IBM ADVANCED DATA SCIENCE

Luciano Guerra

OUTLINES

Water Plant Predictive Model.

LUCIANOGUERRA

OUTLINES

Water Plant Predictive Model.

- Data Set / Use Case
- Data Exploration
- Model Definition / Training
- Model Selection
- Model Evaluation

DATASET

Water Plant Predictive Model.

LUCIANOGUERRA

DATASET

EL SUBTÍTULO VA AQUÍ

- Initial Data (3 CSV Files):
  - The dataset comes in 2 files
  - One for the predictive variables (the X in a model).
  - The other for the target variable (the Y in a model).
- 59,400 instances
- 43 features
- 19.9MB

DATASET

File Descriptions

- train_features.csv : the training set features
- train_labels.csv : the training set labels
- test_features.csv : the test set features

Labels

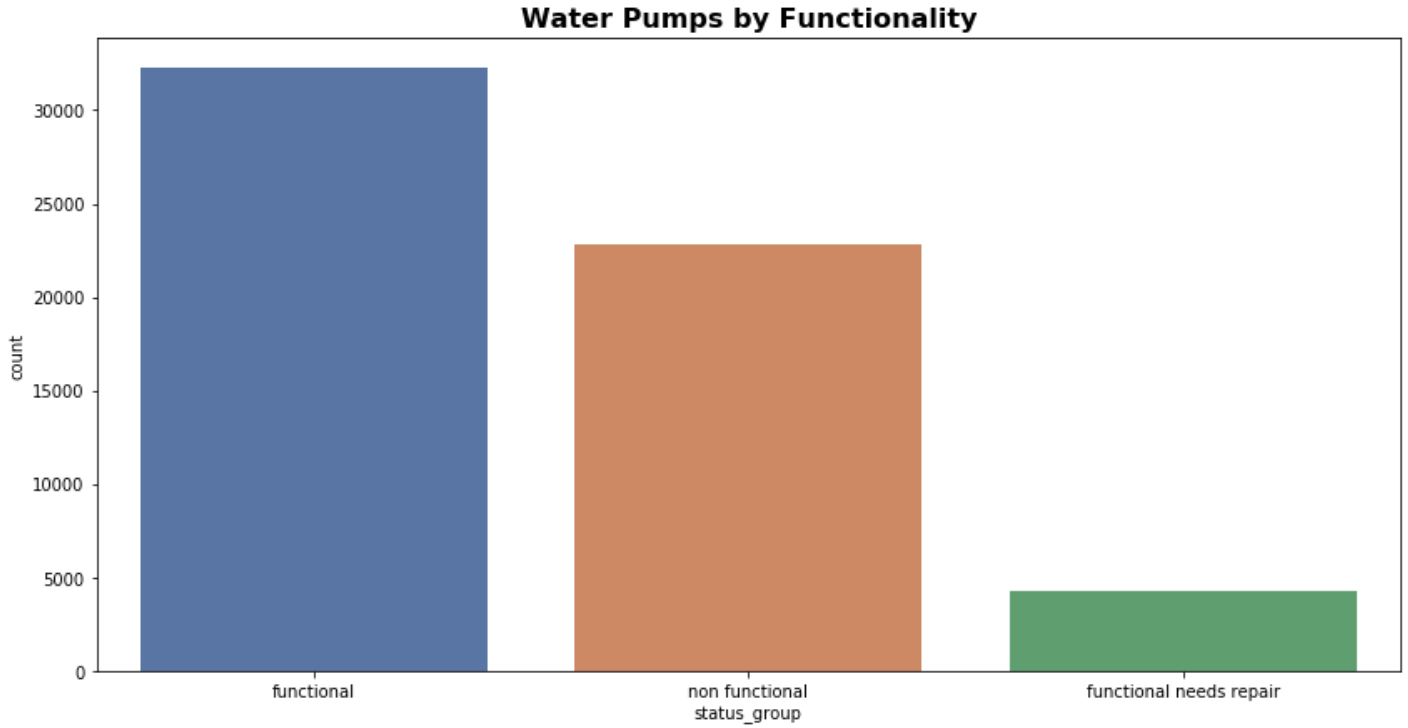
- functional : the waterpoint is operational and there are no repairs needed
- functional needs repair : the waterpoint is operational, but needs repairs
- non functional : the waterpoint is not operational

Features

- amount_tsh : Total static head (amount water available to waterpoint)
- date_recorded : The date the row was entered
- funder : Who funded the well
- gps_height : Altitude of the well
- installer : Organization that installed the well
- longitude : GPS coordinate
- latitude : GPS coordinate
- wpt_name : Name of the waterpoint if there is one
- num_private :
- basin : Geographic water basin
- subvillage : Geographic location
- region : Geographic location
- region_code : Geographic location (coded)
- district_code : Geographic location (coded)
- lga : Geographic location
- ward : Geographic location
- population : Population around the well
- public_meeting : True/False
- recorded_by : Group entering this row of data
- scheme_management : Who operates the waterpoint
- scheme_name : Who operates the waterpoint
- permit : If the waterpoint is permitted
- construction_year : Year the waterpoint was constructed
- extraction_type : The kind of extraction the waterpoint uses
- extraction_type_group : The kind of extraction the waterpoint uses
- extraction_type_class : The kind of extraction the waterpoint uses
- management : How the waterpoint is managed
- management_group : How the waterpoint is managed
- payment : What the water costs
- payment_type : What the water costs
- water_quality : The quality of the water
- quality_group : The quality of the water
- quantity : The quantity of water
- quantity_group : The quantity of water
- source : The source of the water
- source_type : The source of the water
- source_class : The source of the water
- waterpoint_type : The kind of waterpoint
- waterpoint_type_group : The kind of waterpoint

