Data Mining The Water Table

Capstone 1

Introduction

Many rural communities around the world (especially in developing countries) rely on water pumps as their main source of water. This includes water for drinking, cooking, cleaning, growing crops, and more. Obviously if a community’s water pump fails, it can be extremely disruptive to daily life.

So the question is, how can we predict which water pumps are faulty, and which are operational? Using data from the Tanzania Ministry of Water, we are going to make predictions on which water pumps in rural Tanzania are functional, functional but in need of repair, or non functional.

Data Wrangling

For this project, the data come from DrivenData.org, a non-profit organization which hosts data science competitions that have a social impact. As such, the data are for the most part provided in a clean, tidy format, and require little data wrangling to be ready for analysis. However, we will impute missing values with the median and remove variables which contain redundant information.

Data Story

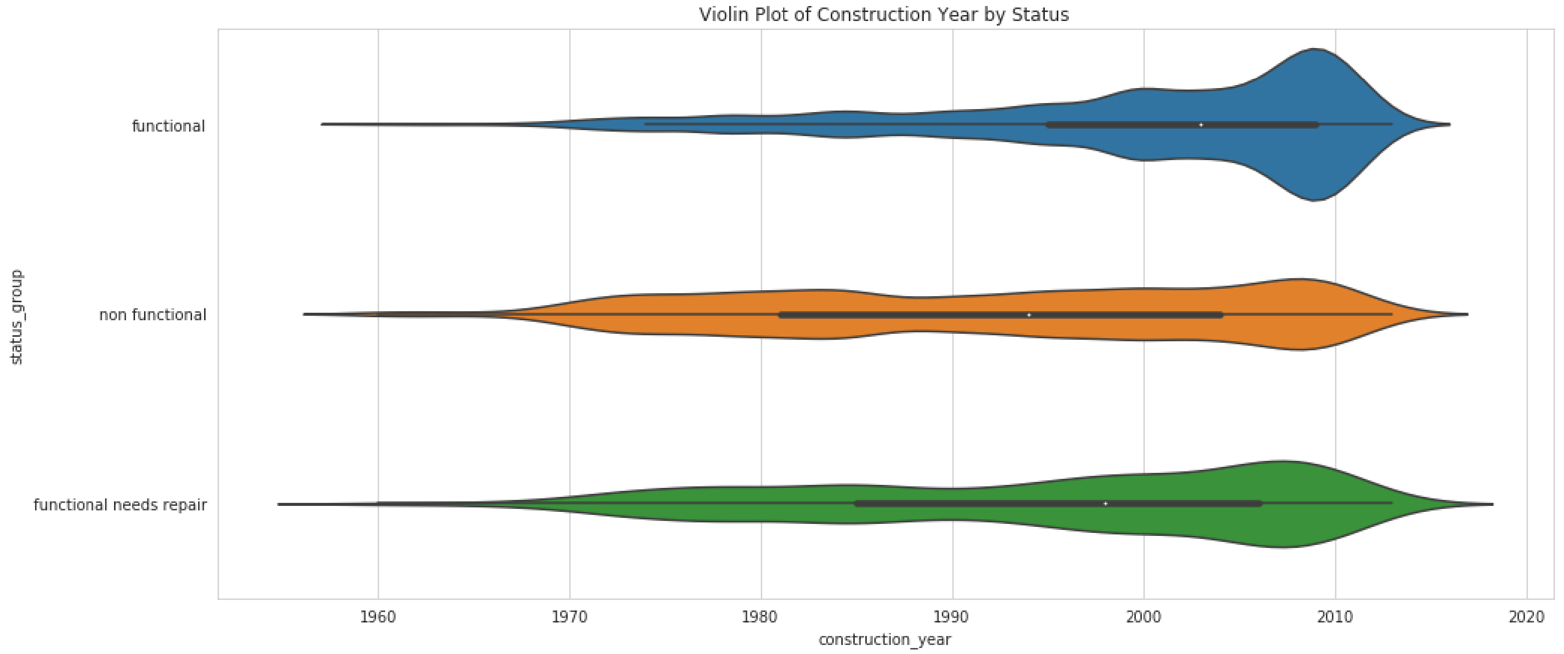
We have a data set where each row represents a water well in Tanzania, and each column is a variable. After removing some redundant features, we have a data set that looks like this:

Out[10]:

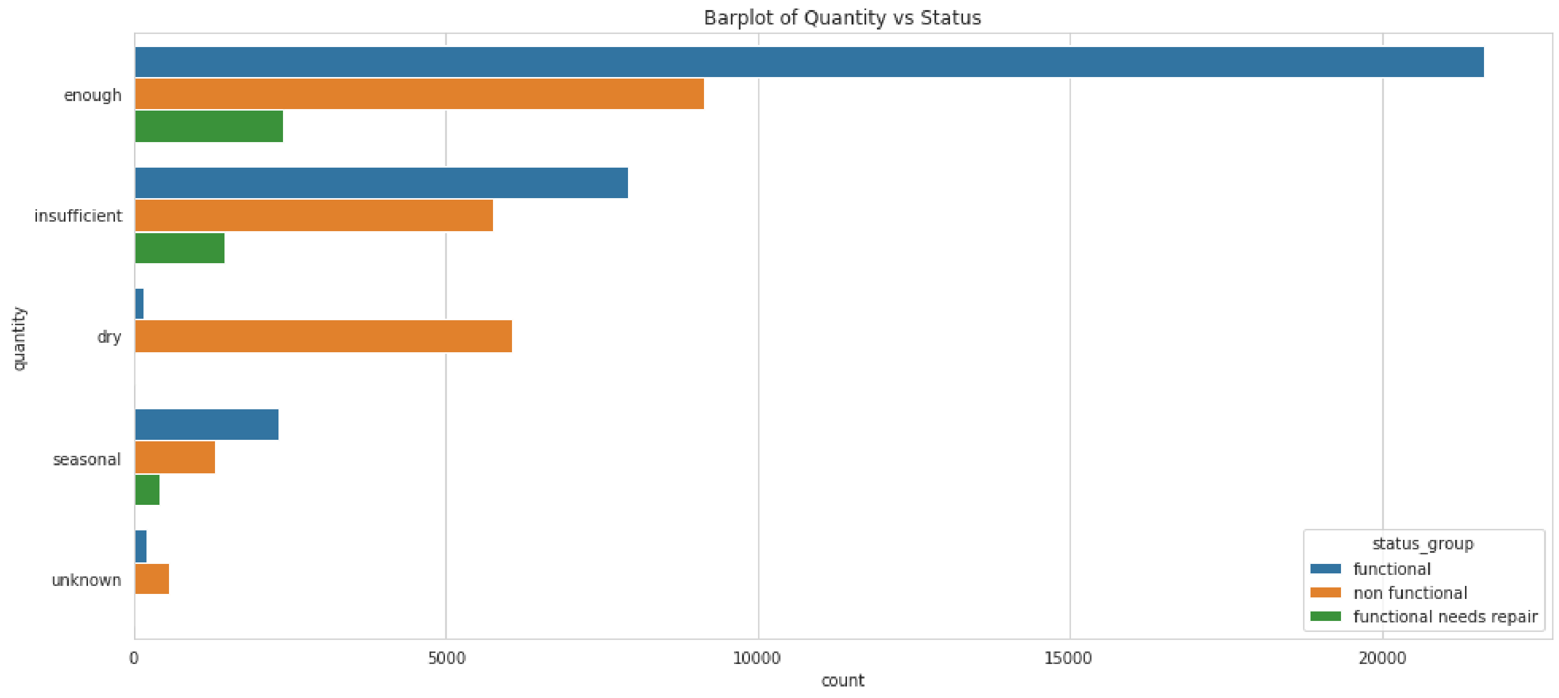
	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	basin	region	district_code	...	populat
0	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	Lake Nyasa	Iringa	5	...	109
1	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Lake Victoria	Mara	2	...	280
2	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	Pangani	Manyara	4	...	250
3	0.0	2013-01-28	Unicef	263	UNICEF	38.486161	-11.155298	Ruvuma / Southern Coast	Mtwara	63	...	58
4	0.0	2011-07-13	Action In A	0	Artisan	31.130847	-1.825359	Lake Victoria	Kagera	1	...	0

