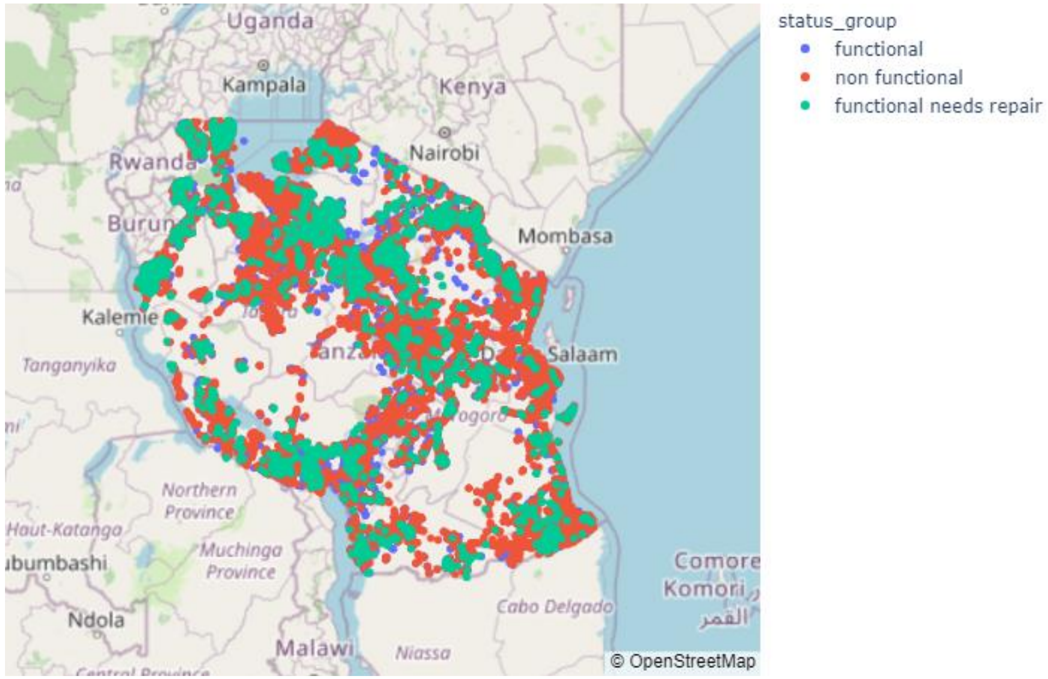
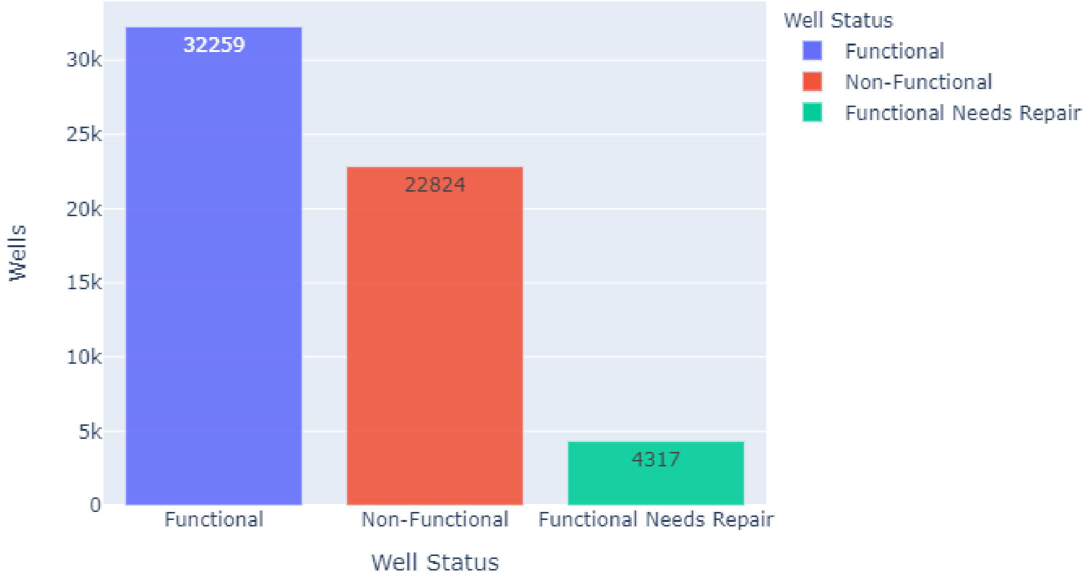
Classifying Functional and non-Functional wells

