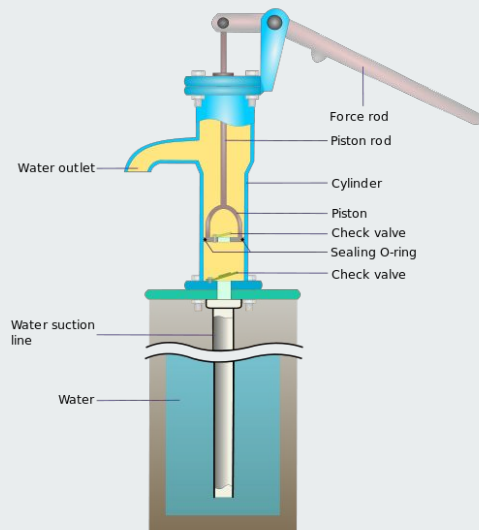



Pump it Up: Data Mining the Water Table

Simona Nitti -- Oleg Sautenkov -- Ayush Gupta



- 
- Describe the problem you want to solve using data science techniques. Why is it important? How your ML model can help (e.g., reduce costs, increase quality, etc.)? (6 points)

Access to water in Tanzania

Water supply and sanitation in Tanzania is characterised by:

- Decreasing access to at least basic water sources in the 2000s (especially in urban areas),
- Steady access to some form of sanitation (around 93% since the 1990s),
- **Intermittent water supply** and generally low quality of service.
- Many utilities are barely able to cover their operation and maintenance costs through revenues due to low tariffs and **poor efficiency**.

Access to Water and Sanitation in Tanzania (2007)^[1]

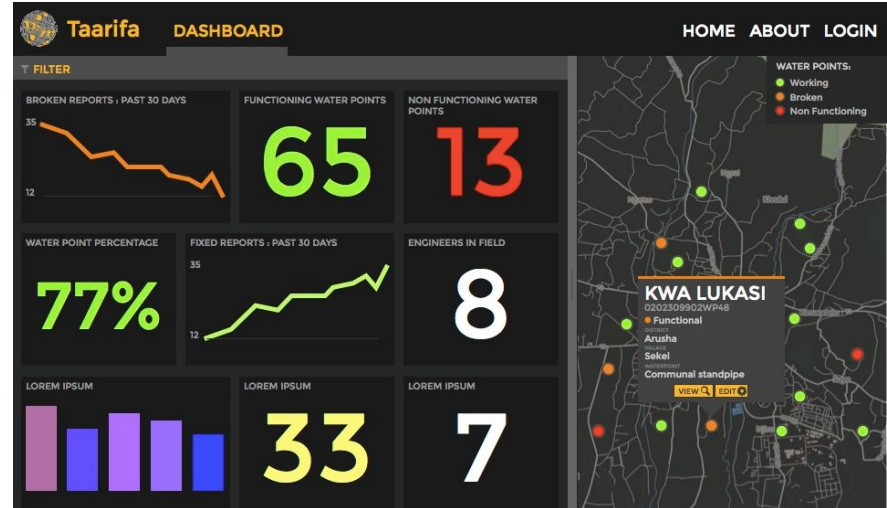
		Dar es Salaam (6% of population) ^[15]	Other urban areas (19% of population)	Rural (75% of population) ^[11]	Total
Water	Broad definition	85%	76%	40%	52%
	House connections ^[16]	8%	13%	1%	4%
Sanitation	All forms, including latrines	97%	97%	90%	93%
	Flush Toilets	10%	6%	1%	3%

***Broad definition:** It is defined as a type of water source that, by nature of its construction or through active intervention, is likely to be protected from outside contamination, in particular from contamination with fecal matter.

DataDriven competition

Predict which pumps are functional, which need some repairs, and which don't work at all.

In **Tanzania**, a large number of water pumps are out of service or do not operate at all; others need to be repaired. **Tanzania's Ministry of Water Resources** reached an agreement with Taarifa, an online data storage service, and the competition was launched on **DataDriven**.



DataDriven competition

Predict which pumps are functional, which need some repairs, and which don't work at all.



ML and Data Mining can help

to get a smart understanding of which waterpoints will fail. Then it can improve:

- maintenance operations
- increase access to at least basic water sources
- ensure continuity in water supply
- increase access to sanitation



Simona

- Load the data using pandas and split the data frame into X (inputs) and y (outputs). (2 points)

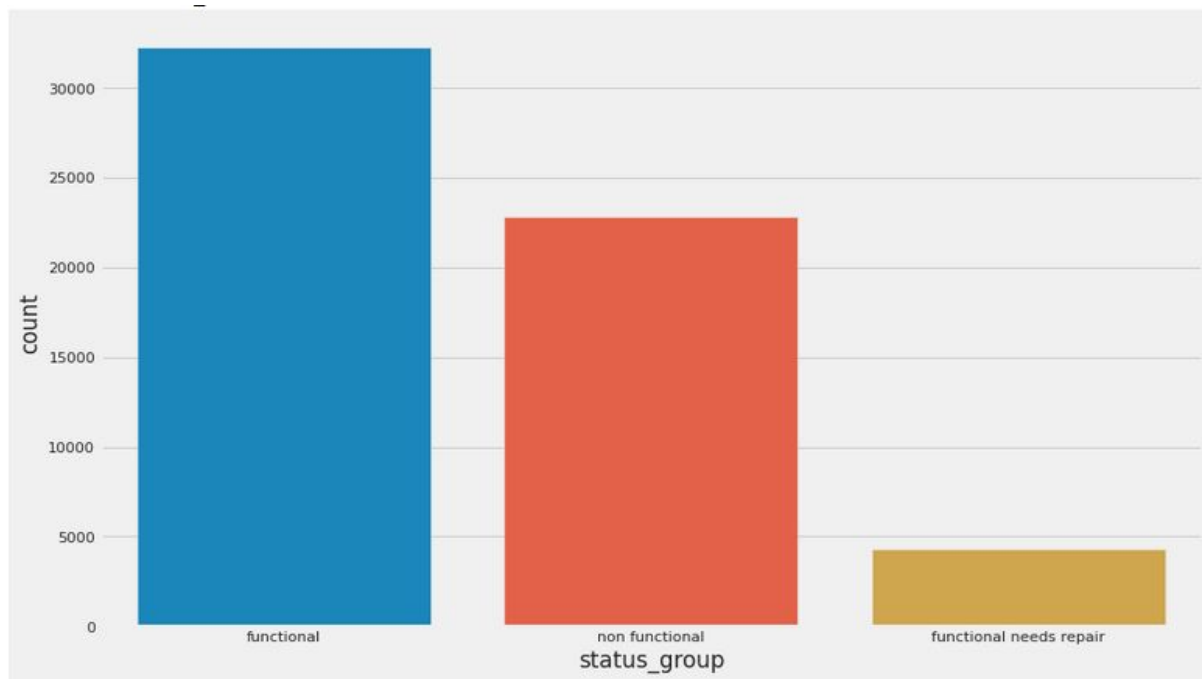
Load the data

The dataset has 59400 rows and 39 columns, without the label that comes in a different file.

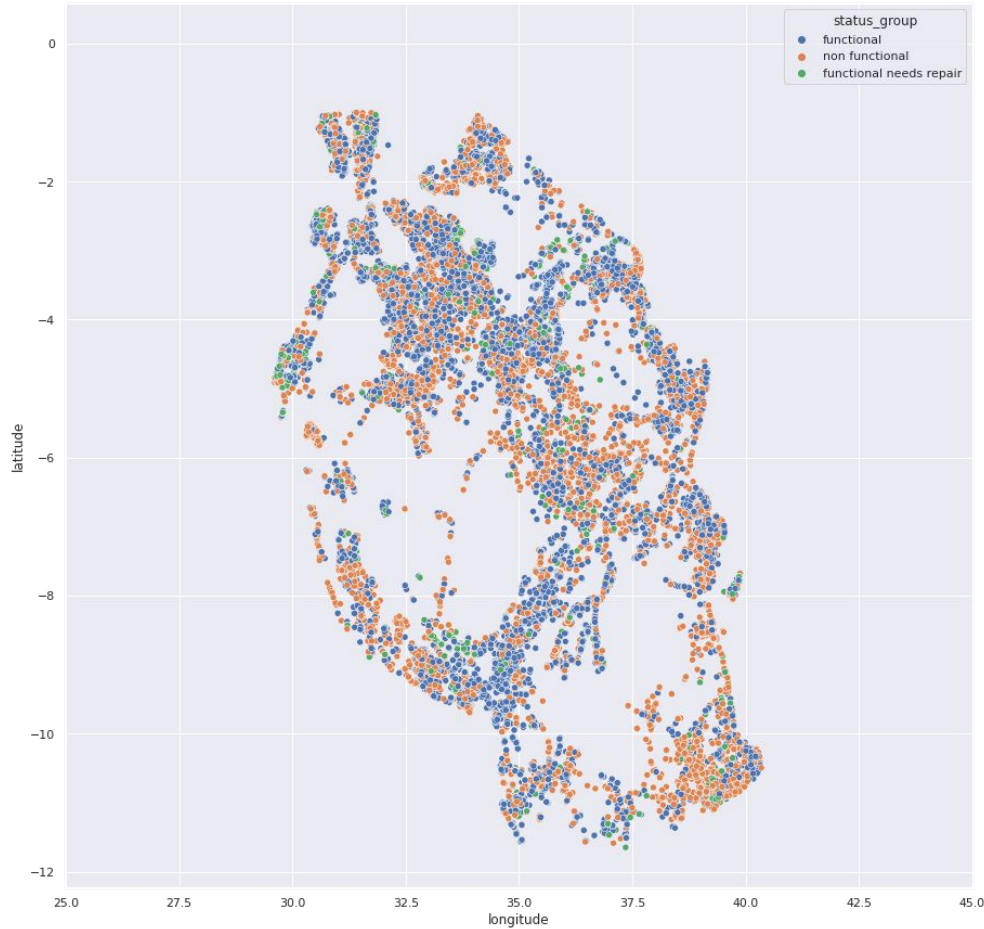
Group	Features	Description	Group	Features	Description
Characteristic associated with wells	gps_height	Altitude of the well	Info on geographical location	longitude	GPS coordinate
	wpt_name	Name of the waterpoint if there is one		latitude	GPS coordinate
	amount_tsh	Total static head (amount water available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

Load the data

The **target** column is characterized by 3 unique status of the pumps, with highly **imbalanced** proportion.

