# Tanzanian Water Pumps Case

repair and replace strategy

Alex de Vries 6 Sept 2021

# Case Overview

- The main water source for many Tanzanian citizens is a water pump.
- Recently data for all of the water pumps has been collected, to get an idea of how many of them are still working.
- Which pumps are most likely to be non-functional, so that the repair efforts can be optimized.

What is the best repair and replace strategy, that **minimizes time/cost** and **optimizes water access**?

# Table of Content

Data analysis

Results

Considerations

1) Dataset Overview

2) Data preparation

3) modeling

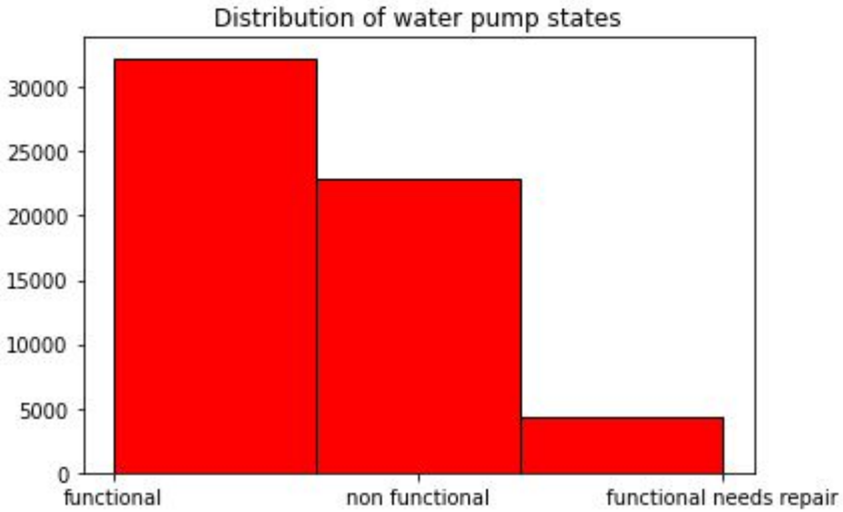
4) Conclusions

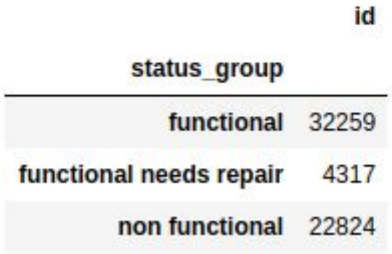
5) Advice

6) Discussion

# Dataset Overview

- Training dataset size: 54900
- Test dataset size: 14850
- Number of instances: 54900
- Number of Attributes: 40 (including id)
- Latest constructed water pump 2013