Rafael Villanueva

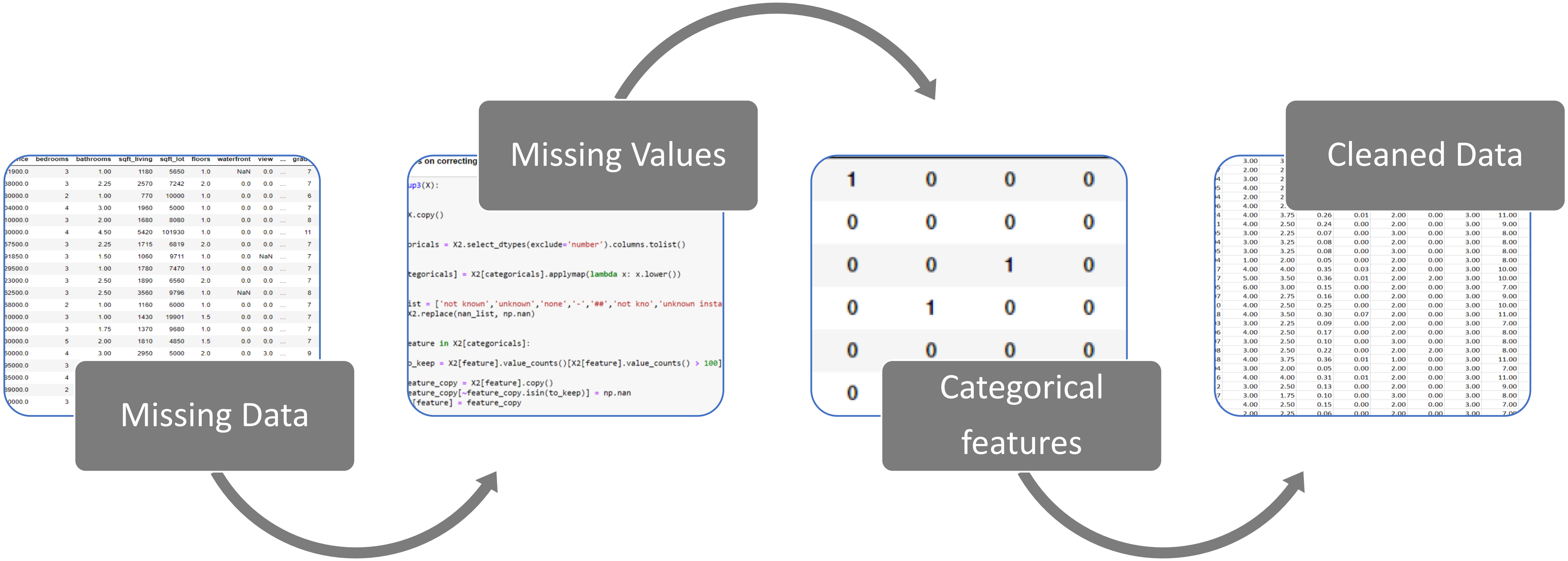
Problem Statement

- Tanzanian’s access to water is tied to accessibility to a working pump
- The Tanzanian Ministry of Water conducted a survey on several aspects of water pumps resulting in one outstanding question: are they still functional?





Data Cleaning



Missing Data

rice	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grad
21900.0	3	1.00	1180	5650	1.0	NaN	0.0	...	7
38000.0	3	2.25	2570	7242	2.0	0.0	0.0	...	7
30000.0	2	1.00	770	10000	1.0	0.0	0.0	...	6
04000.0	4	3.00	1960	5000	1.0	0.0	0.0	...	7
10000.0	3	2.00	1680	8080	1.0	0.0	0.0	...	8
30000.0	4	4.50	5420	101930	1.0	0.0	0.0	...	11
57500.0	3	2.25	1715	6819	2.0	0.0	0.0	...	7
91850.0	3	1.50	1060	9711	1.0	0.0	NaN	...	7
29500.0	3	1.00	1780	7470	1.0	0.0	0.0	...	7
23000.0	3	2.50	1890	6560	2.0	0.0	0.0	...	7
62500.0	3	2.50	3560	9796	1.0	NaN	0.0	...	8
68000.0	2	1.00	1160	6000	1.0	0.0	0.0	...	7
10000.0	3	1.00	1430	19901	1.5	0.0	0.0	...	7
00000.0	3	1.75	1370	9680	1.0	0.0	0.0	...	7
30000.0	5	2.00	1810	4850	1.5	0.0	0.0	...	7
50000.0	4	3.00	2950	5000	2.0	0.0	3.0	...	9
95000.0	3								
35000.0	4								
39000.0	2								
0000.0	3								