DATASET

	id	district_code	gps_height_new	amount_tsh_new	latitude	longitude	subvillage	lga	ward	installer	population	funder	region_code	basin	pump_age	region
10708	20634	13	-9	100	-8.498295	39.25864	Njia Nne	Kilwa	Tingi	Artisan	230	Private	80	Ruvuma / Southern Coast	(5,9]	Lindi
3031	69974	7	1311	500	-3.32659969	36.89806797	Mtoni	Meru	Maji ya Chai	Community	79	Government Of Tanzania	2	Pangani	(5,9]	Arusha
7531	5458	2	1743	500	-1.2646787	31.82863656	Kishoju	Bukoba Rural	Nyakato	DWE	150	World Vision	18	Lake Victoria	(9,13]	Kagera
3555	45028	2	1030	20	-3.472859	36.80627	Marurani Juu	Arusha Rural	Nduruma	DWE	100	Government Of Tanzania	2	Pangani	(9,13]	Arusha
3482	53202	2	430	500	-7.23311762	37.77820772	Ng'Wambe	Morogoro Rural	Kolero	DWE	100	Tanza	5	Wami / Ruvu	(1,5]	Morogoro
9820	33174	7	1088	500	-8.218651	34.78642	Majengo A	Mbarali	Madibira		150	Unknown	12	Rufiji	(9,13]	Mbeya
7229	60173	30	1827	500	-2.44823172	30.81677161	Nyakahanga	Ngara	Rusumo	VWC	150	Migration	18	Lake Victoria	(9,13]	Kagera
12971	58165	4	295	500	-9.02440855	36.01808504	Uzunguni	Ulanga	Ngoheranga	DWE	500	Dhv	5	Rufiji	(13,17]	Morogoro
232	33882	2	1245	500	-5.017148	35.01429	Dodoma	Singida Rural	Siuyu		1	Unknown	13	Internal	(1,5]	Singida
9498	63024	6	1314	500	-9.10943	32.7617	Maweni	Mbozi	Ihanda	KKKT	150	Kkkt	12	Lake Rukwa	(17,21]	Mbeya



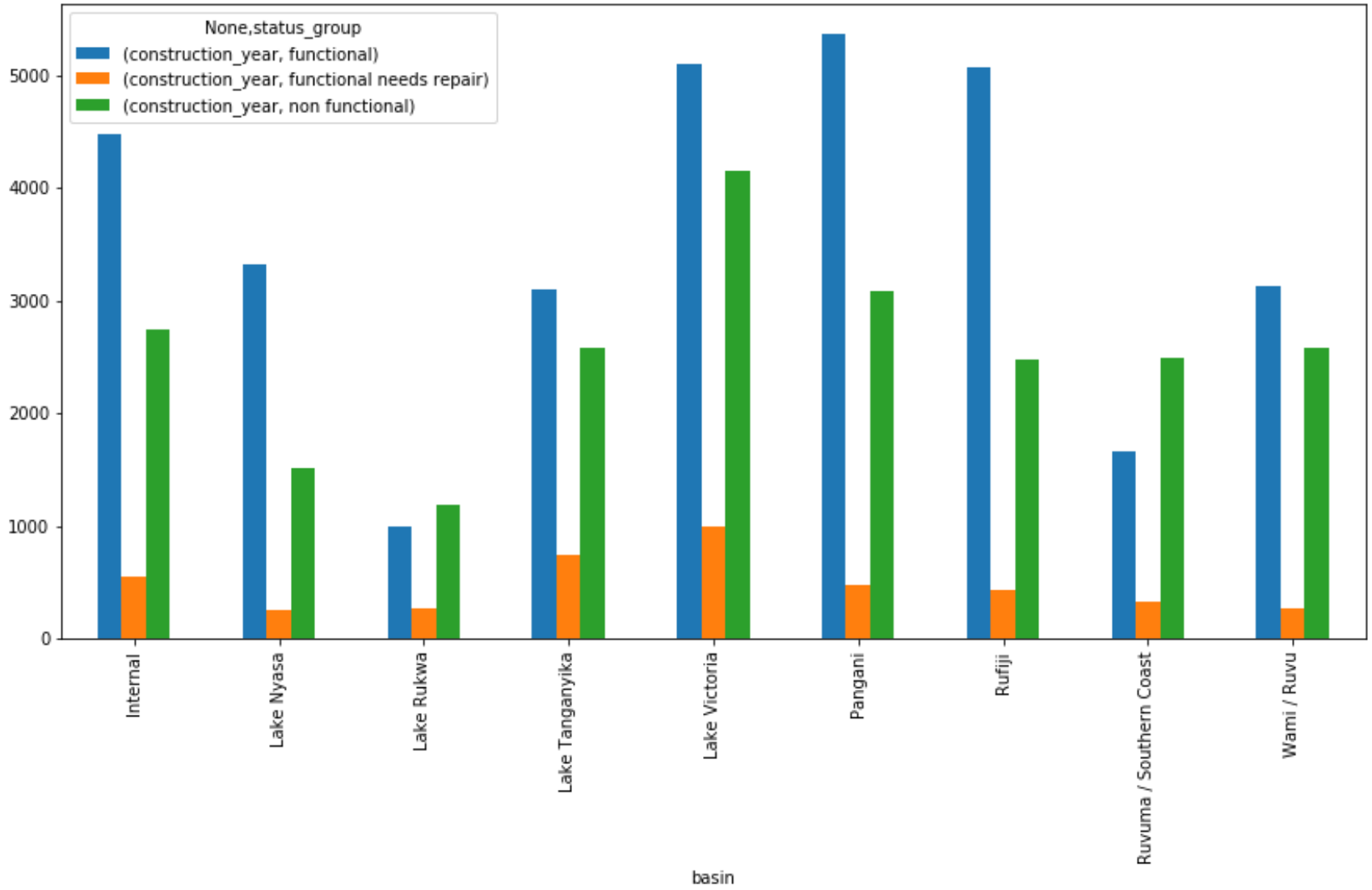

```
In [11]: train_labels.status_group.value_counts(normalize=True)

Out[11]: functional                 0.543081
         non functional             0.384242
         functional needs repair    0.072677
         Name: status_group, dtype: float64
```