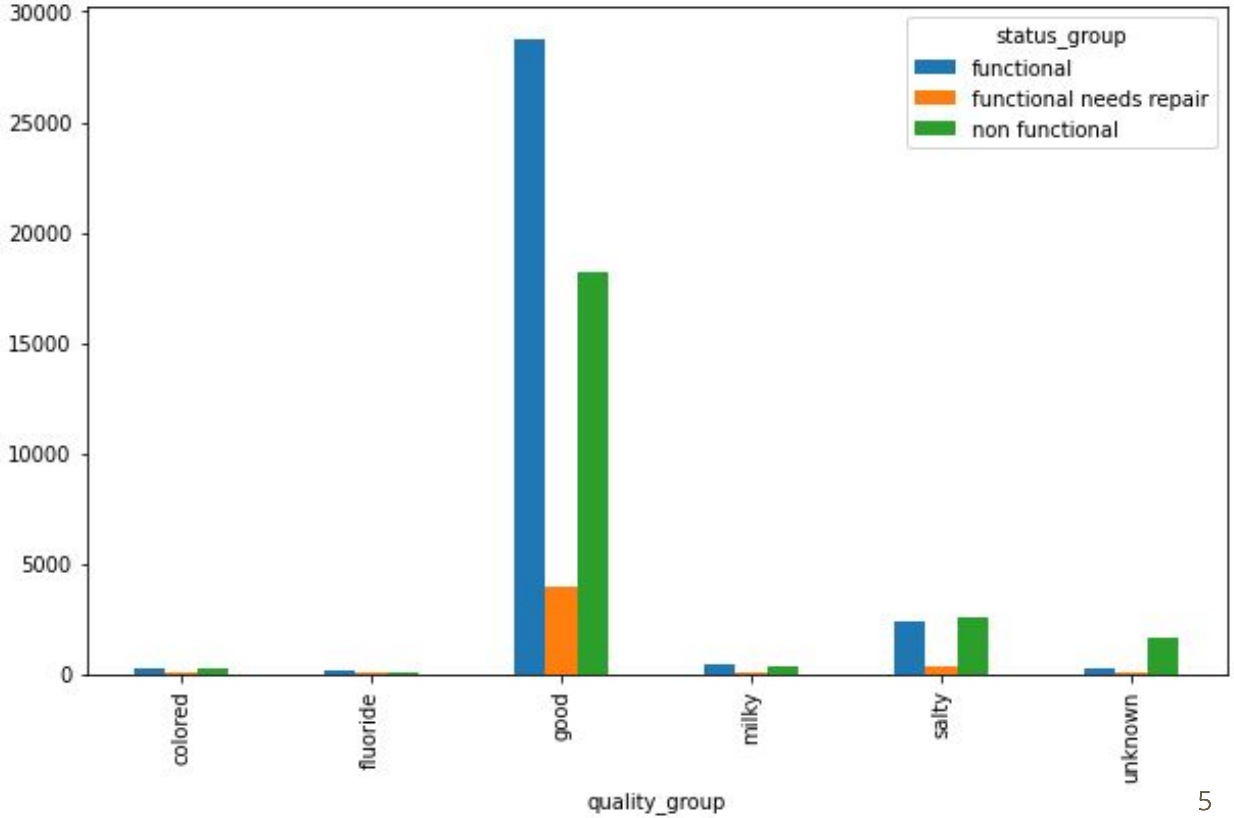




| | id |
|---|---|
| **status_group** | |
| **functional** | 32259 |
| **functional needs repair** | 4317 |
| **non functional** | 22824 |