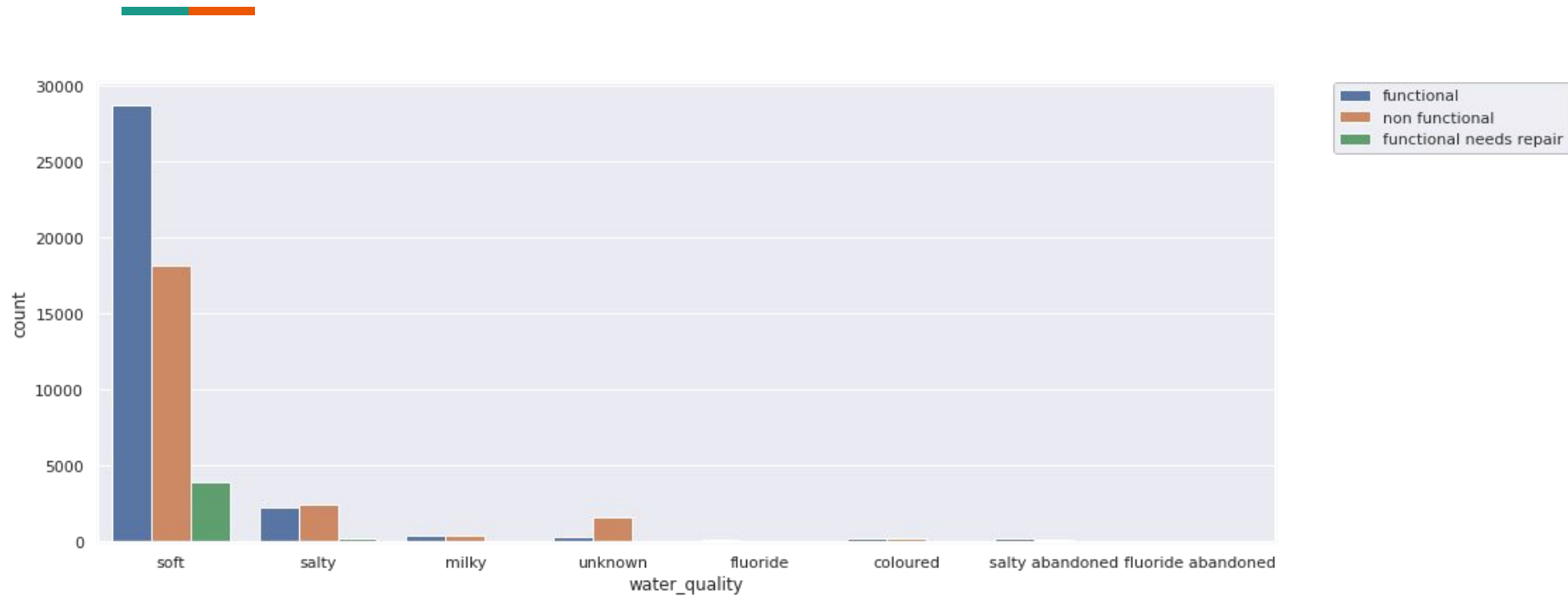


Location of Water pumps over Tanzania

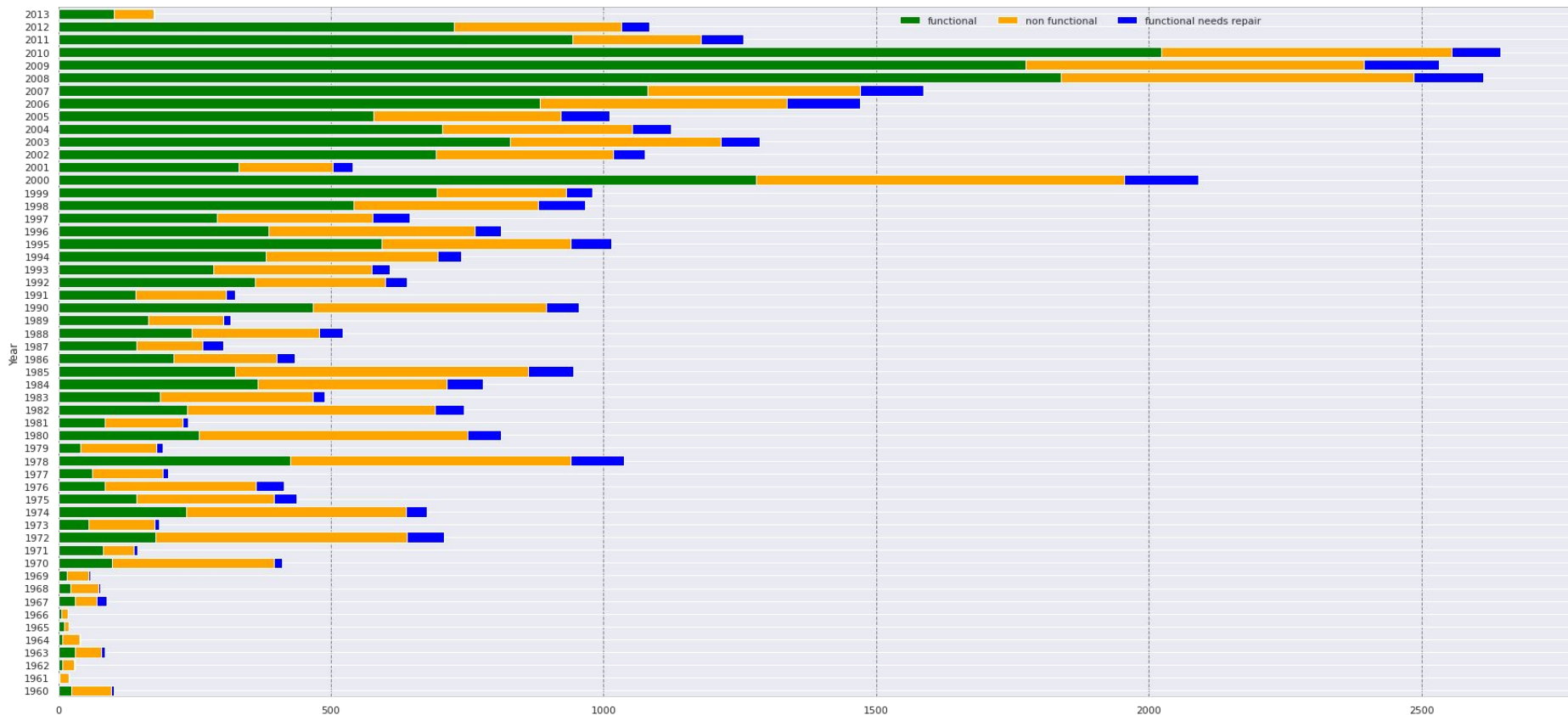


The locations of water pumps in different places in the Tanzania based on the status of their working, not working and needs to be repaired.

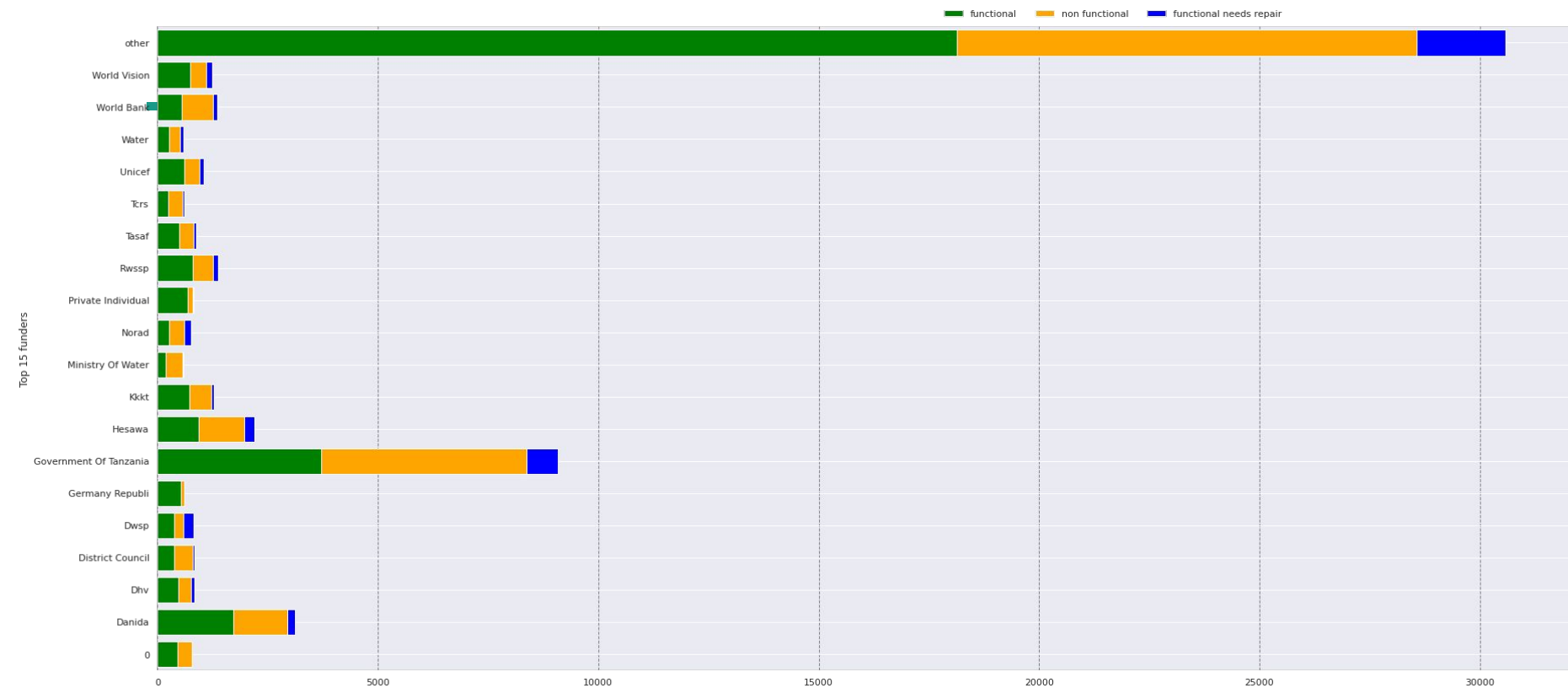
Water quality over different status groups