DATASET

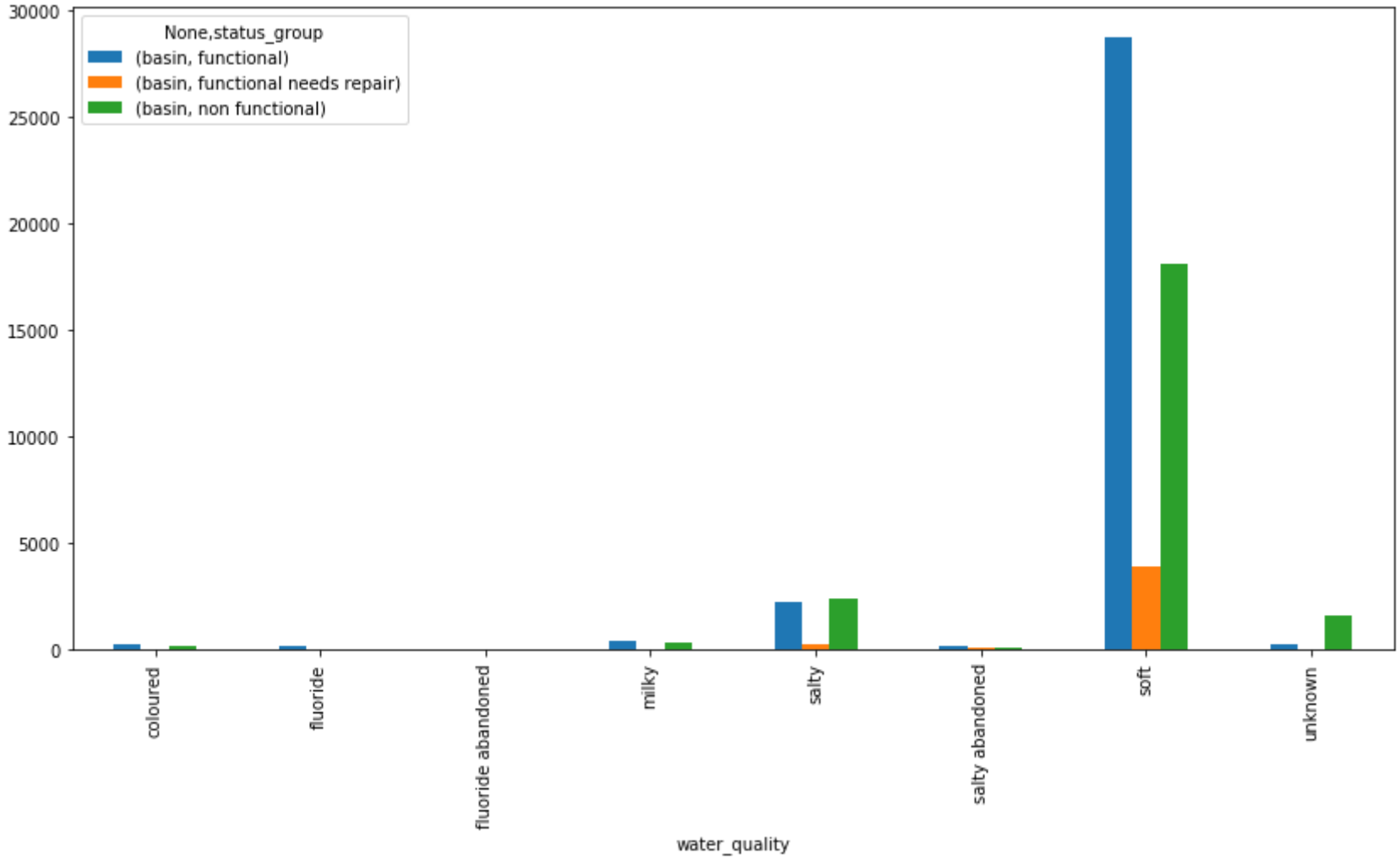
Water Plant Predictive Model

This plot provides a benchmark of sorts for purposes of evaluating each of the variables to be discussed below. The plot tells us that 54.3% of all pumps are functional, 38.4% are non-functional, and 7.3% are functional but in need of repair. We can use these metrics to partially assess each of the individual categorical variable values found within the data set; as we analyze each variable value, we can determine whether or not the percentage of pumps pertaining to that variable value either exceeds or falls short of the overall performance metrics plotted above. For example, those exceeding the 54.3% “functional” metric may share characteristics that poorer performing pumps may benefit from emulating / replicating.