Missing Values

s on correcting

```
up3(X):

X.copy()

oricals = X2.select_dtypes(exclude='number').columns.tolist()

tegoricals] = X2[categoricals].applymap(lambda x: x.lower())

ist = ['not known','unknown','none','-','##','not kno','unknown insta
X2.replace(nan_list, np.nan)

eature in X2[categoricals]:

o_keep = X2[feature].value_counts()[X2[feature].value_counts() > 100]

eature_copy = X2[feature].copy()
eature_copy[~feature_copy.isin(to_keep)] = np.nan
[feature] = feature_copy
```

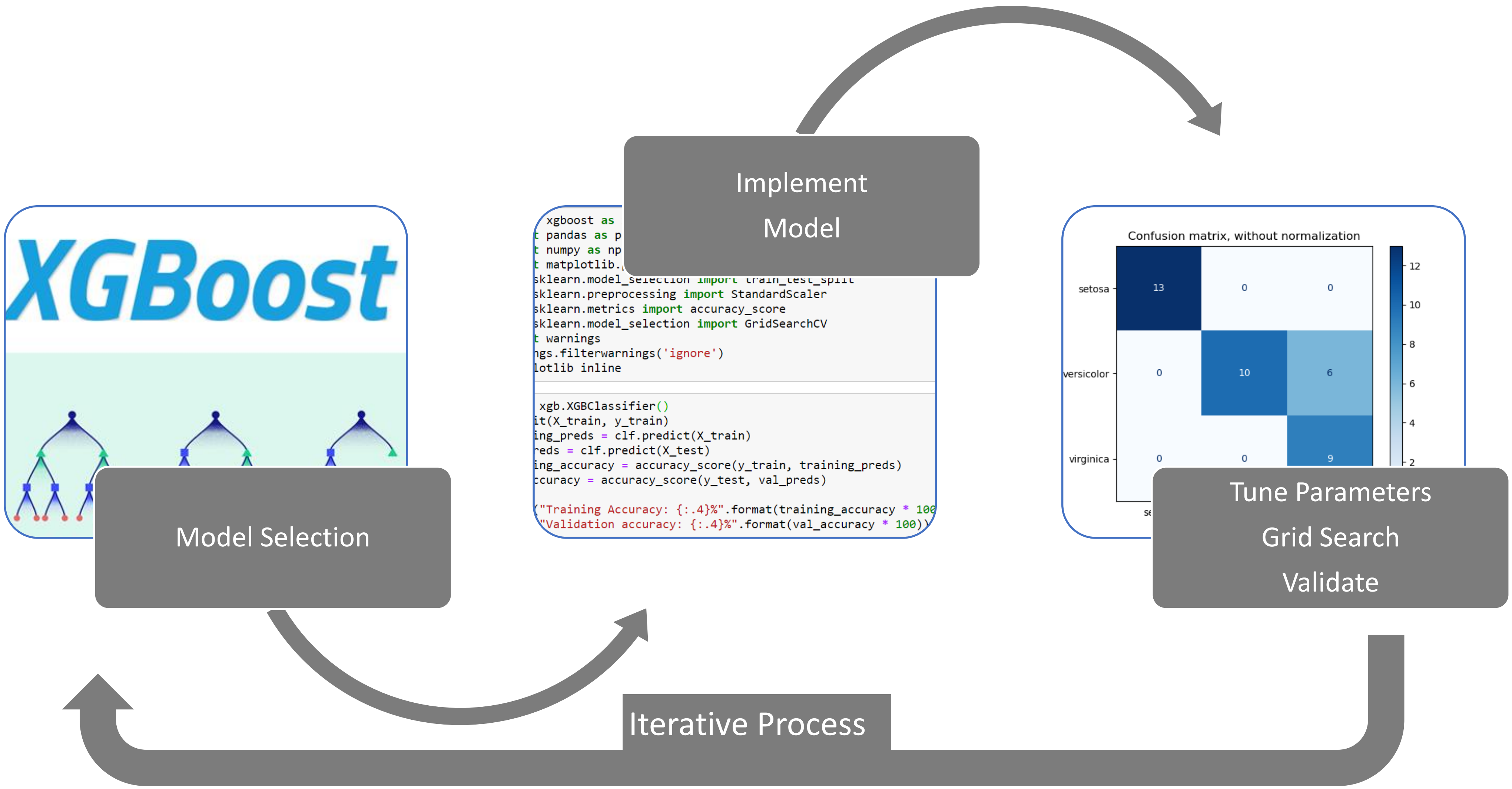
Categorical features

1	0	0	0
0	0	0	0
0	0	1	0
0	1	0	0
0	0	0	0
0			

Cleaned Data

	3.00	3						
7	2.00	2						
4	3.00	2						
5	4.00	2						
4	2.00	2						
6	4.00	2.						
4	4.00	3.75	0.26	0.01	2.00	0.00	3.00	11.00
1	4.00	2.50	0.24	0.00	2.00	0.00	3.00	9.00
5	3.00	2.25	0.07	0.00	3.00	0.00	3.00	8.00