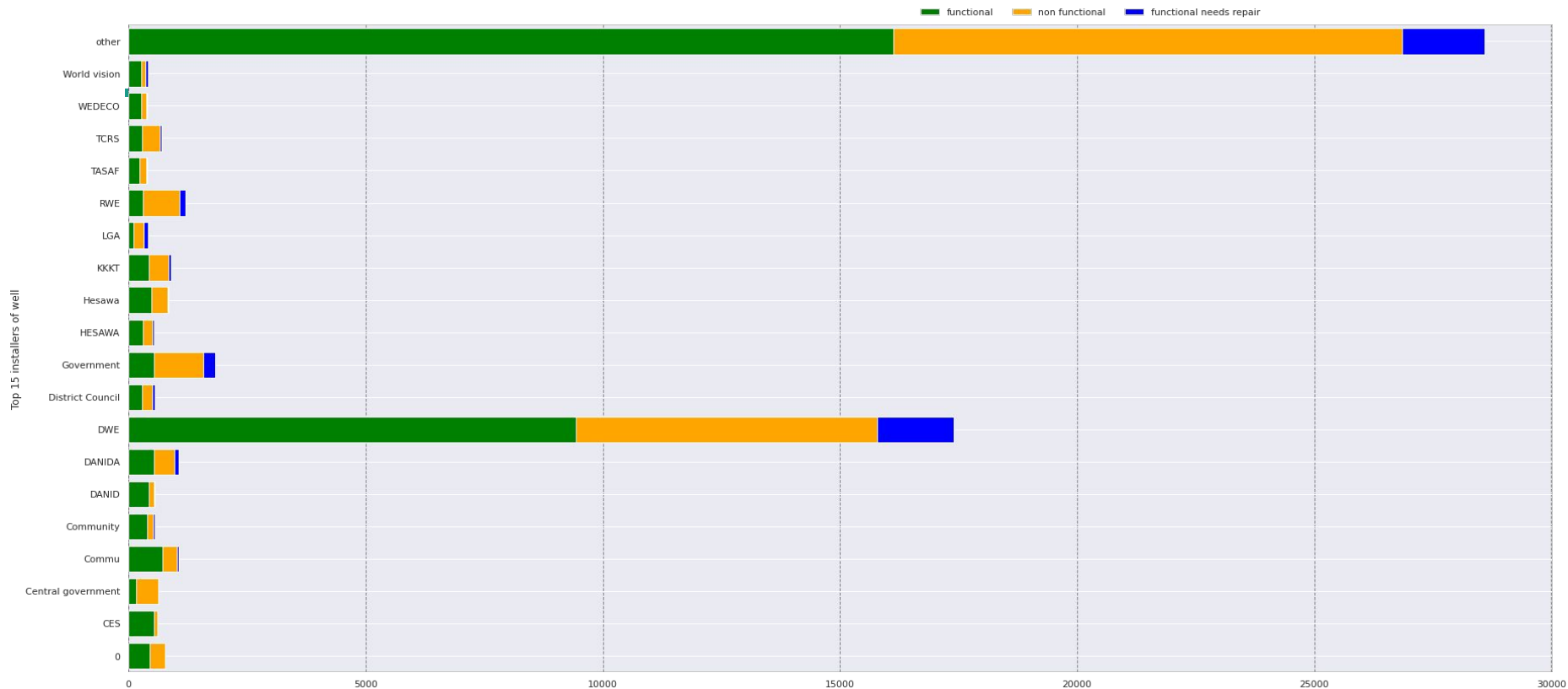
Status group over different construction years



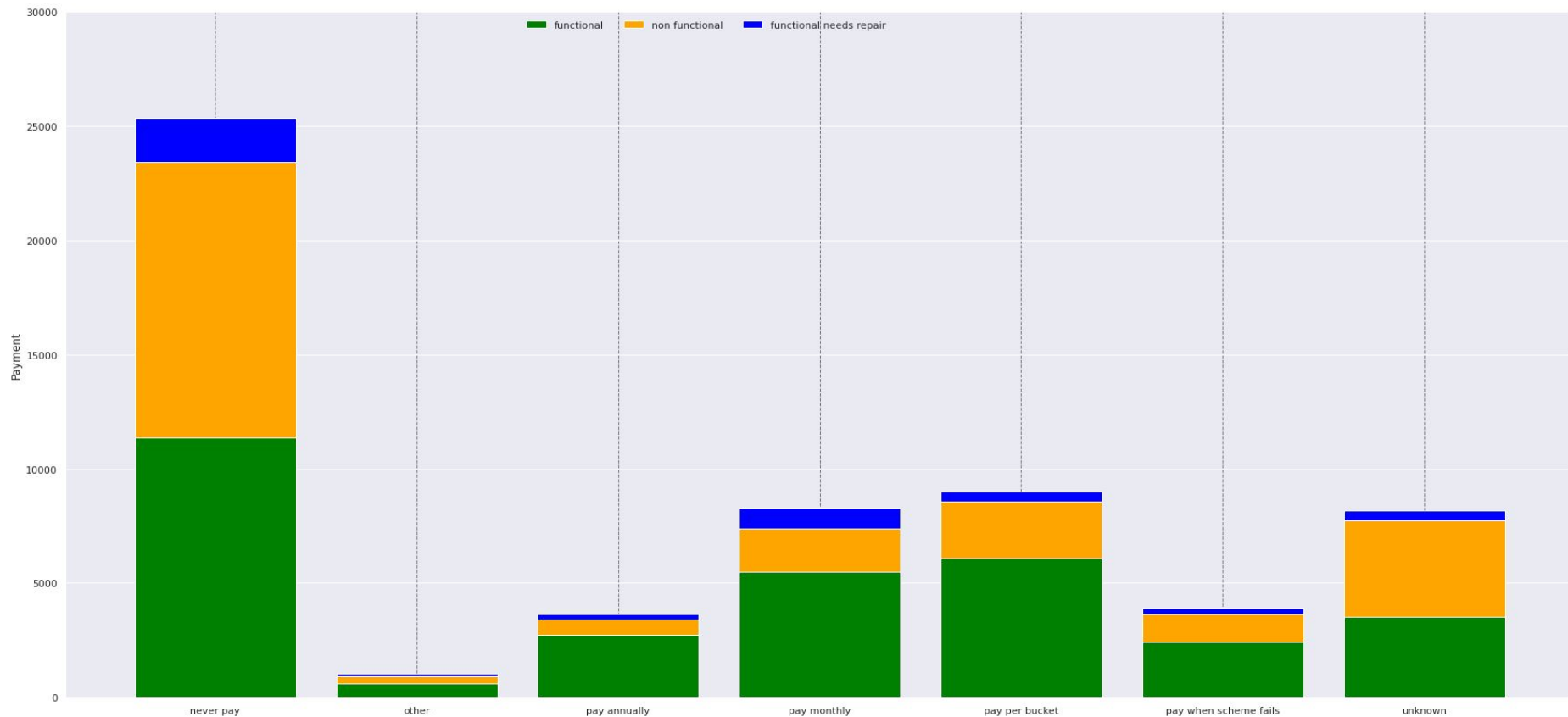
Top 15 funders and status of water pumps



Top 15 Installers and status of water pumps



Type of payment over the status groups



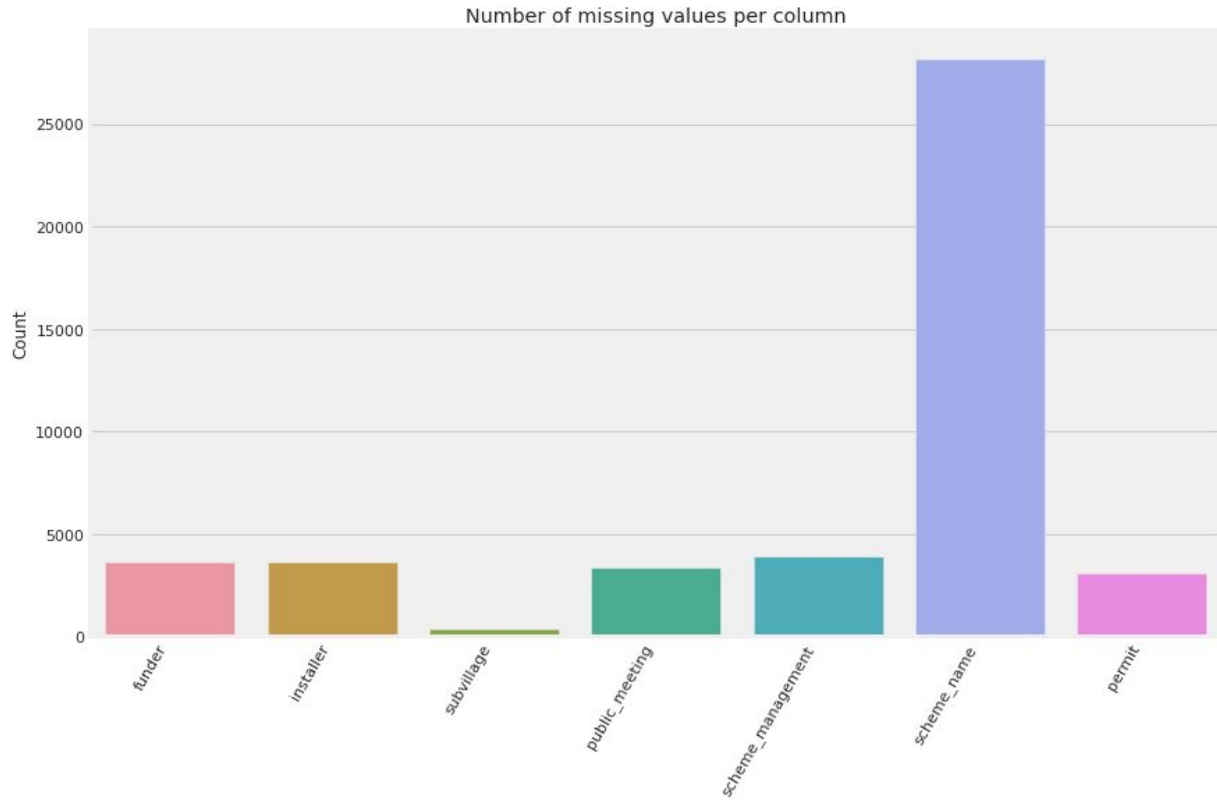


Simona

- Prepare your dataset: encode categorical variables (if any), handle missing variables (if any), generate new features (if you have some intuition that these features can be useful). Preprocess target variable if needed (e.g., combine various classification problems into a single one or convert the target variable to a binary one.) For each transformation give a clear explanation of your motivation. (7 points)