5 rows × 21 columns

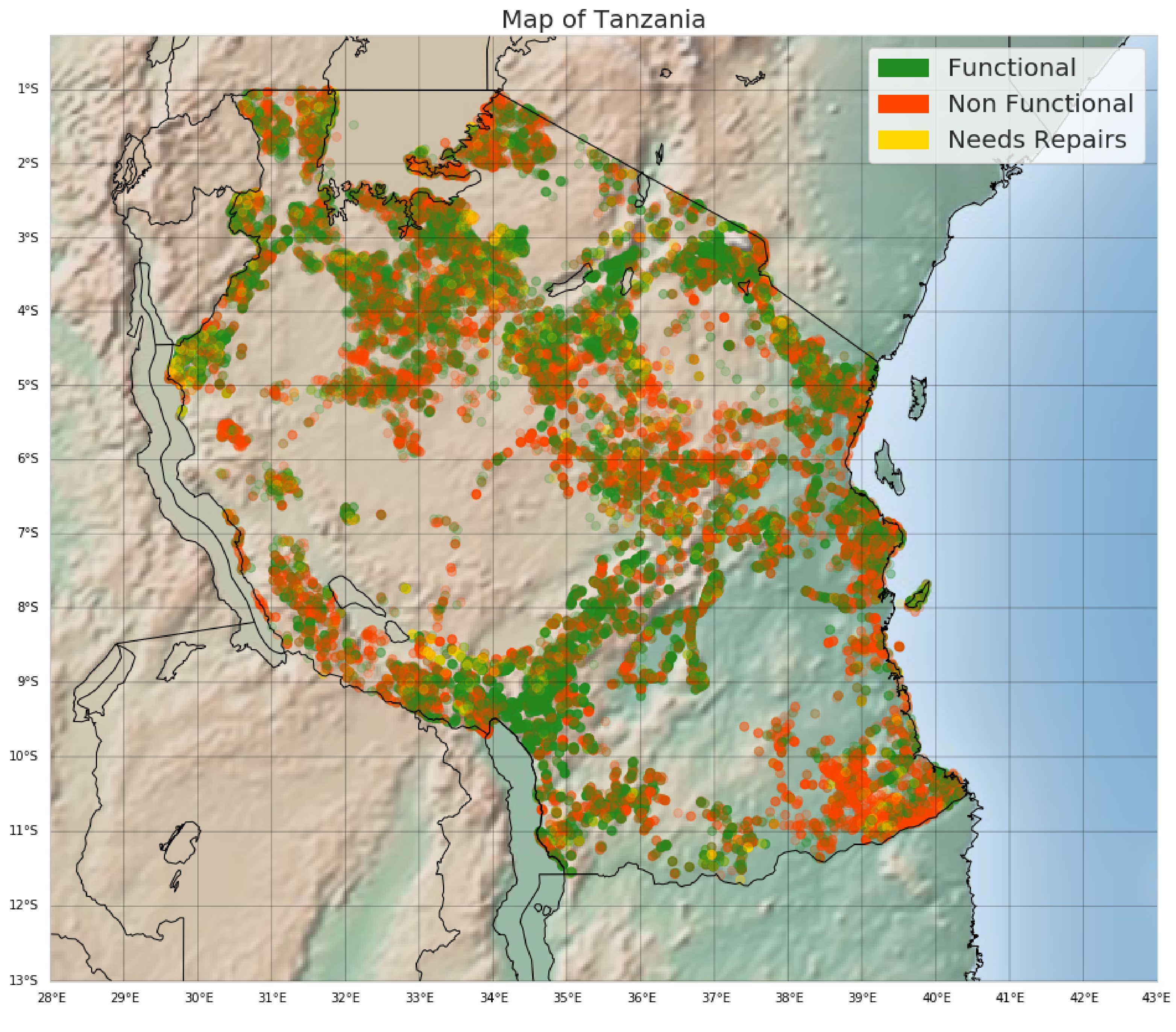
We'll look at some violin plots of construction_year grouped by status_group, and we can see that it seems wells which are functional are more likely to have been constructed more recently. This makse sense, older wells which have been in operation longer may be more likely to break down and need repairs or stop working altogether.