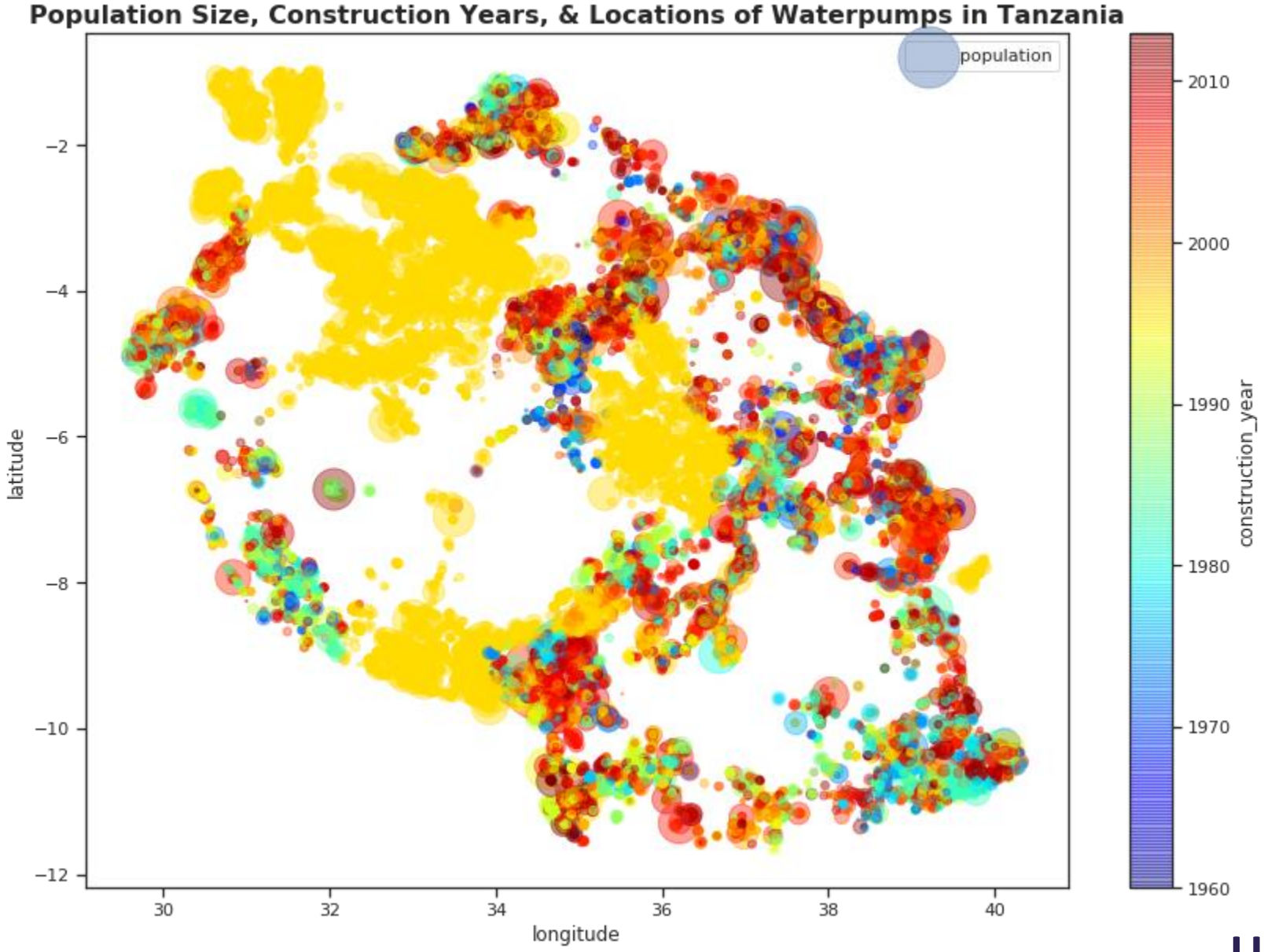
DATASET


None,status_group
(construction_year, functional)
(construction_year, functional needs repair)
(construction_year, non functional)
5000
4000
3000
2000
1000
0
Internal
Lake Nyasa
Lake Rukwa
Lake Tanganyika
Lake Victoria
Pangani
Rufiji
Ruvuma / Southern Coast
Wami / Ruvu
basin