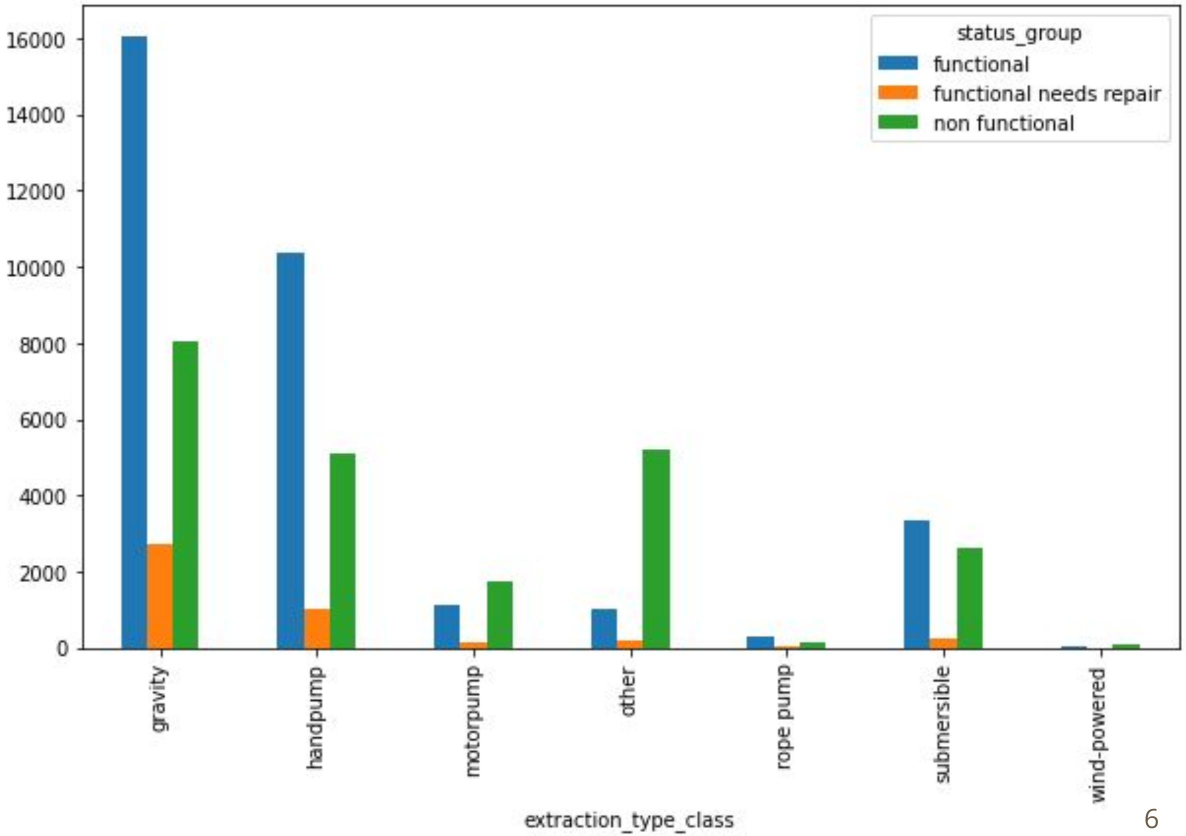
# Dataset Overview

- There are a total of **23,000 non-functional** wells.
-
- Most wells have good water quality but around 17,000 with good water quality are non-functional.



# Dataset Overview

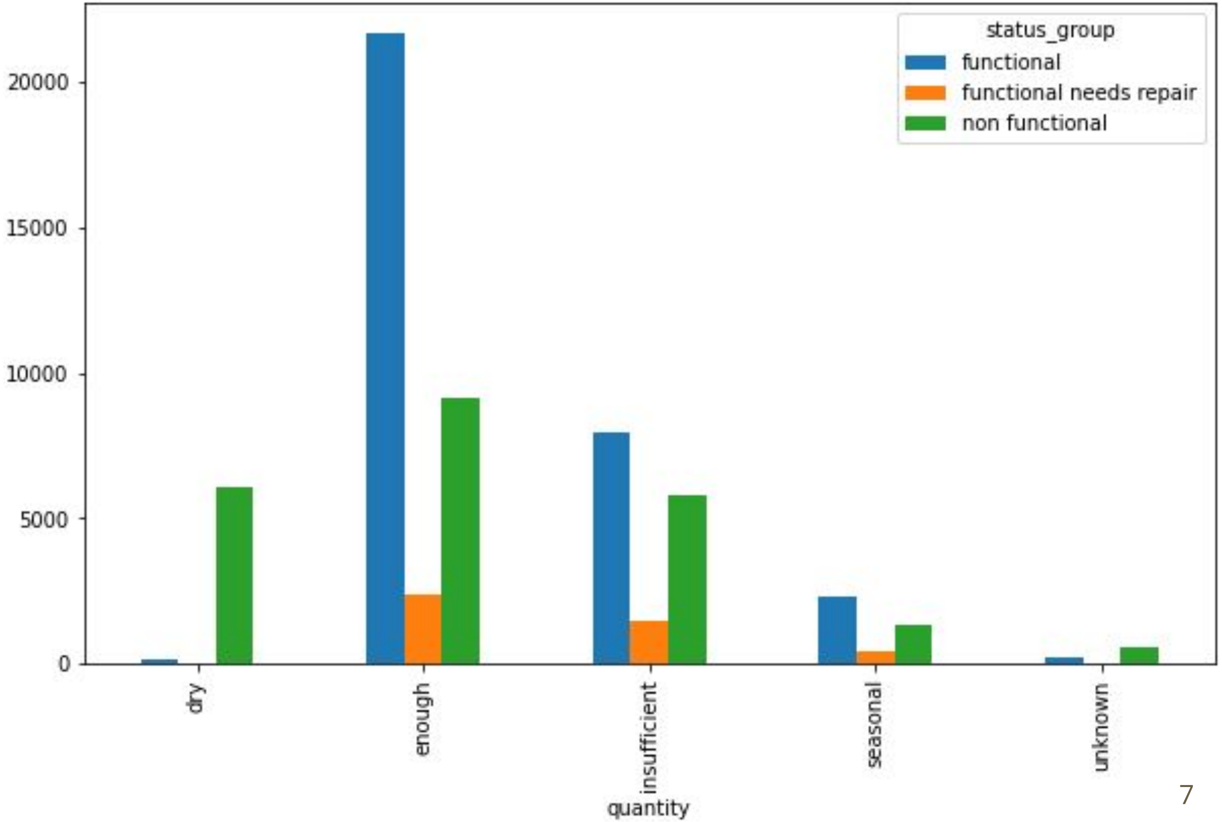
**Motorpumps** and **submersible** pumps tend to be non functional more often than other wells. This could be due to maintenance requirements.