Violin Plot of Construction Year by Status
status_group
functional
non functional
functional needs repair
construction_year
1960
1970
1980
1990
2000
2010
2020

Next we'll take a look at the categorical variables and see if we might find any insights. For example, when we look at a barplot of "quantity" grouped by well status, we noice that wells which are "dry" are much more likely to be non functional than functional, and conversely wells which have "enough" water are more likely to be functional. This may be a useful variable when we build our model.


Barplot of Quantity vs Status
enough
insufficient
dry
seasonal
unknown
quantity
0
5000
10000
15000
20000
count
status_group
functional
non functional
functional needs repair

We'll look at a map of Tanzania with the wells represented as points. The points are color coded to show which are functional, non functional, and in need of repairs.



Map of Tanzania
Functional
Non Functional
Needs Repairs
1°S
2°S
3°S
4°S
5°S
6°S
7°S
8°S
9°S
10°S
11°S
12°S
13°S
28°E
29°E
30°E
31°E
32°E
33°E
34°E
35°E
36°E
37°E
38°E
39°E
40°E
41°E
42°E
43°E

It's difficult to see if there are any meaningful patterns in the map. There appear to be a few patches where there are more wells which are functional or non functional: for example at roughly 9 degrees S and 34.5 degrees E there are a lot of functional wells, and at about 10.5 degrees S and 39 degrees E there are many non-functional wells. This may or may not be useful data for making our model.

Statistical Inference

Next we'll run some hypothesis tests to see if we can find any helpful information about which features might be significant predictors of well functionality. Since we want to compare differences in means across different sized groups, we will use Welch's two sample T-test. We'll run the tests pair-wise to compare the 'functional', 'non functional', and 'functional needs repair' groups. We'll set our alpha to equal 0.01.

```
First test: functional vs non functional
Mean longitude for functional wells: 35.191149
Mean longitude for non-functional wells: 35.239450
P-value for Welch's T-test: 0.0351669196

Second test: functional vs needing repairs
Mean longitude for functional wells: 35.191149
Mean longitude for wells needing repairs: 34.309867
P-value for Welch's T-test: 0.0000000000

Third test: non functional vs needing repairs
Mean longitude for non-functional wells: 35.239450
Mean longitude for wells needing repairs: 34.309867
P-value for Welch's T-test: 0.0000000000
```

After running many hypothesis tests, we’ve found almost every p-value to be significant. Part of this is due to the fact that we have so many data observations. The only test that doesn't show significance is the average longitude for functional vs non functional wells. It might be a good indicator that longitude is not a significant factor in determining if a well is functional or not functional, since despite the large amount of data, we did not get significance.

Feature Engineering

Next we'll want to do some feature engineering, to see if we can extract some useful information out of the variables. We're going to create a new variable called 'age' which will represent the age of each well (in years) at the time the data observation was recorded. We will compute this by finding the difference between 'construction_year' and 'date_recorded'.