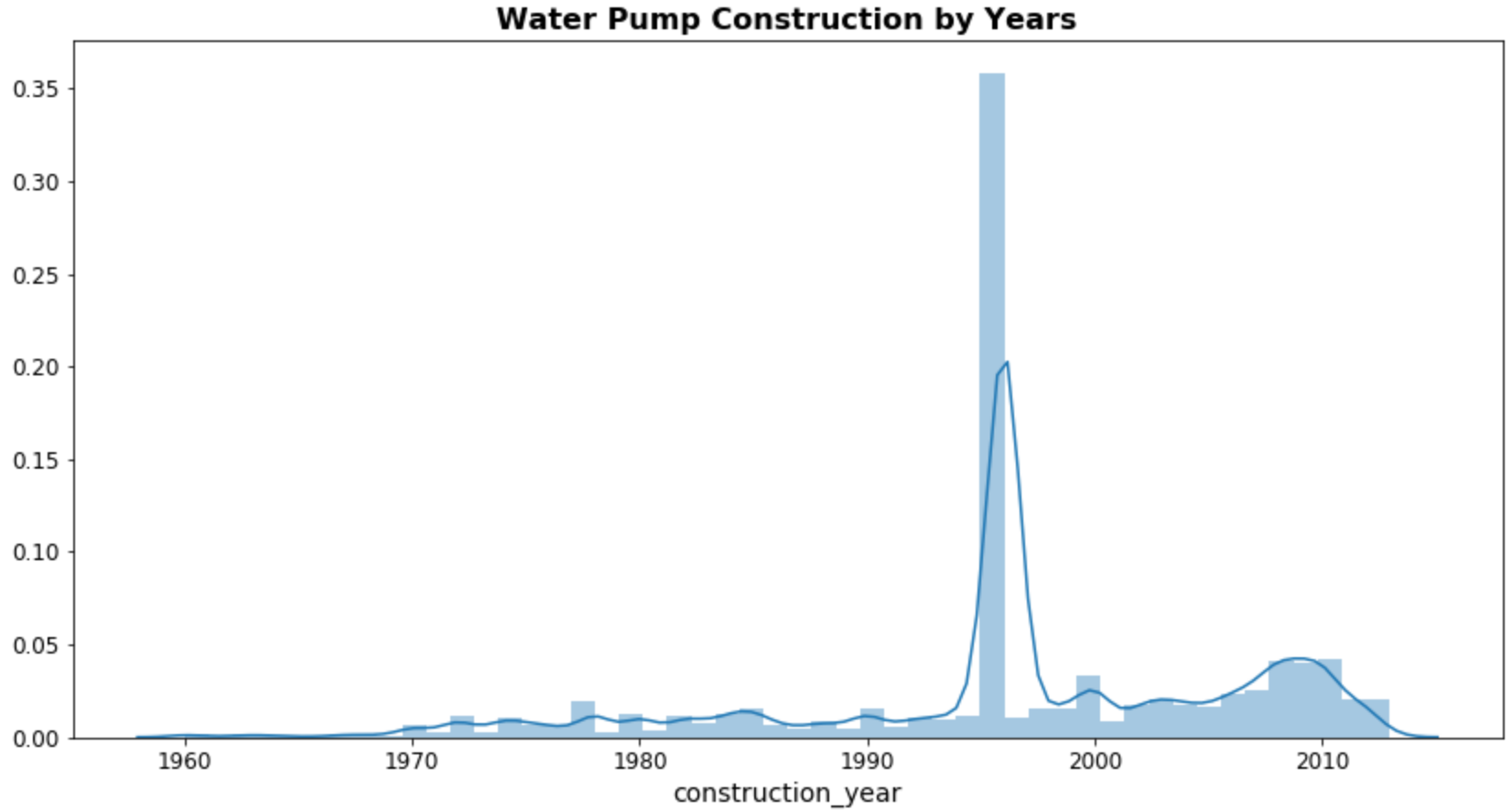
DATASET


None,status_group
(basin, functional)
(basin, functional needs repair)
(basin, non functional)
30000
25000
20000
15000
10000
5000
0
coloured
fluoride
fluoride abandoned
milky
salty
salty abandoned
soft
unknown
water_quality

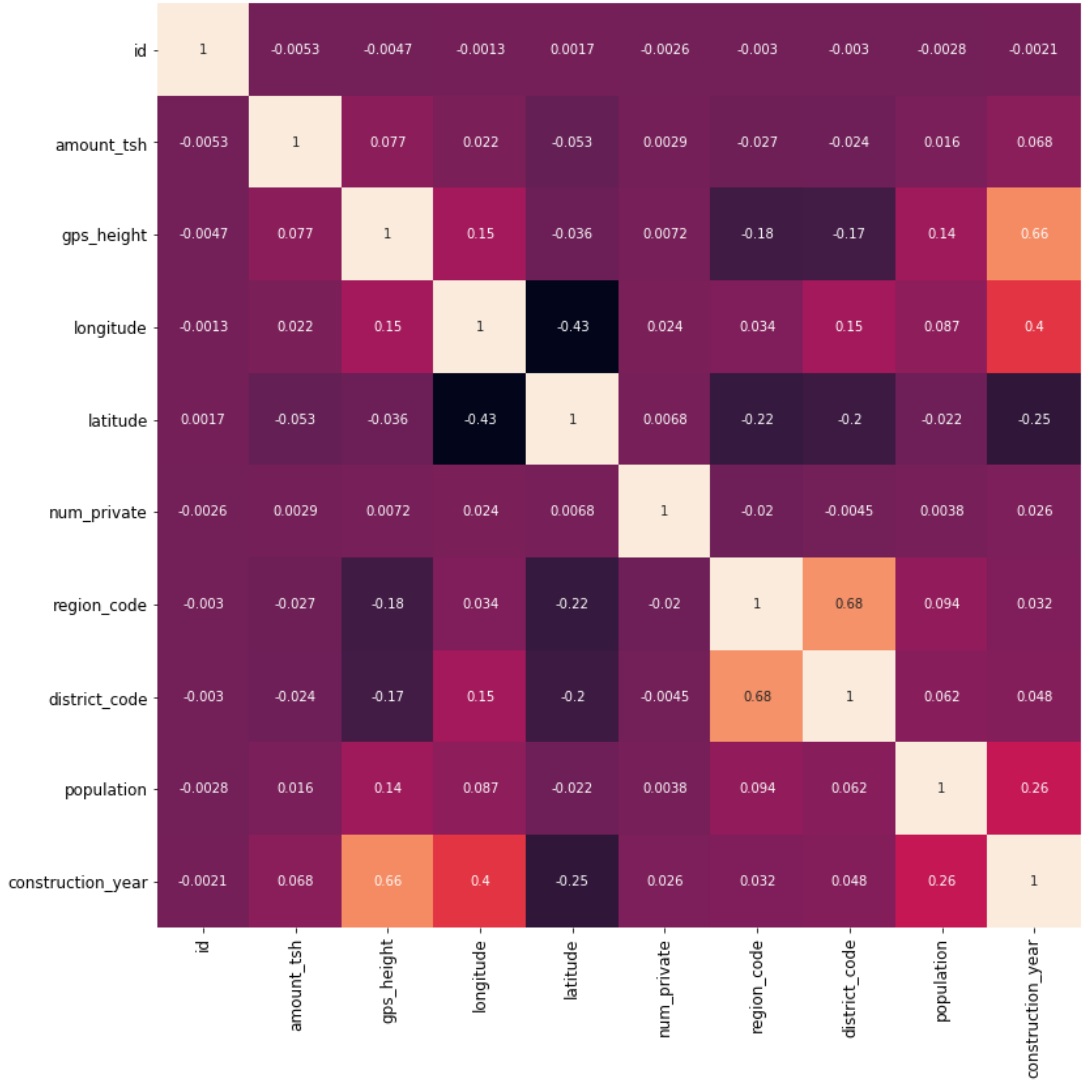
DATASET


Population Size, Construction Years, & Locations of Waterpumps in Tanzania
population
latitude
longitude
construction_year
−2
−4
−6
−8
−10
−12
30
32
34
36
38
40
2010
2000
1990
1980
1970
1960