4	3.00	3.25	0.08	0.00	2.00	0.00	3.00	8.00
5	3.00	3.25	0.08	0.00	3.00	0.00	3.00	8.00
4	1.00	2.00	0.05	0.00	2.00	0.00	3.00	8.00
7	4.00	4.00	0.35	0.03	2.00	0.00	3.00	10.00
7	5.00	3.50	0.36	0.01	2.00	2.00	3.00	10.00
5	6.00	3.00	0.15	0.00	2.00	0.00	3.00	7.00
7	4.00	2.75	0.16	0.00	2.00	0.00	3.00	9.00
0	4.00	2.50	0.25	0.00	2.00	0.00	3.00	10.00
8	4.00	3.50	0.30	0.07	2.00	0.00	3.00	11.00
3	3.00	2.25	0.09	0.00	2.00	0.00	3.00	7.00
6	4.00	2.50	0.17	0.00	2.00	0.00	3.00	8.00
7	3.00	2.50	0.10	0.00	3.00	0.00	3.00	8.00
8	3.00	2.50	0.22	0.00	2.00	2.00	3.00	8.00
8	4.00	3.75	0.36	0.01	1.00	0.00	3.00	11.00
4	3.00	2.00	0.05	0.00	2.00	0.00	3.00	7.00
6	4.00	4.00	0.31	0.01	2.00	0.00	3.00	11.00
2	3.00	2.50	0.13	0.00	2.00	0.00	3.00	9.00
7	3.00	1.75	0.10	0.00	3.00	0.00	3.00	8.00
	4.00	2.50	0.15	0.00	2.00	0.00	3.00	7.00
	2.00	2.25	0.06	0.00	2.00	0.00	3.00	7.00