Out[202]:

	funder	gps_height	installer	longitude	latitude	basin	region	district_code	lga	population	construction_ye
0	Roman	1390.0	Roman	34.938093	-9.856322	Lake Nyasa	Iringa	5	Ludewa	109.0	1999.0
1	Grumeti	1399.0	GRUMETI	34.698766	-2.147466	Lake Victoria	Mara	2	Serengeti	280.0	2010.0
2	Lottery Club	686.0	World vision	37.460664	-3.821329	Pangani	Manyara	4	Simanjiro	250.0	2009.0
3	Unicef	263.0	UNICEF	38.486161	-11.155298	Ruvuma / Southern Coast	Mtwara	63	Nanyumbu	58.0	1986.0
4	Action In A	369.0	Artisan	31.130847	-1.825359	Lake Victoria	Kagera	1	Karagwe	25.0	1986.0

Machine Learning

Now it's finally time to start building our model. We'll start by splitting our data into train and validation sets, so we can check the accuracy. Then we'll want to start with a very simple model to use as a baseline. From this, we'll check the accuracy and use it to compare. Hopefully, more complicated models will produce more accurate results (if not, we'll know something didn't go right). We'll be using a decision tree for this benchmark.

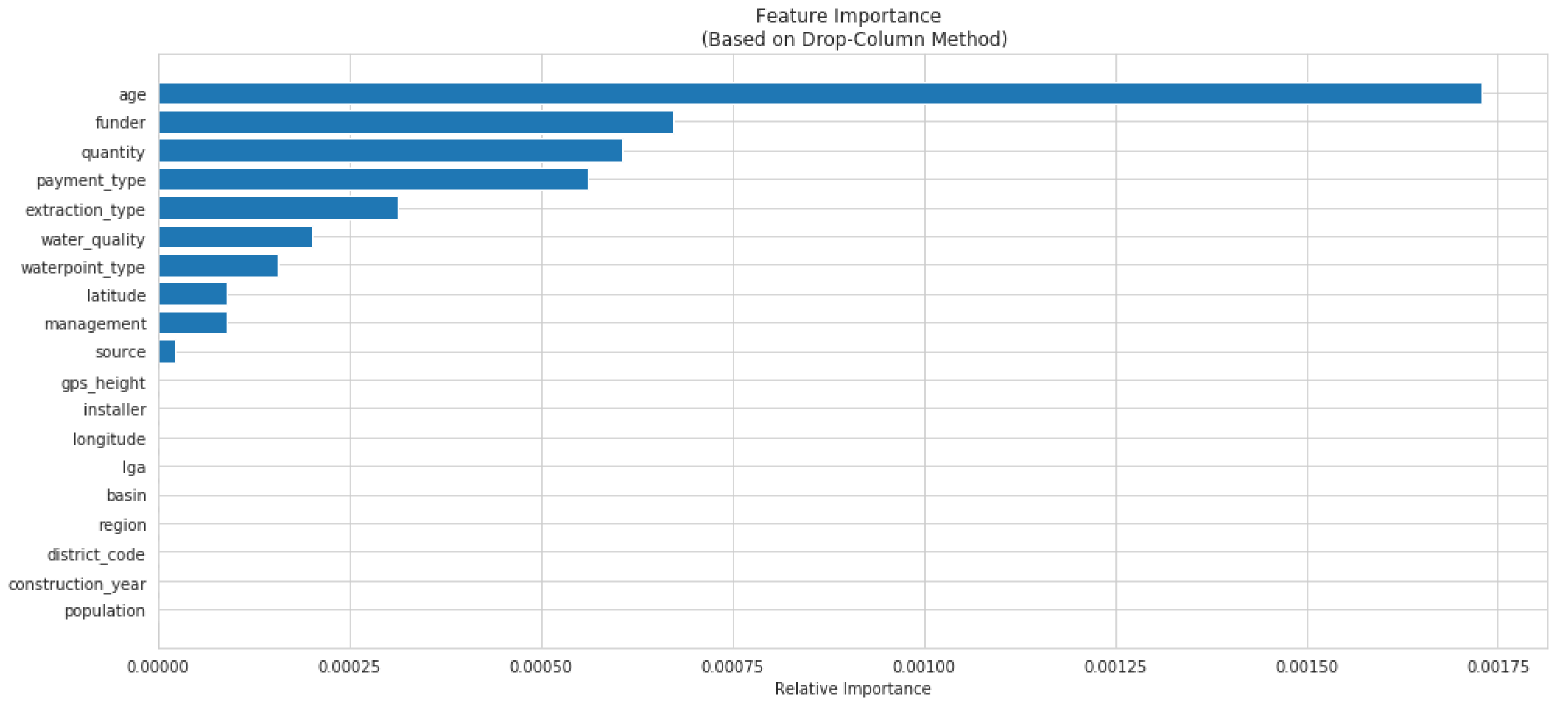
```
Decision Tree benchmark accuracy: 0.7571717171717172

First Random Forest model accuracy: 0.8061952861952862
```