# Dataset Overview

- About **9,000 wells** that are currently non-functional have enough water availability. However, they can not be accessed.

- It is might be best to focus on wells with **enough water quantity**, **good water quality** and **easy to maintain** pumps.



# Data Preparation (preprocessing)

- Removing null values
- Keep feature with few unique values

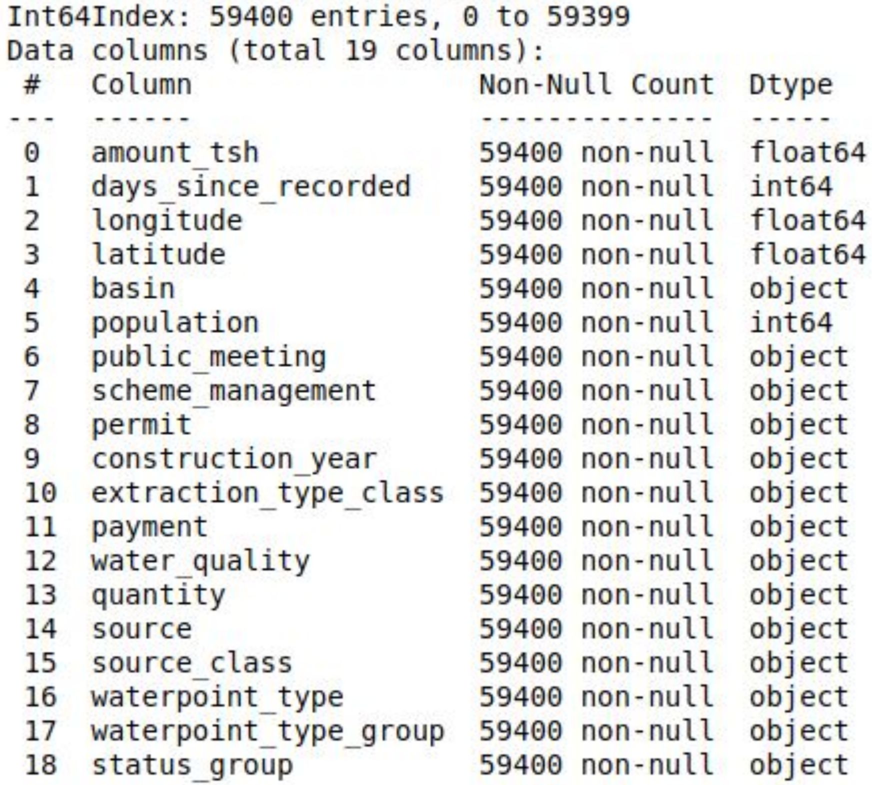
- Make string feature categorical (construction year -> construction decade)

| Feature | Unique values |
|---|---|
| funders | 1897 |
| villages | 19287 |
| installer | 2145 |
| scheme_name | 2696 |
| public_meeting | 2 |
| permit | 2 |
| scheme_management | 12 |