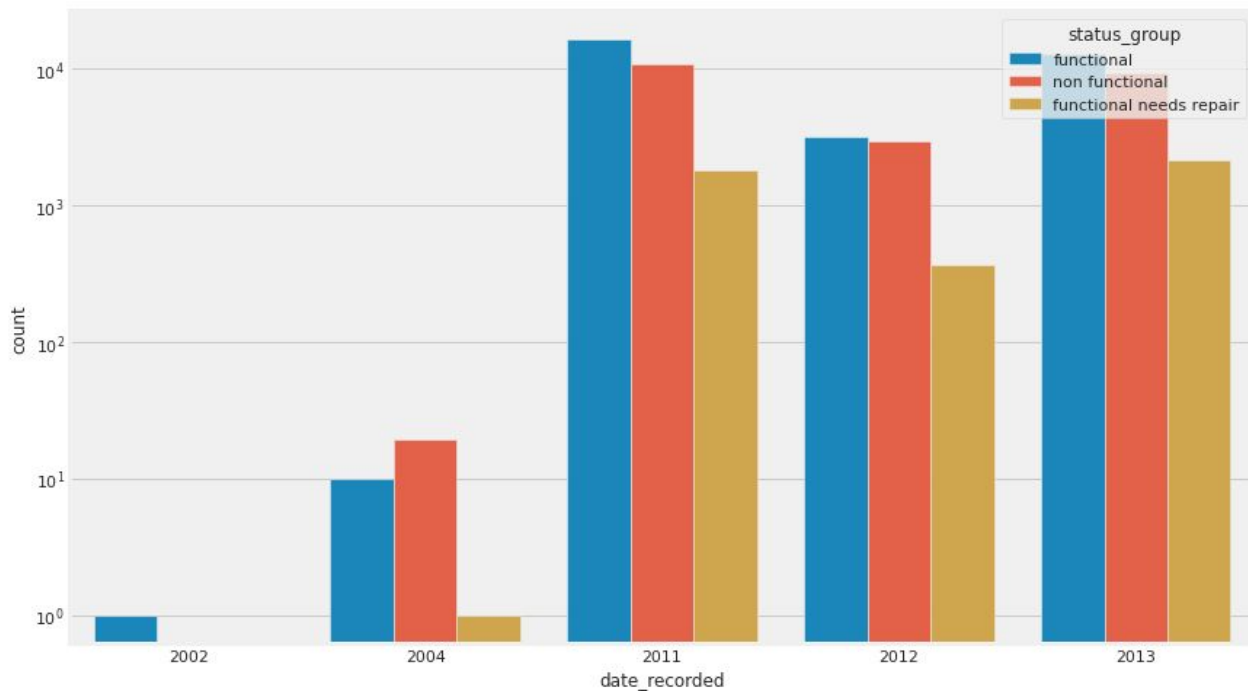
Drop scheme_name

As it has more than 10^4 missing values



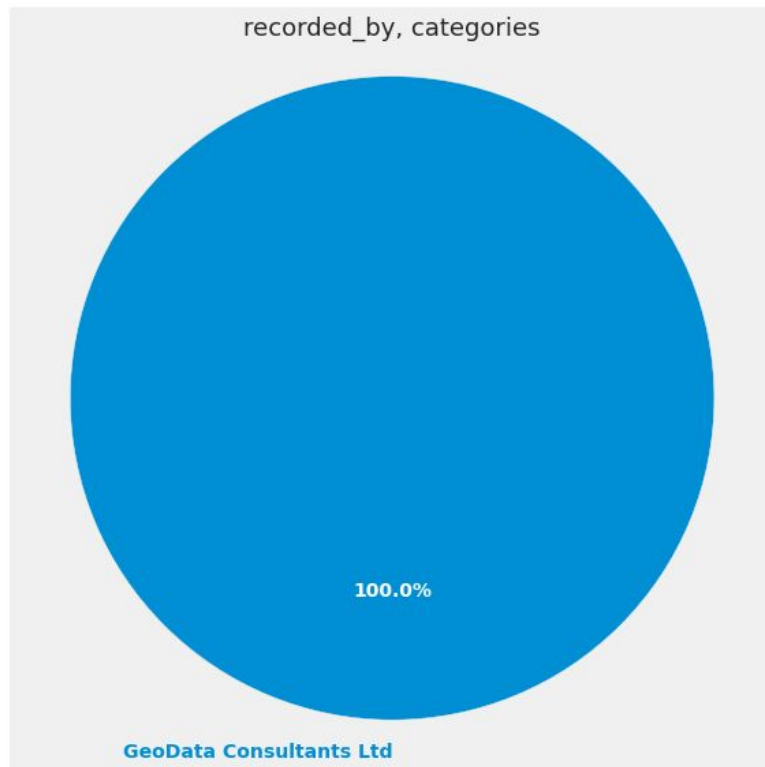
Drop date_recorded

This feature can be considered meaningless, as it only represent the date between 2002 and 2013 when the row was entered in the dataset.



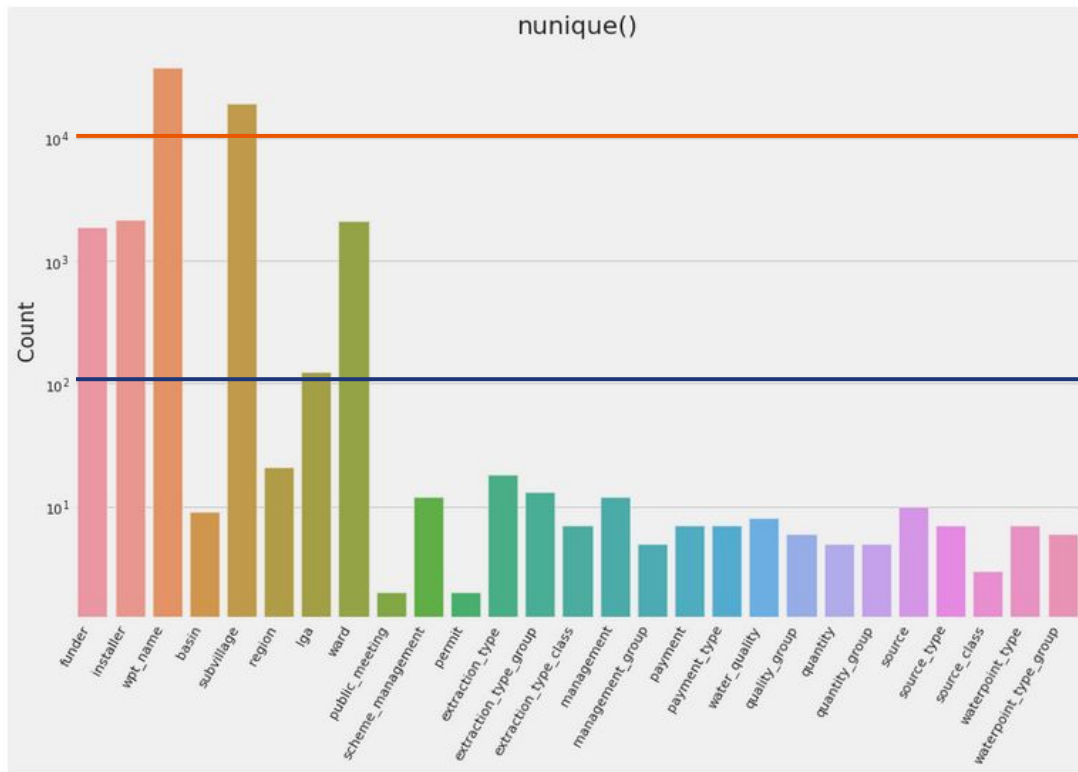
Drop recorded_by

As it has only one category: GeoData Consultant Ltd



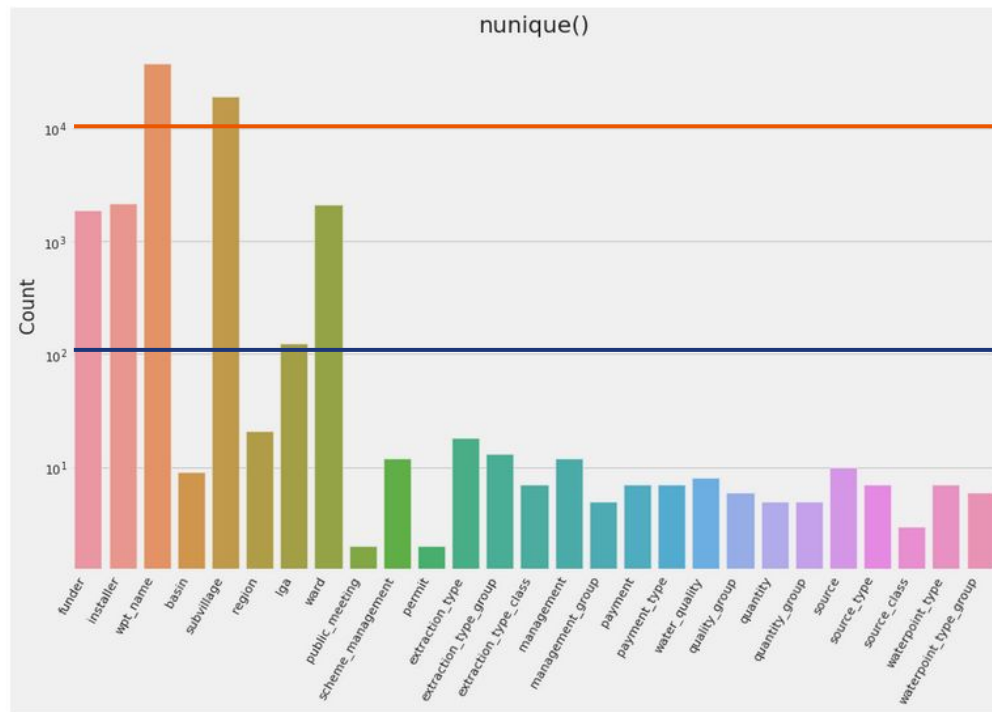
Check number of unique values

Manage the column features based on the number of unique values (nunique):



Columns with `nunique > 104`

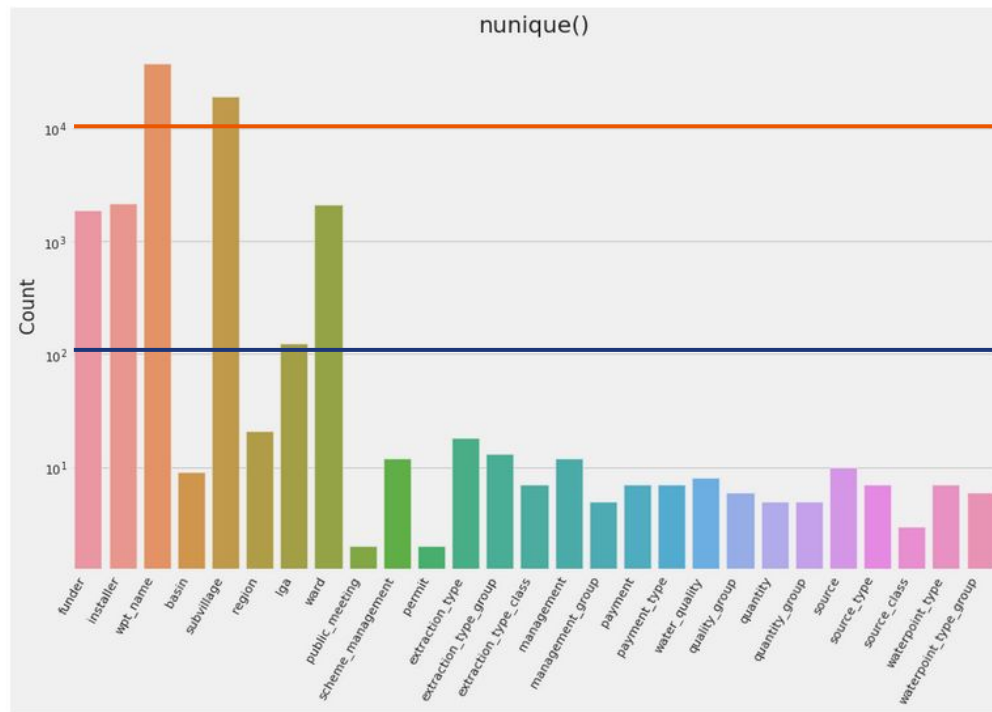
They are dropped from the dataset as their number of unique values (`nunique`) has the same magnitude of the dataset size.



Columns with nunique < 10²

We encode each item in these columns as distinct values.