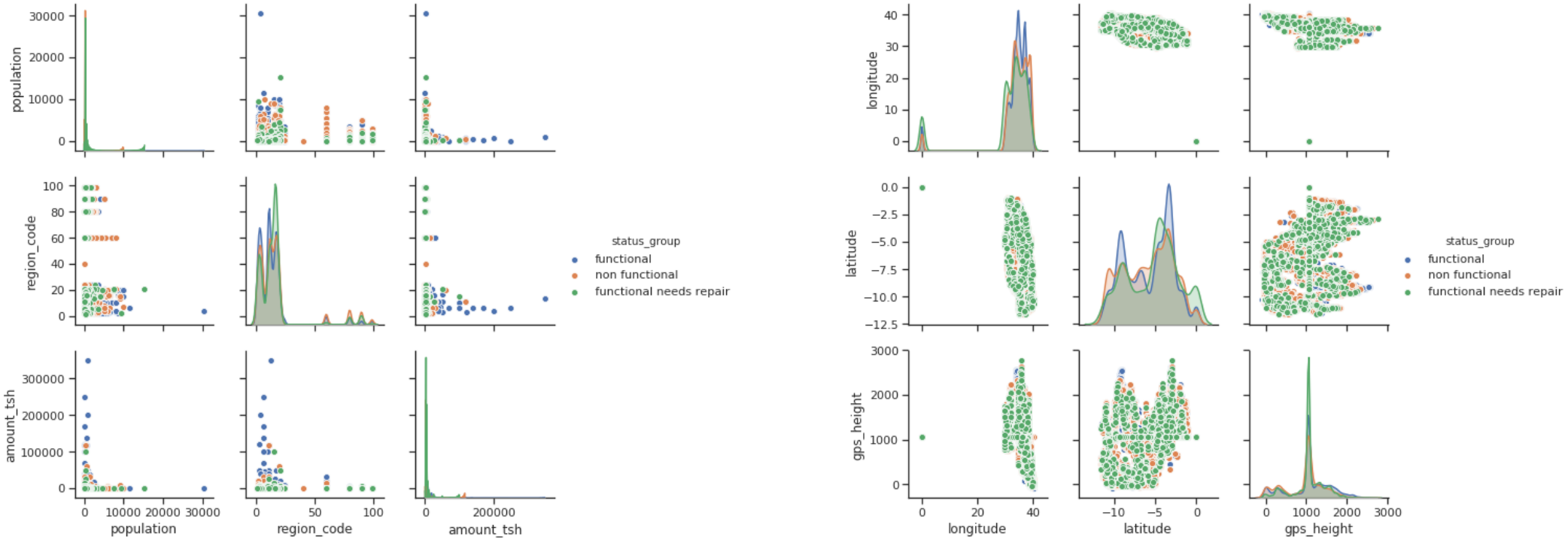
DATASET


Water Pump Construction by Years
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1960
1970
1980
1990
2000
2010
construction_year