# Data Preparation (feature selection)

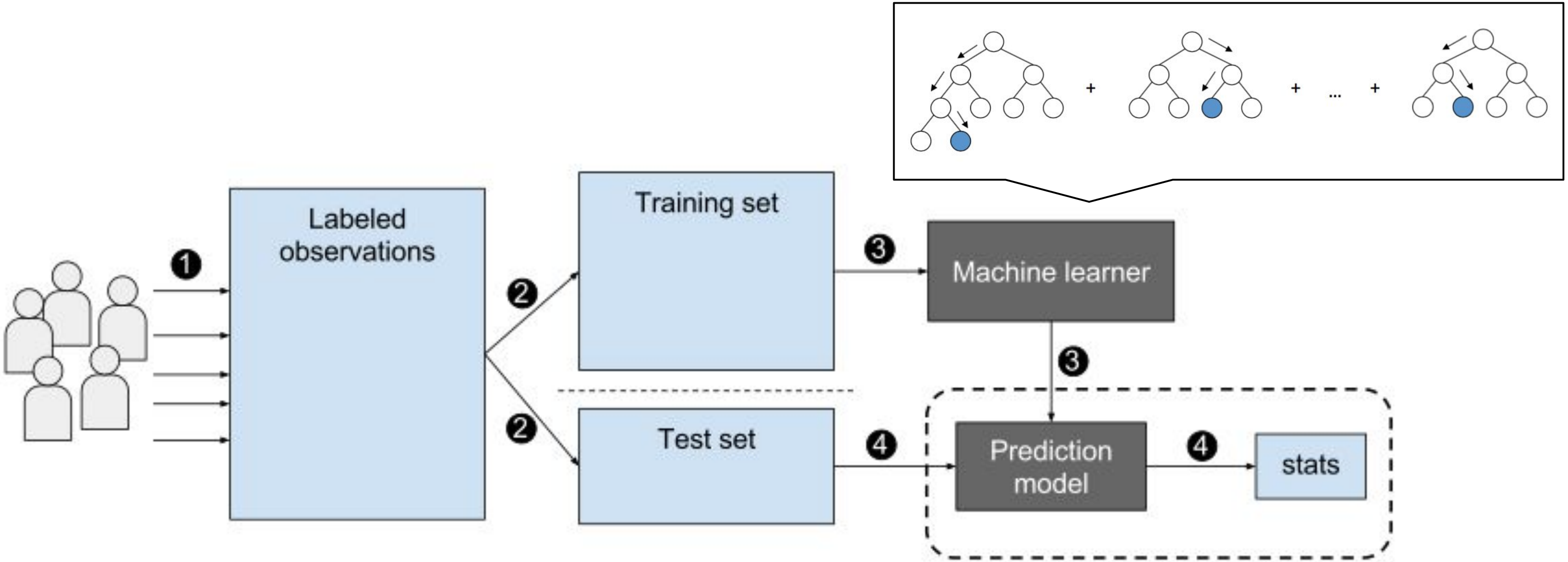
- **Removing duplicate features** (example: payment vs payment_type)

- **Removing highly related features** (geo data such as: district_code, region code, gps_height)