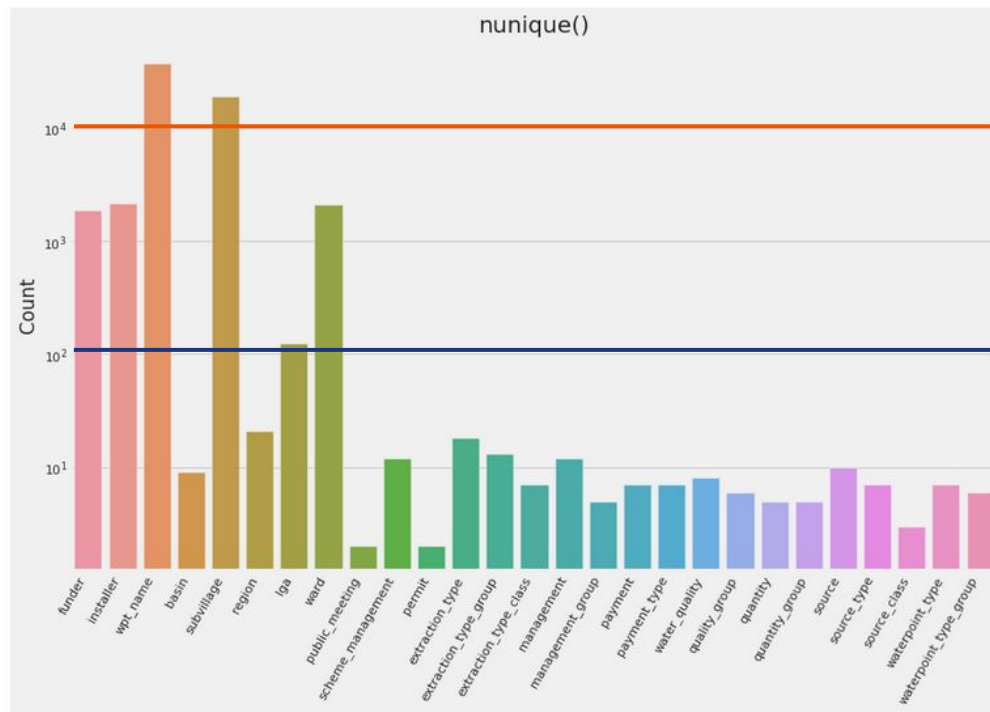
function = pandas.factorize



Columns with $10^2 < \text{nunique} < 10^4$

We reduced the number of categories to less than 10^2 according to the following **pipeline**:

1. Compute the **frequency** with which each value appear
2. Compute the **20th percentile** of the frequency
3. **Keep** the values **above** the 20th percentile
4. **Replace** the values **below** with "Unknown"
5. **Encode** the categorical values into different integers



Particular case: 'installer'

Organization that installed the well.



Once we reduced the number of categories, we noticed that many of them appear with different spelling or typos. Hence, we:

1. Convert all the string in lower case
2. Manually aggregated those with different spelling or typos.

DWE	17402
unknown	3656
<u>Government</u>	1825
RWE	1206
<u>Commu</u>	1060
DANIDA	1050
KKKT	898
Hesawa	840
0	777
TCRS	707
<u>Central government</u>	622
CES	610
<u>Community</u>	553
DANID	552
<u>District Council</u>	551
HESAWA	539
LGA	408
<u>World vision</u>	408
WEDECO	397
TASAF	396
<u>District council</u>	392
<u>Gover</u>	383
AMREF	329
TWESA	316
WU	301
Dmdd	287
ACRA	278
<u>World Vision</u>	270

Drop redundant or similar columns

Checked, by using `df.groupby()`, columns with the same feature description

- 'payment' and 'payment_type'
- 'source' and 'source_type'
- 'waterpoint_type' and 'waterpoint_type_group'
- 'quantity' and 'quantity_group'
- 'water_quality' and 'quality_group'
- 'extraction_type' and 'extraction_type_group'
- 'region_code' and 'region'

```
df.groupby(['payment', 'payment_type']).size()
```


payment	payment_type	
never pay	never pay	25348
other	other	1054
pay annually	annually	3642
pay monthly	monthly	8300
pay per bucket	per bucket	8985
pay when scheme fails	on failure	3914
unknown	unknown	8157

dtype: int64

Figure: Example of identical columns checking

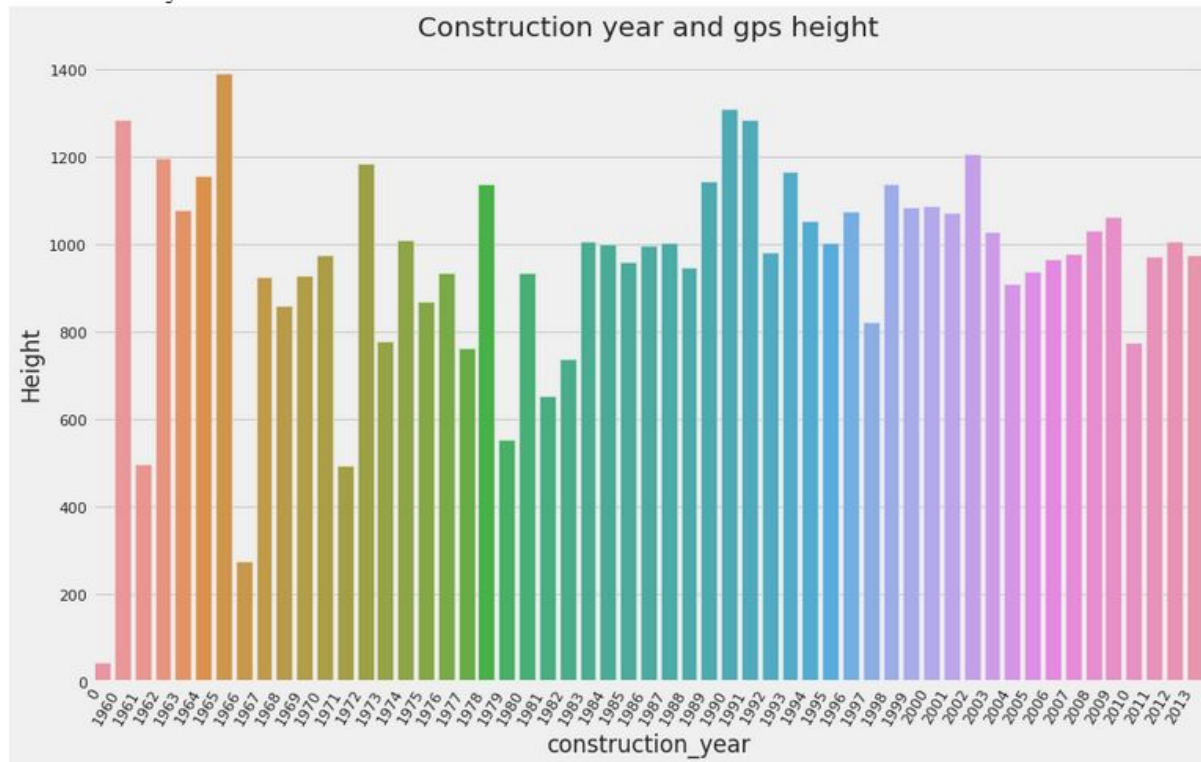
Fill NaNs

NaNs are replaced according to the **type** of the column:

- 
- **object**: fill NaNs in categorical columns with “Unknown”
 - **float64**: fill NaNs in numerical (float) columns with the mean value of that column
 - **int**: fill NaNs in numerical (int) columns with the median value of that column

Remove outliers in 'year' column

Replace the year = 0 with the median value of the column

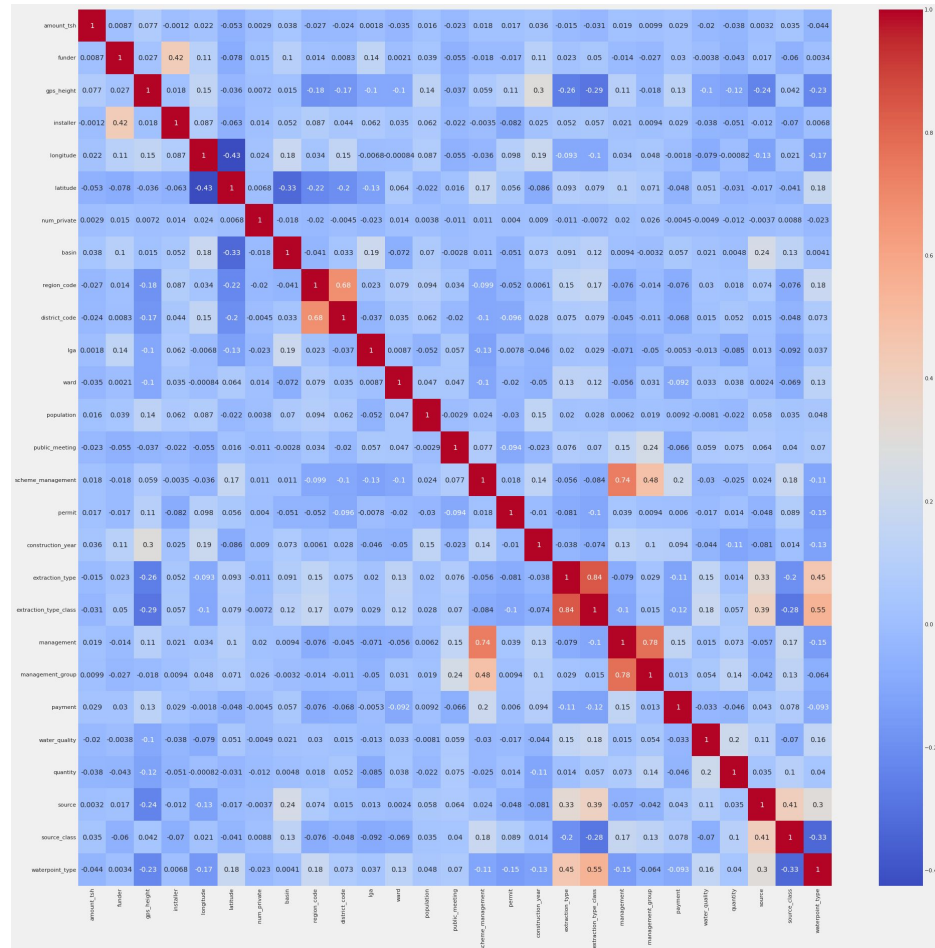


Drop columns with high correlation

When two features have $\text{corr} > 0.65$, one of them is dropped.