Our benchmark to beat is about 0.757. Our first attempt at building a random forest model gave an accuracy of about 0.806, not bad.

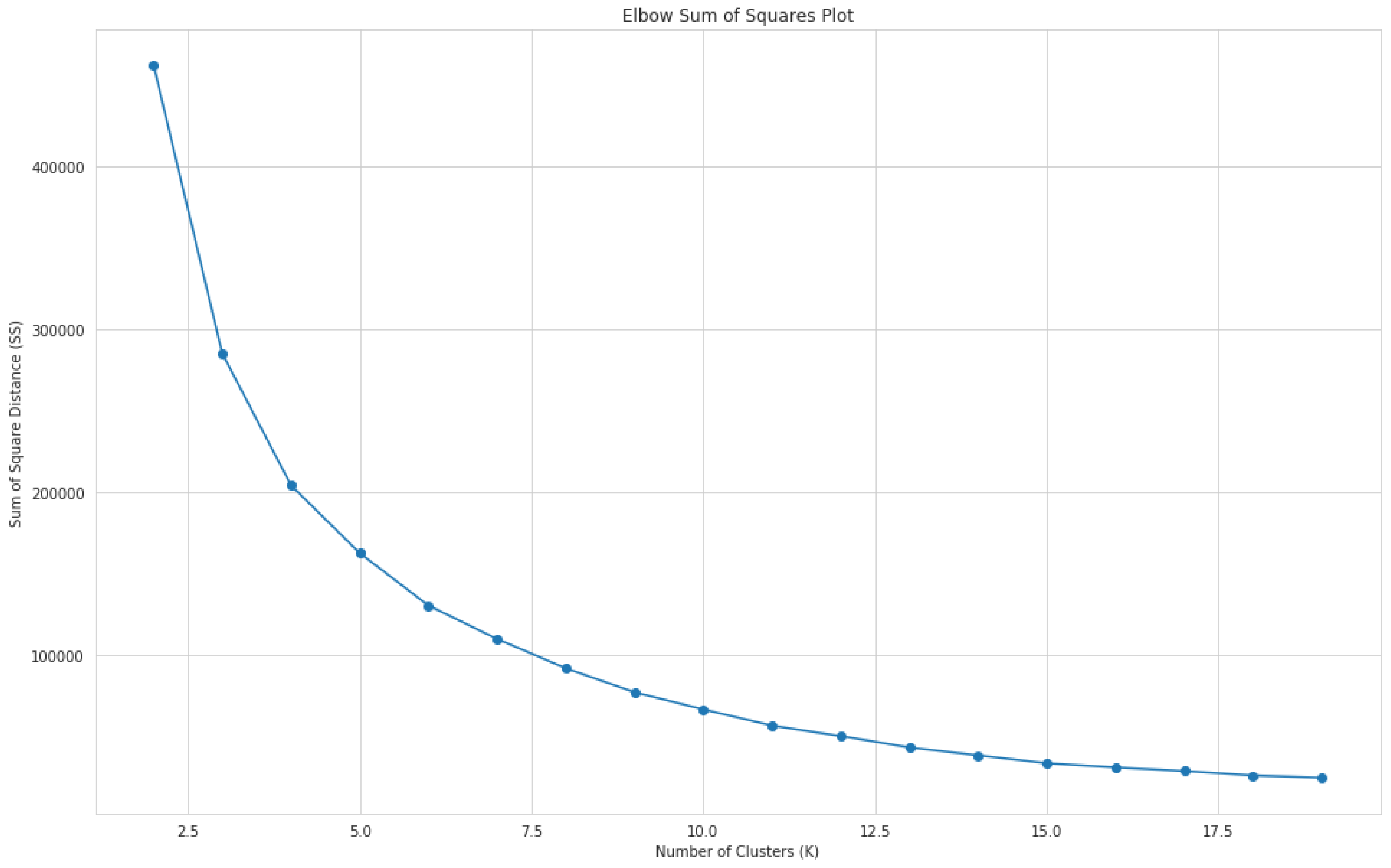
Feature Importance

One problem we might have is overfitting. To combat this, we are first going to look at feature importance, to determine which variables are actually important for our model.


Feature Importance
(Based on Drop-Column Method)
age
funder
quantity
payment_type
extraction_type
water_quality
waterpoint_type
latitude
management