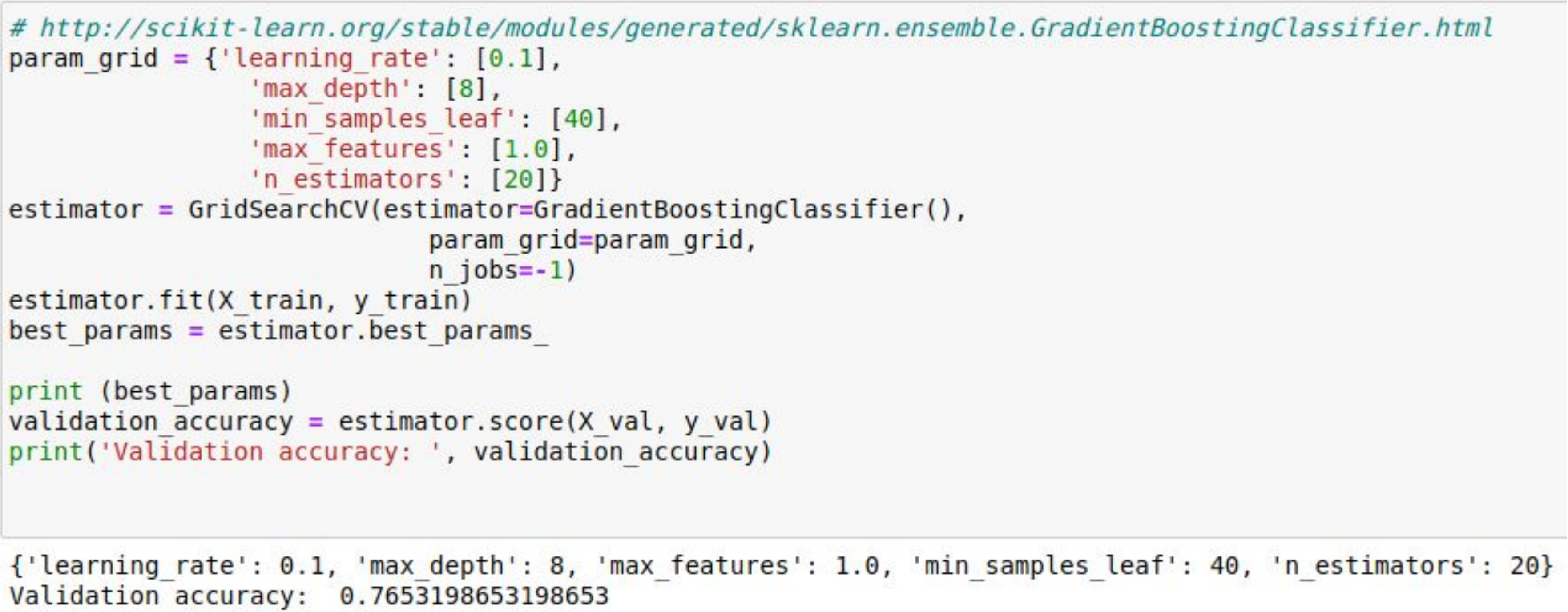


```
Int64Index: 59400 entries, 0 to 59399
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   amount_tsh             59400 non-null  float64
 1   days_since_recorded    59400 non-null  int64  
 2   longitude              59400 non-null  float64
 3   latitude               59400 non-null  float64
 4   basin                  59400 non-null  object 
 5   population             59400 non-null  int64  
 6   public_meeting         59400 non-null  object 
 7   scheme_management      59400 non-null  object 
 8   permit                 59400 non-null  object 
 9   construction_year      59400 non-null  object 
 10  extraction_type_class  59400 non-null  object 
 11  payment                59400 non-null  object 
 12  water_quality          59400 non-null  object 
 13  quantity               59400 non-null  object 
 14  source                 59400 non-null  object 
 15  source_class           59400 non-null  object 
 16  waterpoint_type        59400 non-null  object 
 17  waterpoint_type_group  59400 non-null  object 
 18  status_group           59400 non-null  object 
```

# Gradient boosting algorithm, what is it?



# Gradient boosting results



```
# http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
param_grid = {'learning_rate': [0.1],
                'max_depth': [8],
                'min_samples_leaf': [40],
                'max_features': [1.0],
                'n_estimators': [20]}
estimator = GridSearchCV(estimator=GradientBoostingClassifier(),
                            param_grid=param_grid,
                            n_jobs=-1)
estimator.fit(X_train, y_train)
best_params = estimator.best_params_

print (best_params)
validation_accuracy = estimator.score(X_val, y_val)
print('Validation accuracy: ', validation_accuracy)
```