- 'extraction_type' and 'extraction_type_class'
- 'management' and 'management_group'
- 'region_code' and 'district_code'
- 'management' and 'scheme_management'

In this case we removed the feature with less number of categorical values



Create new feature: distance

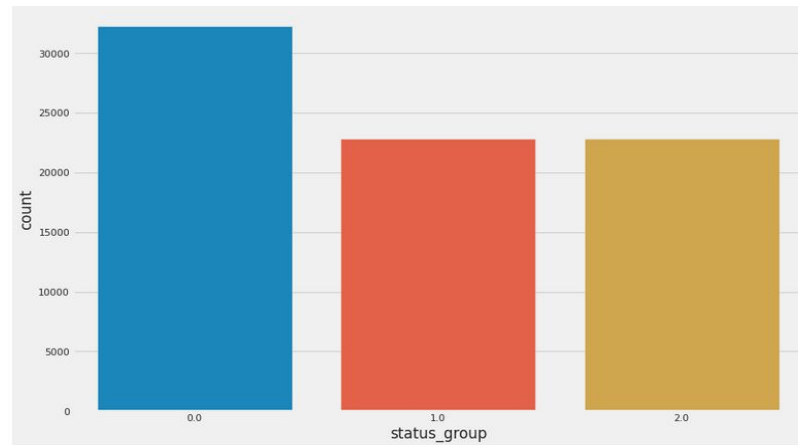
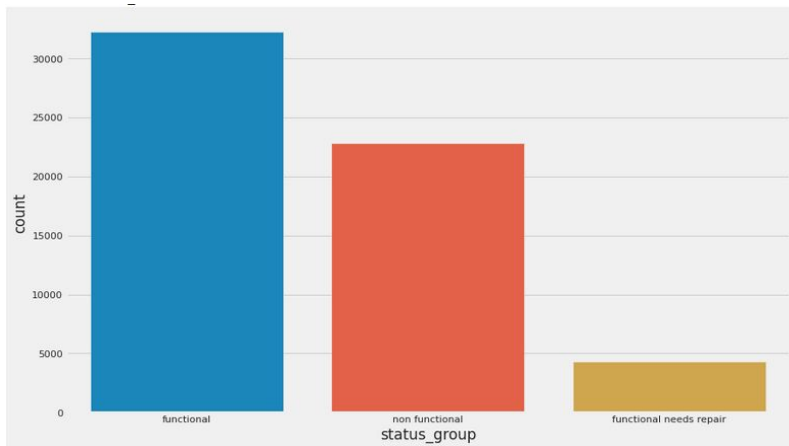
```
df['distance'] = df['longitude']**2+df['latitude']**2
```

To lower the number of columns we substitute the features of 'longitude' and 'latitude', with the sum of squares, to get a sort of distance from the center of the map.



Oversample: SMOTE

As the problem is highly unbalanced we oversample the minority class by using the SMOTE technique, number of neighbors = 7.



We created 18507 synthetic rows, to balance the dataset.

Model choosing



 Logistic regression

 Random forest

 Ensemble KNN

 Gradient boosting.

 CatBoost

Additionally, there was recommendation from Kaggle to use RFC and XGBoost

Metric: f1- macro

Macro averaging is sensitive to errors for rare classes.



Given Macro Average Precision and Recall:

$$\text{MacroAveragePrecision} = \frac{\sum_{k=1}^K \text{Precision}_k}{K}$$

$$\text{MacroAverageRecall} = \frac{\sum_{k=1}^K \text{Recall}_k}{K}$$

Macro F1-Score is the harmonic mean of Macro-Precision and Macro-Recall:

$$\text{Macro F1-Score} = 2 * \left(\frac{\text{MacroAveragePrecision} * \text{MacroAverageRecall}}{\text{MacroAveragePrecision}^{-1} + \text{MacroAverageRecall}^{-1}} \right)$$

Classes with different size are **equally weighted** at the numerator. This implies that the effect of the biggest classes have the same importance as small ones have. high Macro-F1 values indicate that the algorithm has good performance on all the classes, whereas low Macro-F1 values refers to poorly predicted classes.

Basemodel without modified dataset and hyperparameters

```
rfc = RandomForestClassifier()

# max_depth = 90, n_estimators = 400
rfc.fit(X_train, Y_train)

Y_pred=rfc.predict(X_test)
print('f1_score = ',f1_score(Y_test,Y_pred,average='macro'))
print('rfc = ',rfc.get_params())

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4:
after removing the cwd from sys.path.
f1_score = 0.6834787514215327
rfc = {'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None,
```

f1_score for RFC

Is F1 score is enough high, it means, we can use it as a metric.

```
4 knn = BaggingClassifier(KNeighborsClassifier(), max_samples=0.5, max_features=0.5)
5
6 knn.fit(X_train, y_train)
7
8 y_pred = knn.predict(X_test)

/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_bagging.py:645: DataConversionWarning:
y = column_or_1d(y, warn=True)

1 from sklearn.metrics import f1_score
2
3 print(['f1-macro = ', f1_score(y_pred, y_test,average = 'macro')])
4
5 best_param = knn.base_estimator_.get_params()
6 print("Best parameter: ", best_param)

['f1-macro = ', 0.6182691773248422]
Best parameter: {'kneighborsclassifier__algorithm': 'auto', 'kneighborsclassifier__leaf_size': 30, 'kneighborsclassifier__metric': 'minkowski', 'kneighborsclassifier__metric_params': None, 'kneighborsclassifier__n_jobs': None, 'kneighborsclassifier__n_neighbors': 5, 'kneighborsclassifier__p': 2, 'kneighborsclassifier__weights': 'uniform'}
```

f1_score for Ensemble KNN

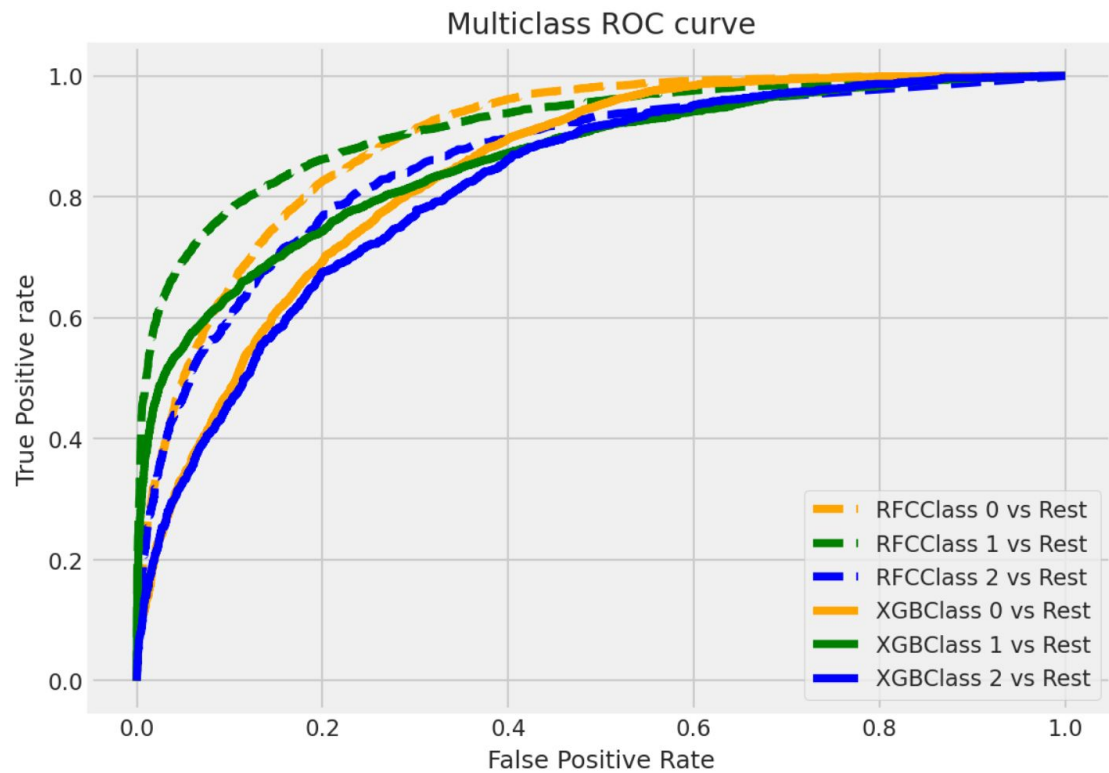
```
# Y_pred = grid_search.predict(X_test)
Y_pred = XGB.predict(X_test)

print('f1_score = ',f1_score(Y_test,Y_pred, average='macro'))
print('XGB = ',XGB.get_params())

/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_data.py:117:
y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_data.py:117:
y = column_or_1d(y, warn=True)
f1_score = 0.5549780819927252
XGB = {'base_score': 0.5, 'booster': 'gbtree', 'colsample_bytree': 1.0,
```

f1_score for XGB

ROC Curves



Let's scale numerical data and use modified Data!

```
[91] num_columns = ['distance', 'construction_year', 'population', 'gps_height', 'amount_tsh']

[92] scaler = StandardScaler()
     scaler.fit(X_train[num_columns])

     X_train[num_columns] = scaler.transform(X_train[num_columns])
     X_test[num_columns] = scaler.transform(X_test[num_columns])
```

```
rfc = RandomForestClassifier()

# max_depth = 90, n_estimators = 400
rfc.fit(X_train, Y_train)

Y_pred_rfc=rfc.predict(X_test)
print('f1_score = ',f1_score(Y_test,Y_pred_rfc))
print('rfc = ',rfc.get_params())

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DataConversionWarning:
  after removing the cwd from sys.path.
f1_score = 0.8495201975760193
rfc = {'bootstrap': True, 'ccp_alpha': 0.
```

```
XGB.fit(X_train, Y_train)

# Y_pred = grid_search.predict(X_test)
Y_pred_xgb = XGB.predict(X_test)

print('f1_score = ',f1_score(Y_test,Y_pred_xgb))
print('XGB = ',XGB.get_params())

/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_gb.py:145: DataConversionWarning:
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_gb.py:145: DataConversionWarning:
  y = column_or_1d(y, warn=True)
f1_score = 0.7967807218410665
```