DATASET

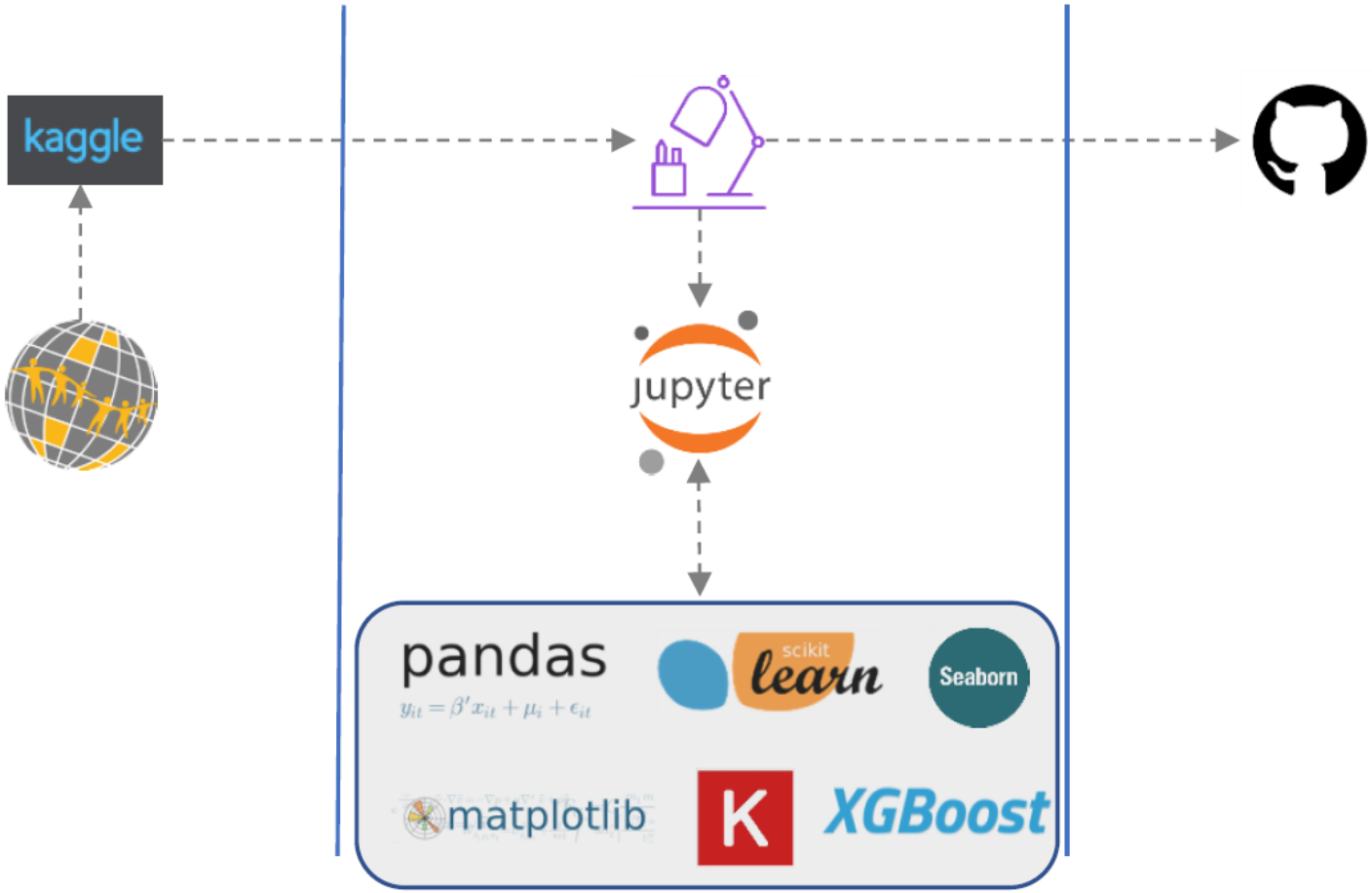

id 1 -0.0053 -0.0047 -0.0013 0.0017 -0.0026 -0.003 -0.003 -0.0028 -0.0021
amount_tsh -0.0053 1 0.077 0.022 -0.053 0.0029 -0.027 -0.024 0.016 0.068
gps_height -0.0047 0.077 1 0.15 -0.036 0.0072 -0.18 -0.17 0.14 0.66
longitude -0.0013 0.022 0.15 1 -0.43 0.024 0.034 0.15 0.087 0.4
latitude 0.0017 -0.053 -0.036 -0.43 1 0.0068 -0.22 -0.2 -0.022 -0.25
num_private -0.0026 0.0029 0.0072 0.024 0.0068 1 -0.02 -0.0045 0.0038 0.026
region_code -0.003 -0.027 -0.18 0.034 -0.22 -0.02 1 0.68 0.094 0.032
district_code -0.003 -0.024 -0.17 0.15 -0.2 -0.0045 0.68 1 0.062 0.048
population -0.0028 0.016 0.14 0.087 -0.022 0.0038 0.094 0.062 1 0.26
construction_year -0.0021 0.068 0.66 0.4 -0.25 0.026 0.032 0.048 0.26 1
id
amount_tsh
gps_height
longitude
latitude
num_private
region_code
district_code
population
construction_year

DATASET


population
region_code
amount_tsh
longitude
latitude
gps_height
status_group
functional
non functional
functional needs repair

USE CASE

Water Plant Predictive Model.



USE CASE

Water Plant Predictive Model.

The purpose of this small project is to predict, from actual data, which water pumps are operating correctly, which are in need of maintenance and which do not work.

It is a case of Data Science applied to predictive maintenance. Predictive maintenance periodically monitors the machines, based on the analysis of data collected through monitoring or field inspections.

The main objective of predictive maintenance is the timely verification of the equipment in order to anticipate eventual problems that may cause .

MODEL

Water Plant Predictive Model.

MODEL

Process

- ETL Extract_Transform_Load.
- Feature Creation.
- Model Definition.
- Model Training.
- Model Evaluation.
- Model Deployment and Data Product.
- Create Final Deliverables.

MODEL

E T L

- Merge test and train to clean data.
- Convert date recorded.
- Replace low frecuency features with NaN values to reduce categorical classes. With 250 as the minimum frequency allowed.
- For numerical values, we create a dummy column for each feature with NaN's for tracking proposal.
- Create a feature column for NaN's tracking
- Replace nonsense zeros with mean values.
- Remove non relevant columns

MODEL

Feature creation

- Hot Encoding the categorical features

Model Definition

- Decision Tree
- Random Forest
- Logistic Regresion
- XGBoost

MODEL

Feature creation

- Hot Encoding the categorical features

Model Definition

- Decision Tree
- Random Forest
- Logistic Regresion
- XGBoost

MODEL / DECISION TREE

Accuracy / Score

```
In [52]: accuracy_score(y_test, y_pred)
Out[52]: 0.7426936026936027
```

Confusion Matrix

```
In [53]: confusion_matrix(y_test, y_pred)
Out[53]: array([[6366,  549, 1183],
                [ 489,  384,  201],
                [1170,  229, 4279]])
```

Classification Report

```
In [55]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
functional	0.79	0.79	0.79	8098
functional needs repair	0.33	0.36	0.34	1074
non functional	0.76	0.75	0.75	5678
micro avg	0.74	0.74	0.74	14850
macro avg	0.63	0.63	0.63	14850
weighted avg	0.75	0.74	0.74	14850

MODEL / DECISION TREE (STANDARDSCALER)

Accuracy / Score

```
In [54]: accuracy_score(y_test, y_pred)
Out[54]: 0.7416835016835017
```

Confusion Matrix

```
In [55]: confusion_matrix(y_test, y_pred)
Out[55]: array([[6352,  549, 1197],
                [ 488,  379,  207],
                [1174,  221, 4283]])
```

Classification Report

```
In [56]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
functional	0.79	0.78	0.79	8098
functional needs repair	0.33	0.35	0.34	1074
non functional	0.75	0.75	0.75	5678
micro avg	0.74	0.74	0.74	14850
macro avg	0.63	0.63	0.63	14850
weighted avg	0.74	0.74	0.74	14850

MODEL / RANDOM FOREST

Accuracy / Score

```
In [58]: accuracy_score(y_test, y_pred)
Out[58]: 0.808956228956229
```

Confusion Matrix

```
In [59]: confusion_matrix(y_test, y_pred)
Out[59]: array([[7339,  145,  614],
                [ 600,  325,  149],
                [1254,   75, 4349]])
```

Classification Report

```
In [55]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
functional	0.79	0.79	0.79	8098
functional needs repair	0.33	0.36	0.34	1074
non functional	0.76	0.75	0.75	5678
micro avg	0.74	0.74	0.74	14850
macro avg	0.63	0.63	0.63	14850
weighted avg	0.75	0.74	0.74	14850

MODEL / RANDOM FOREST (STANDARDSCALER)

Accuracy / Score

```
In [63]: accuracy_score(y_test, y_pred)
Out[63]: 0.8094276094276094
```

Confusion Matrix

```
In [64]: confusion_matrix(y_test, y_pred)
Out[64]: array([[7342,  141,  615],
                [ 597,  327,  150],
                [1251,   76, 4351]])
```

Classification Report

```
In [65]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
functional	0.80	0.91	0.85	8098
functional needs repair	0.60	0.30	0.40	1074
non functional	0.85	0.77	0.81	5678
micro avg	0.81	0.81	0.81	14850
macro avg	0.75	0.66	0.69	14850
weighted avg	0.80	0.81	0.80	14850

MODEL / LOGISTIC REGRESSION

Accuracy / Score

```
In [67]: accuracy_score(y_test, y_pred)
Out[67]: 0.7077441077441078
```