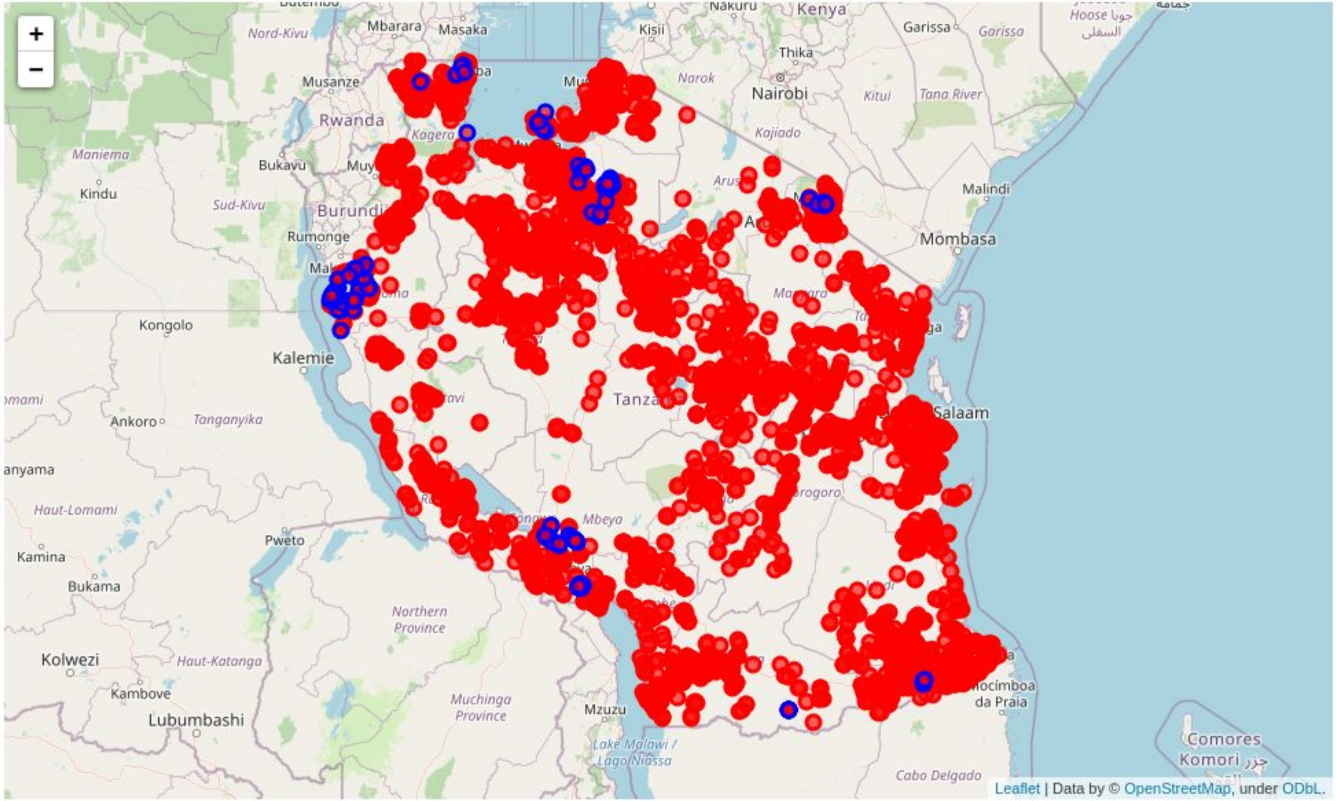
```
{'learning_rate': 0.1, 'max_depth': 8, 'max_features': 1.0, 'min_samples_leaf': 40, 'n_estimators': 20}
Validation accuracy:  0.7653198653198653
```

# Conclusion

Need repair (blue)

Non functional (red)