Modeling



Model Selection

Implement Model

Tune Parameters

Grid Search

Validate

Iterative Process

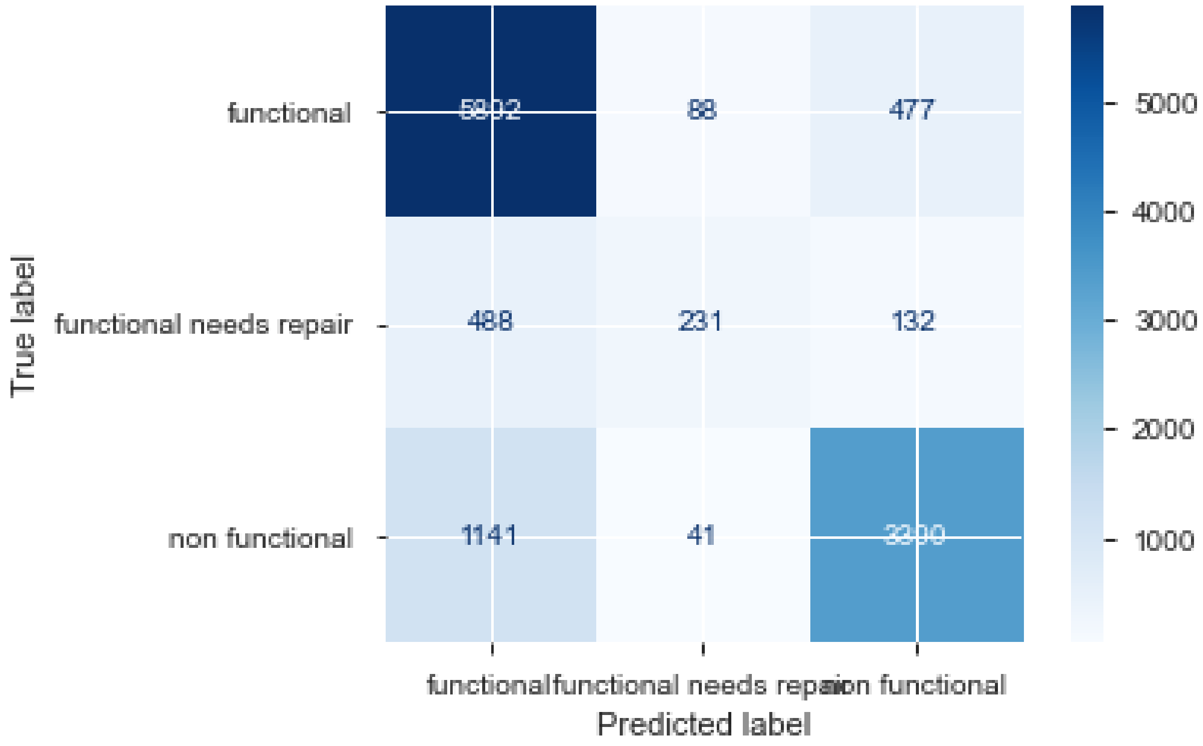
```
 xgboost as
t pandas as p
t numpy as np
t matplotlib.
sklearn.model_selection import train_test_split
sklearn.preprocessing import StandardScaler
sklearn.metrics import accuracy_score
sklearn.model_selection import GridSearchCV
t warnings
ngs.filterwarnings('ignore')
lotlib inline
```

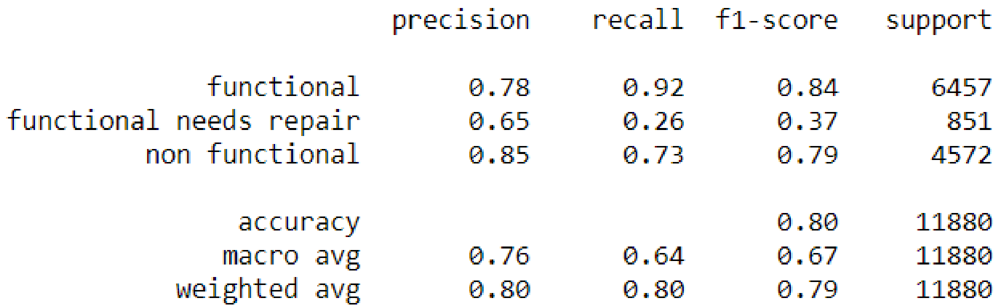
```
 xgb.XGBClassifier()
it(X_train, y_train)
ing_preds = clf.predict(X_train)
reds = clf.predict(X_test)
ing_accuracy = accuracy_score(y_train, training_preds)
ccuracy = accuracy_score(y_test, val_preds)

("Training Accuracy: {:.4}%".format(training_accuracy * 100
"Validation accuracy: {:.4}%".format(val_accuracy * 100))
```

Findings
Ensemble Classifier’s Excel

Method	Train Accuracy Baseline	Tuned Model Train Accuracy	Tuned Model Test Accuracy
Decision Tree	**72%**	**74.20%**	**76.86%**
Random Forest	**78.92%**	**79.91%**	**80%**
XGBoost Classifier	**84.69%**	**80%**	**80%**

Model Selection
XGBoost: With Reservations





```
                         precision    recall  f1-score   support

             functional       0.78      0.92      0.84      6457
functional needs repair       0.65      0.26      0.37       851
         non functional       0.85      0.73      0.79      4572

               accuracy                           0.80     11880
              macro avg       0.76      0.64      0.67     11880
           weighted avg       0.80      0.80      0.79     11880
```

Recommendations

- Complex models may not always be superior
- Predicting water pump functionality is possible!

Future Work

- Focus on preprocessing data
    - In depth research on how the model operates on types of data, missing data etc.

Thank You!

Eli Thomas

Flatiron Cohort 3.02.2020