATE TEMPORARY TABLE funder_imputed
SELECT
funder, funder_functional, funder_needsrepair, funder_nonfunctional
FROM funder_total left join funder_averages using(dummy)
where funder_total.total < 5;

## Connect the rows with calculated and imputed values
DROP TABLE IF EXISTS funders;
CREATE TABLE funders
SELECT * FROM funder_stats
UNION
SELECT * from funder_imputed;
```

The above is an excerpt. See more sample code for feature engineering in [this mySQL script](#).



24 Variables Before Feature Engineering
6 Numeric Variables
Location
gps_height
longitude
latitude
age
amount_tsh
population
3 Boolean Variables
has_num_private
public_meeting
permit
15 Categorical Variables
Location
ward
lga
region_code
Too many levels
People
scheme_management
management
funder
installer
Too many levels
Engineering
waterpoint_type
basin
wpt_name
water_quality
quantity
source
extr_type
Payment
payment


34 Final Variables After Feature Engineering
6 Numeric Variables
Location
gps_height
longitude
latitude
age
amount_tsh
population
3 Boolean Variables
has_num_private
public_meeting
permit
25 Categorical Variables or Numeric Substitutes
Location
ward_percent_functional
ward_percent_needsrepair
ward_percent_nonfunctional
lga_percent_functional
lga_percent_needsrepair
lga_percent_nonfunctional
region_percent_functional
region_percent_needsrepair
region_percent_nonfunctional
People
scheme_management
management
funder_percent_functional
funder _percent_needsrepair
funder _percent_nonfunctional
installer_percent_functional
installer_percent_needsrepair
installer_percent_nonfunctional
Engineering
waterpoint_type
basin
wpt_name
water_quality
quantity
source
extr_type
Payment
payment

Model Building - R

- Created three models, with limited manual tuning due to time constraints
- The same variables were used for each model

```
# Create random forest model
rf = randomForest(status_group~., data=train, mtry= 8 , ntree = 1000, importance =TRUE)
```

```
# Create gradient boosting model
boost=gbm(status_group~.,data=train,n.trees=1000,interaction.depth=16, n.minobsinnode=10, bag.fraction = .3, cv.folds = 5)
```

```
# Create SVM model
svm.radial = svm(status_group~., data=train, kernel='radial', cost=8)
```

- See full code for model creation and evaluation in this Jupyter notebook.

Results - Model Accuracy Rates

Random Forest	Gradient Boosting	Support Vector Machine
0.7620	0.7578	0.7487

Best performing model was random forest, followed by gradient boosting, then support vector machine.

Limitations & Future Directions

- Train on full dataset: In the interest of time, only trained on 10k observations, rather than the full 59k.
- Try clustering based on latitude & longitude data for more feature engineering
- Try one-hot encoding rather than numerical proxy engineering
- Tune more extensively, using tools/approaches such as caret, grid search, random search, rather than manually
- Reframe problem: do we really want a simple prediction, or would a risk score be more helpful?
- Incorporate urgency: are some waterpoints more crucial than others (e.g. if there are no others nearby)?

Thank you for reading.

Isabel Hirama Isabel.Hirama@gmail.com 07804 507 688