> 5000 pumps



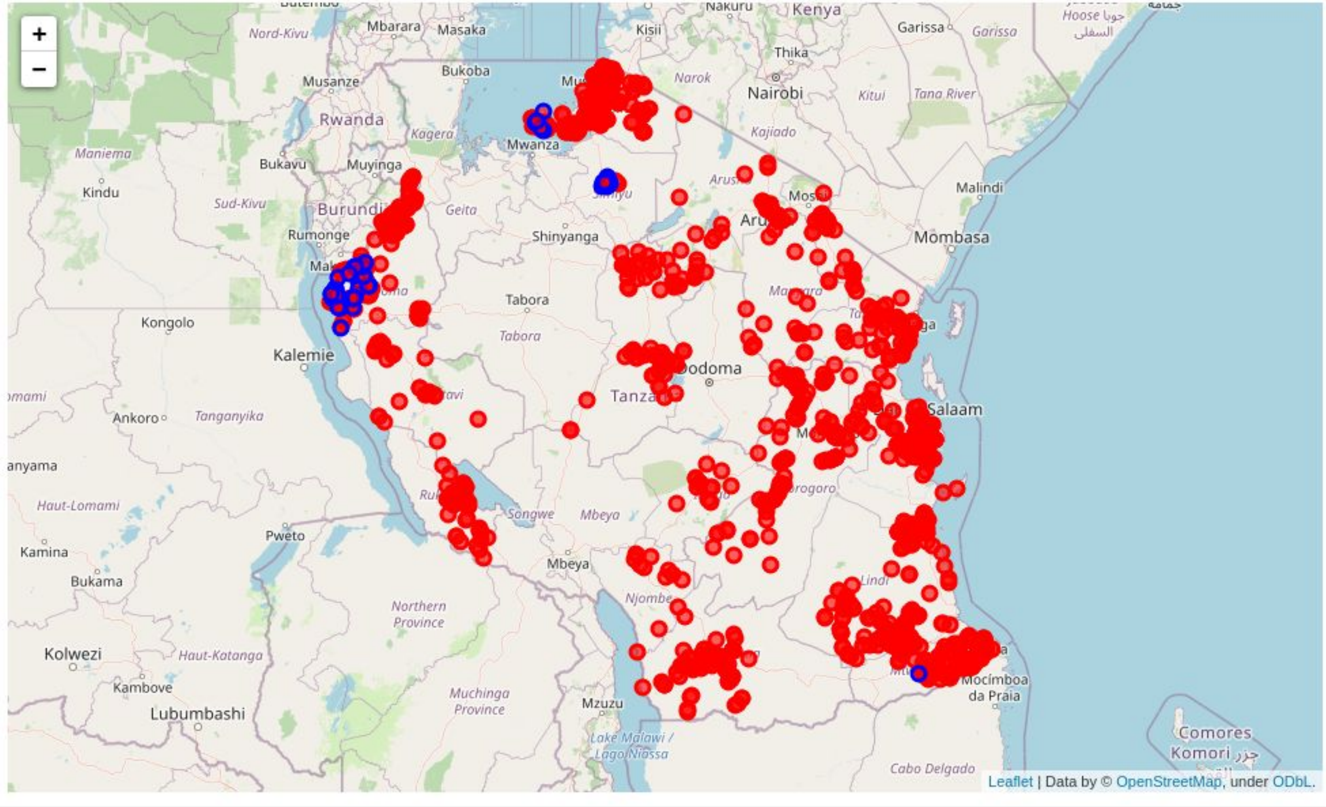
# Conclusion

Need repair (blue)

Non functional (red)

Population >= 170

~ 1500 pumps



# Advice

What is the best repair and replace strategy, that **minimizes time/cost** and **optimizes water access**?

- Focus on wells with **enough water quantity**, **good water quality** and **easy to maintain** pumps.
- Focus on densely populated areas in the corners of the country.

# Discussion

- Compare distance of non functional pumps to functional pumps
- Group geo data using kmeans
- Improve feature selection process

# Questions

Alex de Vries

alexthevries@gmail.com

linkedin.com/in/alex-de-vries-nl

# Thank you for your time

Alex de Vries

alexthevries@gmail.com

linkedin.com/in/alex-de-vries-nl