source
gps_height
installer
longitude
lga
basin
region
district_code
construction_year
population
0.00000
0.00025
0.00050
0.00075
0.00100
0.00125
0.00150
0.00175
Relative Importance

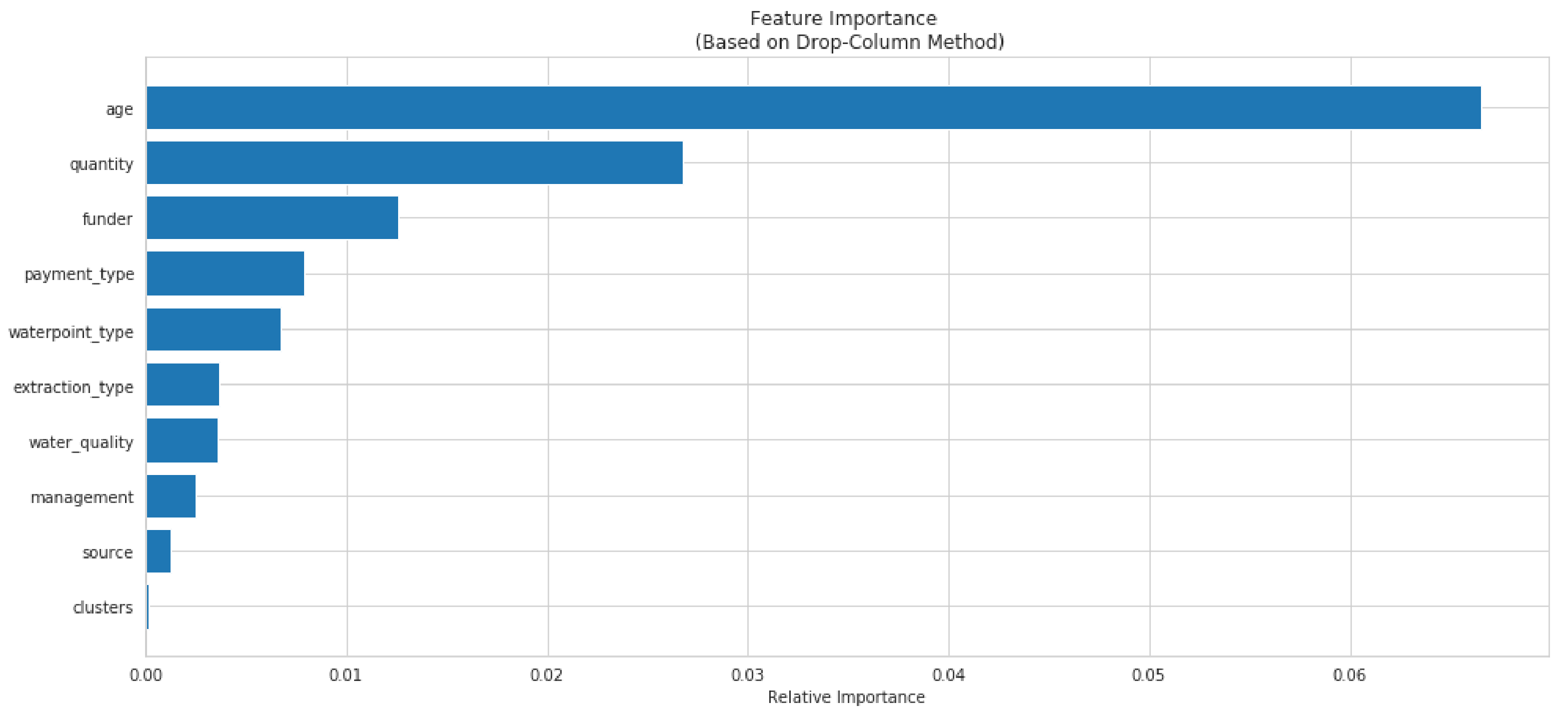
We have some interesting results. Longitude does not appear to have any importance at all, along with several other variables. Also interesting, the age variable we created earlier, appears to be the most important feature. So it’s a good thing we took the time to do some extra feature engineering!