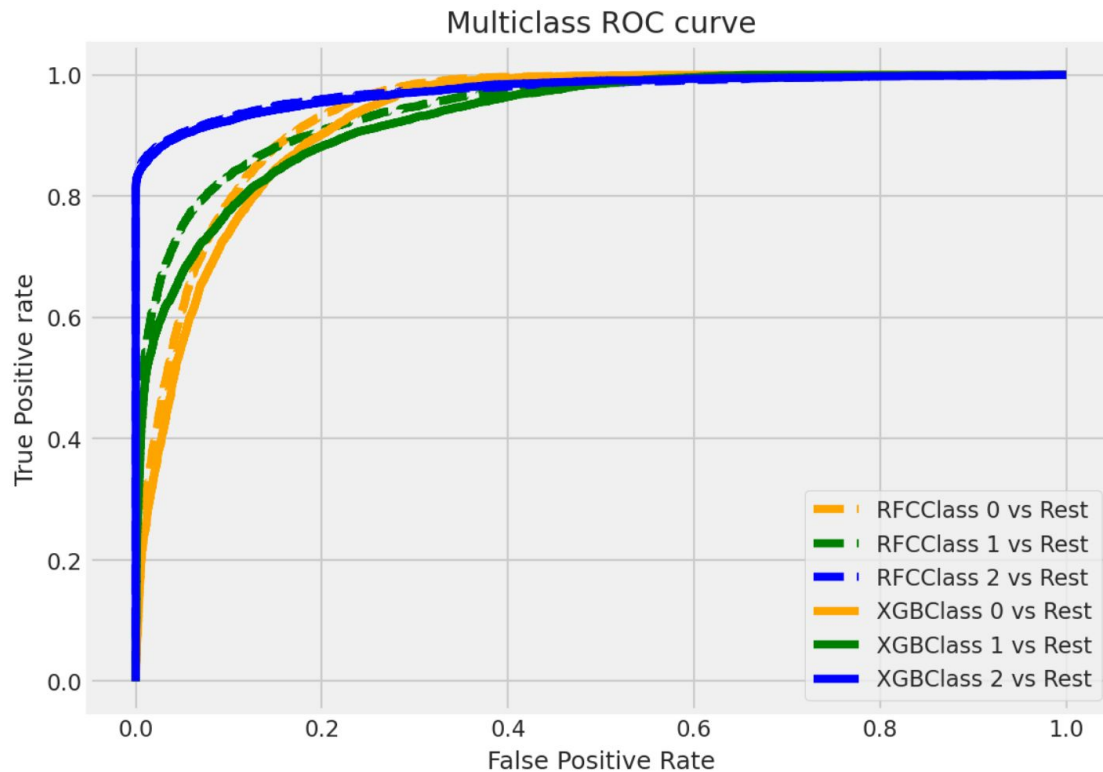
```
2 knn_cls = KNeighborsClassifier(n_neighbors = acc.index(max(acc))+1).fit(X_train,y_train)
3 y_pred = knn_cls.predict(X_test)
4 print(['f1-macro = ', f1_score(y_pred, y_test, average = 'macro')])

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DataConversionWarning:
  f1-macro = ', 0.7636997747632034]

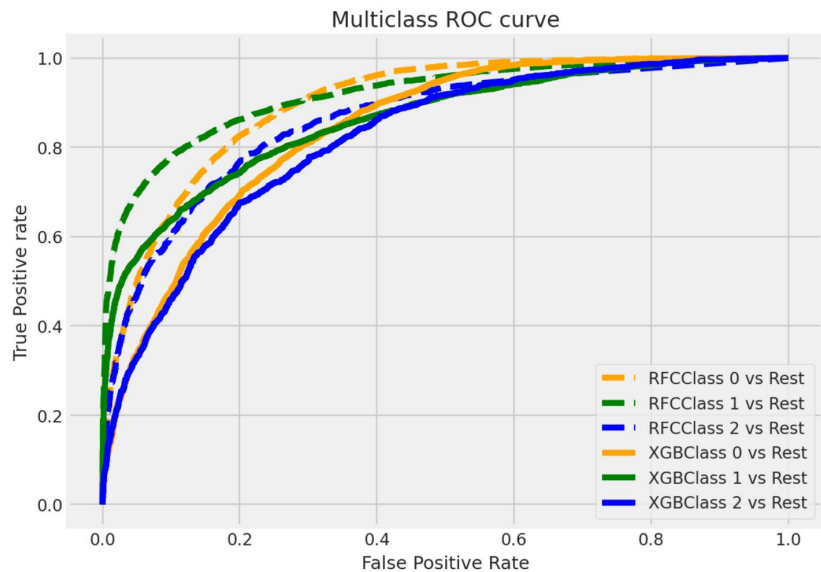
1 bag_cls = BaggingClassifier(base_estimator=knn_cls, n_estimators=25, random_state=0)
2 bag_cls.fit(X_train, y_train)
3 y_pred = bag_cls.predict(X_test)
4 print(['f1-macro = ', f1_score(y_pred, y_test, average = 'macro')])

/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_bagging.py:645: DataConversionWarning:
  y = column_or_1d(y, warn=True)
f1-macro = ', 0.7630929622898127]
```

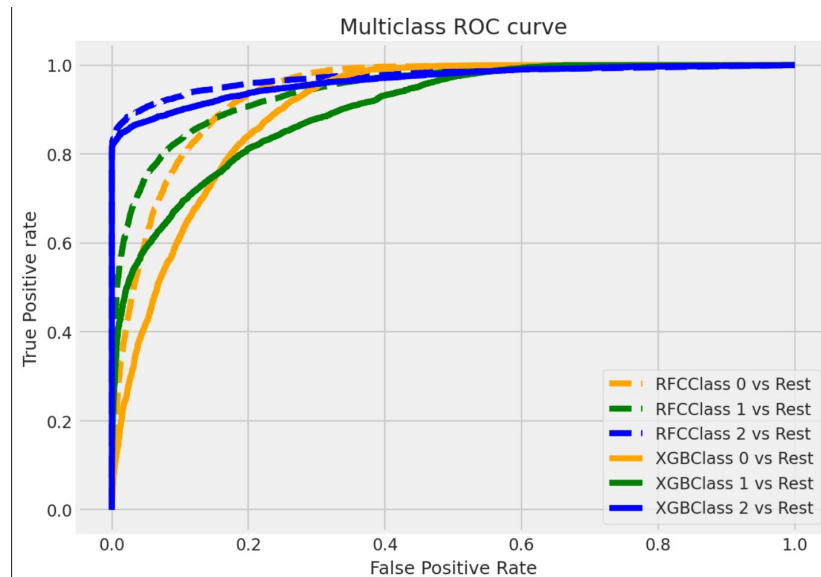
ROC- curves after Grid Search CV



ROC Curves



Before Dataset Modifying



After Dataset Modifying

Table 1. Demographic characteristics of the study population

Oleg: RFC+XGB

Ayush: KNN

- Build a proper cross-validation procedure; select an appropriate measure of quality (the selection of both things should be motivated by your data). Choose an ML model reasonably; look for a good set of hyperparameters. Use the prepared cross-validation procedure to estimate the quality of prediction (9 points).

```
# Create the parameter grid based on the results of random search
param_grid = {
    'max_depth': [80, 90, 100, 110],
    'max_features': ['auto'],
    # 'min_samples_leaf': [3, 4, 5],
    # 'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 400, 600, 800]
}

# Create a based model
rfc_cv = RandomForestClassifier()
```



F1-score

`sklearn.metrics.f1_score`

```
sklearn.metrics.f1_score(y_true, y_pred, *, labels=None, pos_label=1, average='binary', sample_weight=None, zero_division='warn')
```

[\[source\]](#)

Compute the F1 score, also known as balanced F-score or F-measure.


The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

$$F1 = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

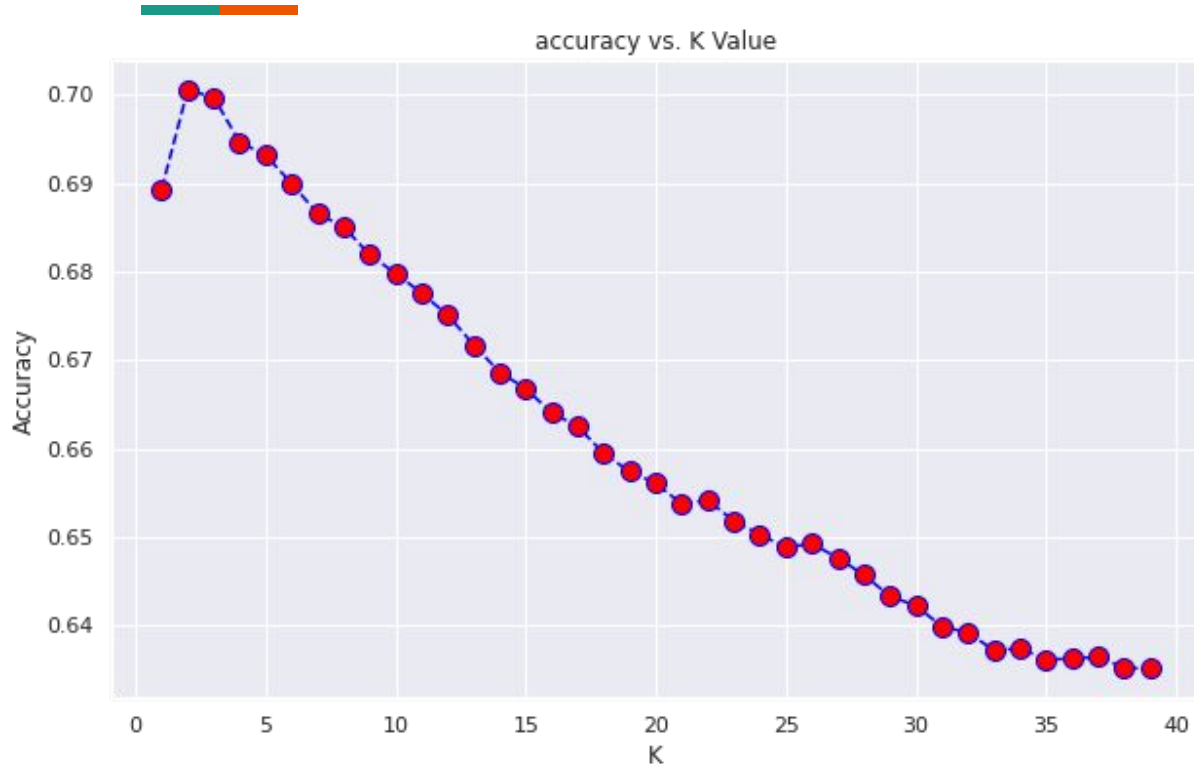
F1 macro for multi classification problem

Oleg: RFC+XGB

Ayush: KNN

- 
- Build a proper cross-validation procedure; select an appropriate measure of quality (the selection of both things should be motivated by your data). Choose an ML model reasonably; look for a good set of hyperparameters. Use the prepared cross-validation procedure to estimate the quality of prediction (9 points).
 1. Pipeline
 2. KFold
 3. ML-model choosing??? not affordable: classification, Regression.
Affordable: Ensemble KNN Classifiers, XGBoost, Randomforest Classifier, Catboost
 4. Cross-Validation
 5. F1 - because we have classification

Ensemble KNN Classifier - Before



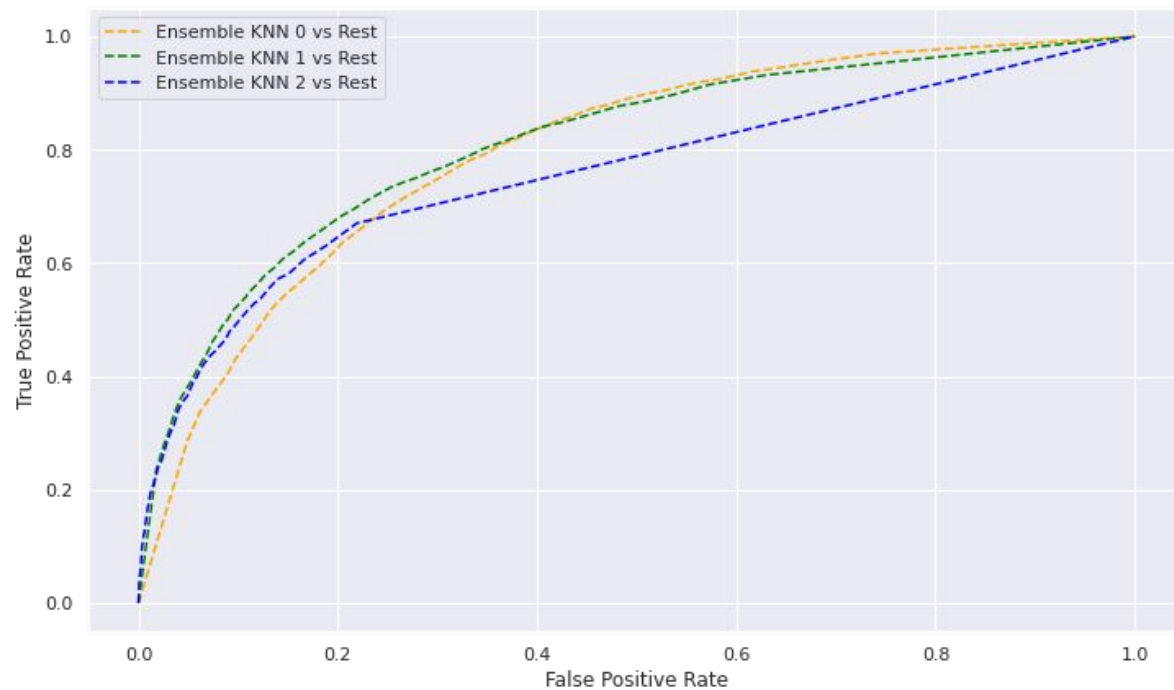
Before modifying the data:

Maximum accuracy:- 0.734
at K = 2 neighbours

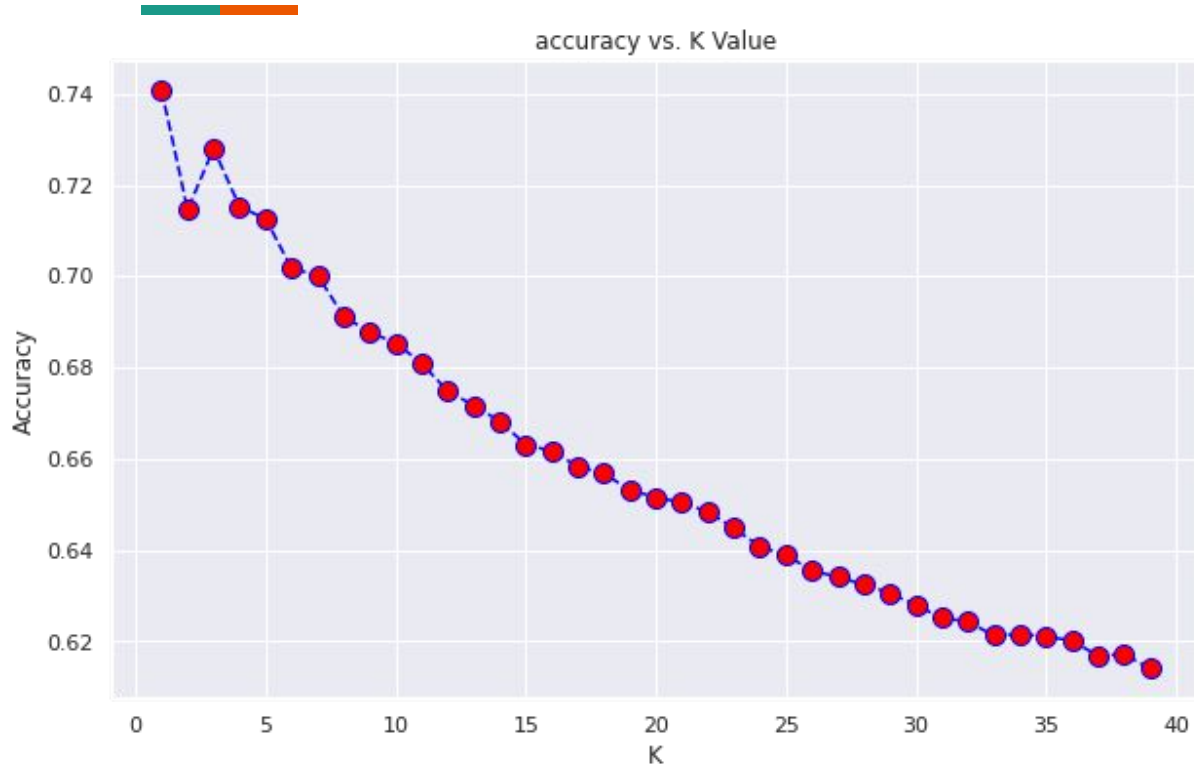
f1-macro (Best KNN) =0.608

f1-macro (Bagging) =0.631

ROC- curves before modifying dataset



Ensemble KNN Classifier - After



After modifying the data:

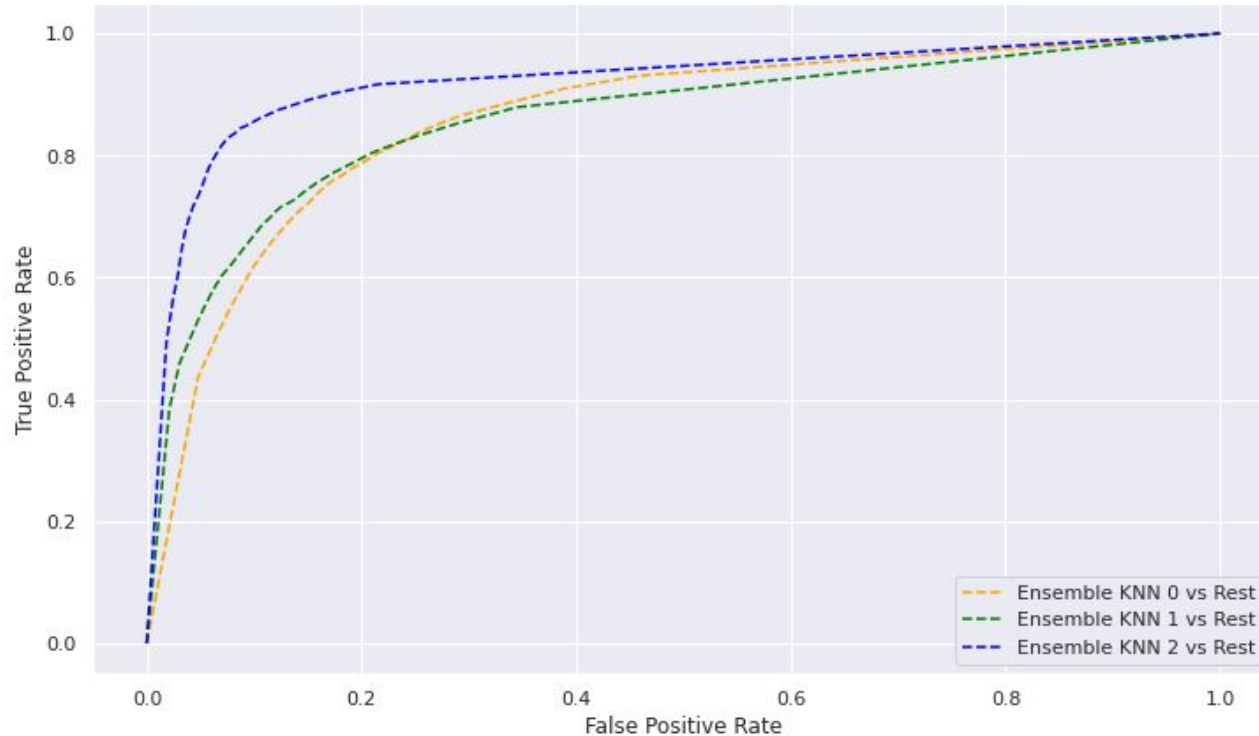
Maximum accuracy:- 0.762

at K = 1 neighbour

f1-macro (Best KNN) =0.763

f1-macro (Bagging) =0.763

ROC- curves before modifying dataset



Let's use Grid Search Cross Validation!



	Before Dataset Modifying	After Dataset Modifying	
Model	Not tuned	Not tuned	Grid Search
RFC	0.69	0.849	0.853
XGB	0.55	0.796	0.831
Catboost	0.66	0.837	0.838
Ensemble KNN	0.631	0.763	0.823



Oleg: RFC+XGB

Ayush: KNN


	Before Dataset Modifiyng		After Dataset Modifiyng	
Model	Not tuned	Tuned	Not tuned	Tuned
RFC	0.69	0.69	0.822	0.852
XGB	0.55	0.55	0.761	0.797
Catboost	0.66		0.822	0.838
Ensemble KNN	0.618	0.631	0.775	

Main parameters
of hypertuning

`'max_depth', 'n_estimators',`

Oleg: RFC+XGB

Ayush: KNN

- 
- Build a proper cross-validation procedure; select an appropriate measure of quality (the selection of both things should be motivated by your data). Choose an ML model reasonably; look for a good set of hyperparameters. Use the prepared cross-validation procedure to estimate the quality of prediction (9 points).
 1. Pipeline
 2. KFold
 3. ML-model choosing??? not affordable: classification, Regression.
Affordable: Ensemble KNN Classifiers, XGBoost, Randomforest Classifier, Catboost
 4. Cross-Validation
 5. F1 - because we have classification

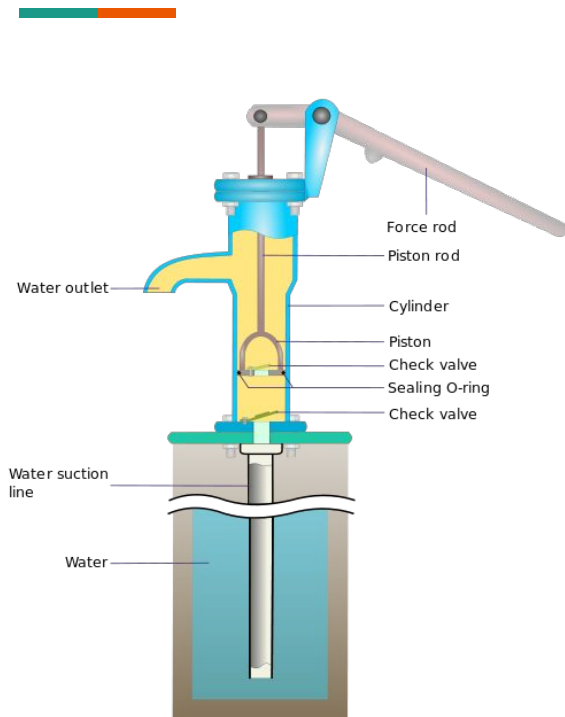


Oleg

- Analyze the obtained results (interesting findings, remarks on ML experiments, the applicability of the model in a real-life scenario) and make an overall conclusion. Does your model solve the problem stated at the beginning? Can you estimate the impact of your ML model on the problem domain (10 points)?

Analyze the results

Remarks on ML experiments:



- Data Processing matters a lot. Much more than grid search and hyperparameter tuning in our case.
- It is really hard to use GridSearch with this amount of data. It takes a lot of time and resources.

Analyze the results



→ Feature engineering is also important part. We substitute some columns by others, dropped correlated data.

Analyze the results



Analyze the results



By predicting the pumps which are functional but needs repair, decreases the overall cost for the Tanzanian Ministry of Water. Which can improve the maintenance operations of the water pumps and make sure that clean, potable water is available to communities across Tanzania.



Arigato Gozaimasu!