Confusion Matrix

```
In [68]: confusion_matrix(y_test, y_pred)
Out[68]: array([[7435,    0,  663],
                [ 911,    0,  163],
                [2603,    0, 3075]])
```

Classification Report

```
In [69]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
functional	0.68	0.92	0.78	8098
functional needs repair	0.00	0.00	0.00	1074
non functional	0.79	0.54	0.64	5678
micro avg	0.71	0.71	0.71	14850
macro avg	0.49	0.49	0.47	14850
weighted avg	0.67	0.71	0.67	14850

MODEL / LOGISTIC REGRESSION(STANDARDSCALER)

Accuracy / Score

```
In [72]: accuracy_score(y_test, y_pred)
Out[72]: 0.7424242424242424
```

Confusion Matrix

```
In [73]: confusion_matrix(y_test, y_pred)
Out[73]: array([[7258,   50,  790],
                [ 766,   87,  221],
                [1976,   22, 3680]])
```

Classification Report

```
In [74]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
functional	0.73	0.90	0.80	8098
functional needs repair	0.55	0.08	0.14	1074
non functional	0.78	0.65	0.71	5678
micro avg	0.74	0.74	0.74	14850
macro avg	0.69	0.54	0.55	14850
weighted avg	0.74	0.74	0.72	14850

MODEL / LR-NEWTON_CG

Accuracy / Score

```
In [78]: accuracy_score(y_test, y_pred)
Out[78]: 0.7423569023569023
```

Confusion Matrix

```
In [79]: confusion_matrix(y_test, y_pred)
Out[79]: array([[7241,   74,  783],
                [ 755,  119,  200],
                [1979,   35, 3664]])
```

Classification Report

```
In [80]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
functional	0.73	0.89	0.80	8098
functional needs repair	0.52	0.11	0.18	1074
non functional	0.79	0.65	0.71	5678
micro avg	0.74	0.74	0.74	14850
macro avg	0.68	0.55	0.56	14850
weighted avg	0.74	0.74	0.72	14850

MODEL / LR-NEWTON_CG(STANDARDSCALER)

Accuracy / Score

```
In [85]: accuracy_score(y_test, y_pred)
Out[85]: 0.7445117845117845
```

Confusion Matrix

```
In [86]: confusion_matrix(y_test, y_pred)
Out[86]: array([[7242,   80,  776],
                [ 742,  125,  207],
                [1949,   40, 3689]])
```

Classification Report

```
In [87]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
functional	0.73	0.89	0.80	8098
functional needs repair	0.51	0.12	0.19	1074
non functional	0.79	0.65	0.71	5678
micro avg	0.74	0.74	0.74	14850
macro avg	0.68	0.55	0.57	14850
weighted avg	0.74	0.74	0.72	14850

MODEL / XGBOOST

Accuracy / Score

```
In [90]: accuracy_score(y_test, y_pred)
Out[90]: 0.7652525252525253
```

Confusion Matrix

```
In [91]: confusion_matrix(y_test, y_pred)
Out[91]: array([[7530,   58,  510],
                [ 747,  174,  153],
                [1978,   40, 3660]])
```

Classification Report

```
In [92]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
functional	0.73	0.93	0.82	8098
functional needs repair	0.64	0.16	0.26	1074
non functional	0.85	0.64	0.73	5678
micro avg	0.77	0.77	0.77	14850
macro avg	0.74	0.58	0.60	14850
weighted avg	0.77	0.77	0.75	14850

MODEL / XGBOOST(STANDARDSCALER)

Accuracy / Score

```
In [94]: accuracy_score(y_test, y_pred)
Out[94]: 0.7445117845117845
```

Confusion Matrix

```
In [95]: confusion_matrix(y_test, y_pred)
Out[95]: array([[7242,   80,  776],
                [ 742,  125,  207],
                [1949,   40, 3689]])
```

Classification Report

```
In [96]: print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
functional	0.73	0.89	0.80	8098
functional needs repair	0.51	0.12	0.19	1074
non functional	0.79	0.65	0.71	5678
micro avg	0.74	0.74	0.74	14850
macro avg	0.68	0.55	0.57	14850
weighted avg	0.74	0.74	0.72	14850

MODEL / SUMMARY

MODEL	SCALED	ACCURACY	PRECISSION	RECALL	F1-SCORE
Decision Tree	No	0.7416	0.74	0.74	0.74
Decision Tree	Yes	0.7417	0.74	0.74	0.74
Random Forest	No	0.8090	0.80	0.81	0.80
Random Forest	Yes	0.8094	0.80	0.81	0.80
Logistic Regresion	No	0.7077	0.67	0.71	0.67
Logistic Regresion	Yes	0.7424	0.74	0.74	0.72
LR – Newton-CG	No	0.7424	0.74	0.74	0.72
LR – Newton-CG	Yes	0.7445	0.74	0.74	0.72
XGBoost	No	0.7625	0.77	0.77	0.75
XGBoost	Yes	0.7445	0.74	0.74	0.72

MODEL

Random Fores

- Cross Validation
  - CV = 5
- Cross Validation (accuracy)
  - ✓ 0.8166821
  - ✓ 0.81222119
  - ✓ 0.81447811
  - ✓ 0.80984848
  - ✓ 0.80998485
- Similar values -> OK

CONCLUSION

Water Plant Predictive Model.

CONCLUSION

EL SUBTÍTULO SE ESCRIBE AQUÍ

The Random Forest model scored highest within the context of the **DrivenData.org** data challenge, achieving an overall accuracy of .8195 (just .009 less than the top score of .8285) and a ranking within the top seven percent of all submissions. Therefore, the **Random Forest** model is recommended for use by Tanzania's Ministry of Water if overall accuracy across each of the three possible pump statuses is of most importance, while the **Bootstratp Aggregation** model should be preferred if identifying the largest number of pumps that are functional but in need of repair is the top priority.


THANK YOU