With that in mind, we'll attempt to do some extra feature engineering, and see if we can take the latitude and longitude data and transform it into a feature that might be useful. We'll try using KMeans clustering to transform the data. We'll need to test out different values for k (the number of clusters) and compare the sum of squares using an "elbow" plot. Where the bend appears in the plot will indicate which value we might want to choose for k.



Elbow Sum of Squares Plot
Sum of Square Distance (SS)
400000
300000
200000
100000
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Number of Clusters (K)

There appears to be a bend or “elbow” in the curve at k = 4, so that’s what we’ll choose for our algorithm. We’ll organize the data into these four clusters, and add the cluster assignment as a new feature in our data set. We’ll then drop the features that were determined to be unimportant, and take another look at feature importance.


Feature Importance
(Based on Drop-Column Method)
age
quantity
funder
payment_type
waterpoint_type
extraction_type
water_quality
management
source
clusters
0.00
0.01
0.02
0.03
0.04
0.05
0.06
Relative Importance

These results show our feature engineering did not prove to be effective. The "clusters" feature we created based on latitude and longitude was almost completely unimportant in our model. It appears organizing the latitude and longitude data into clusters did not really accomplish much. For this reason we will not be using this feature going forward, and we will use only the features found to be important for the next step.

Hyperparameter Tuning

The next step we’ll want to take is to fine-tune some hyperparameters of our model. We’ll be using a new data set with all the unimportant features (which we’ve just uncovered) dropped from the dataframe. We’ll run a grid search over several different parameter options, and select the best parameters.

First we'll train a model using the same parameters as before, but with the unimportant columns dropped. This will help us get an idea for how well our parameter tuning is working.

```
Random Forest accuracy after dropping unimportant variables: 0.7835016835016835
```

```
Fitting 2 folds for each of 100 candidates, totalling 200 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:706:
UserWarning: A worker stopped while some jobs were given to the executor. This can be
caused by a too short worker timeout or by a memory leak.
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:  9.4min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed: 40.7min
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed: 57.4min finished

{'n_estimators': 1000, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features':
'sqrt', 'bootstrap': True}


Random Forest accuracy after hyperparameter tuning: 0.8005387205387205
```

After running the grid search and using the best hyperparameters to build our model, we have our final results. It seems accuracy is about 0.801, a slight decrease from our original model, but an increase over the benchmark we just ran, which had a score of about 0.784. So parameter tuning definitely improved our results, even though we built the model on less data. Hopefully we also managed to avoid over fitting the data. All that is left is to make a prediction using the final test set, convert the results to the proper format, and submit it.