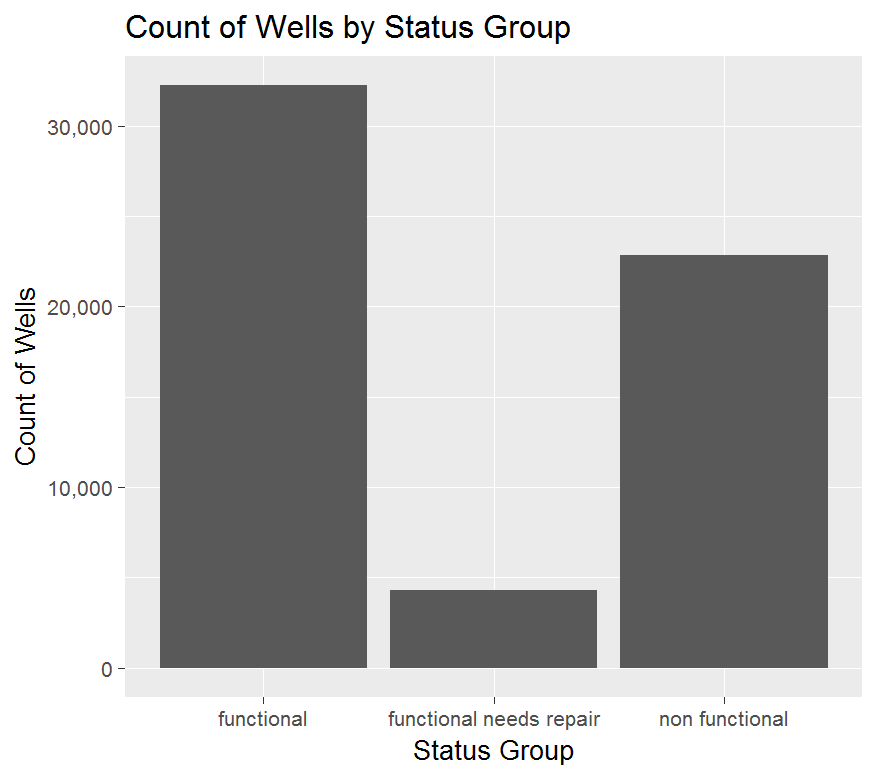
# Introduction

DrivenData, a data mining competition much like Kaggle, has provided a challenge called *Pump it Up: Data Mining the Water Table*. The task for data miners working on *Pump it Up* is to use the data provided on Tanzanian water systems to classify wells into one of three status groups: ”functional”, “functional needs repair”, or “non functional”. The Tanzanian Water Ministry has provided data with the hopes of creating a better understanding of which pumps fail which would aid their efforts to maintain safe drinking water for all Tanzanians.[[1]](#footnote-2) Data is provided for nearly 60,000 wells with 40 different features with information on factors such as geography, water quality, and governance of the well.

I explored several different models trying to classify these wells including a J48 decision tree, a k-Nearest Neighbors learner, a random forest, and a model using the AdaBoostM1 algorithm. The random forest proved to be the best classifier based on a number of metrics, including accuracy.

# Exploratory Data Analysis

Though there are three categories, the “functional needs repair” group is small relative to the others. The chart below shows the number of wells in each class for the training data.



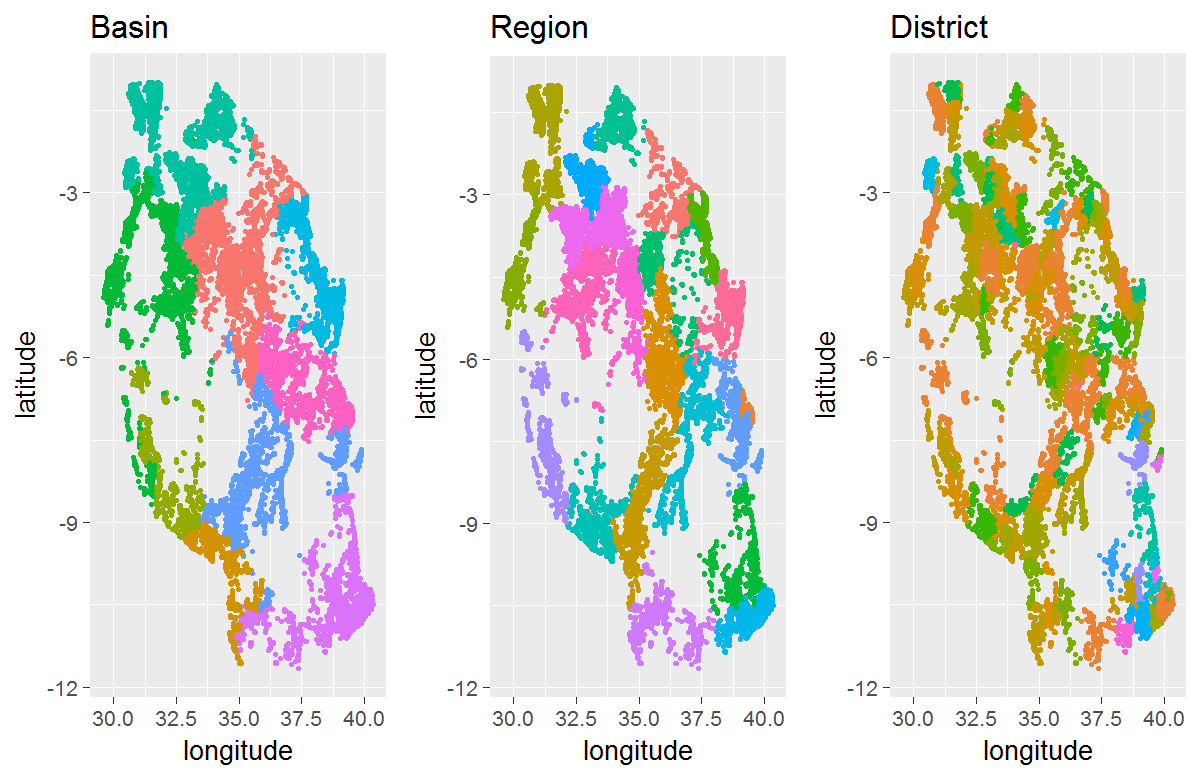
The primary goal of this analysis, as I see it, is to identify wells that have failed. Given that the pumps that need repair but are still functional represent a small portion of the total number of wells, it makes sense to look specifically at the wells that are non-functional. This is particularly useful in for EDA because it helps condense the information. For example, looking at the percent of wells that are in the class “non functional” (NF rate) across extraction type categories is much easier to interpret than trying to visualize several rates at one time.

## Geographic Variables

This data contains several geographic variables to use. Each of these has a different number of possible categories or levels. A quick summary table is provided below with the variables that made it into the models shaded in blue.

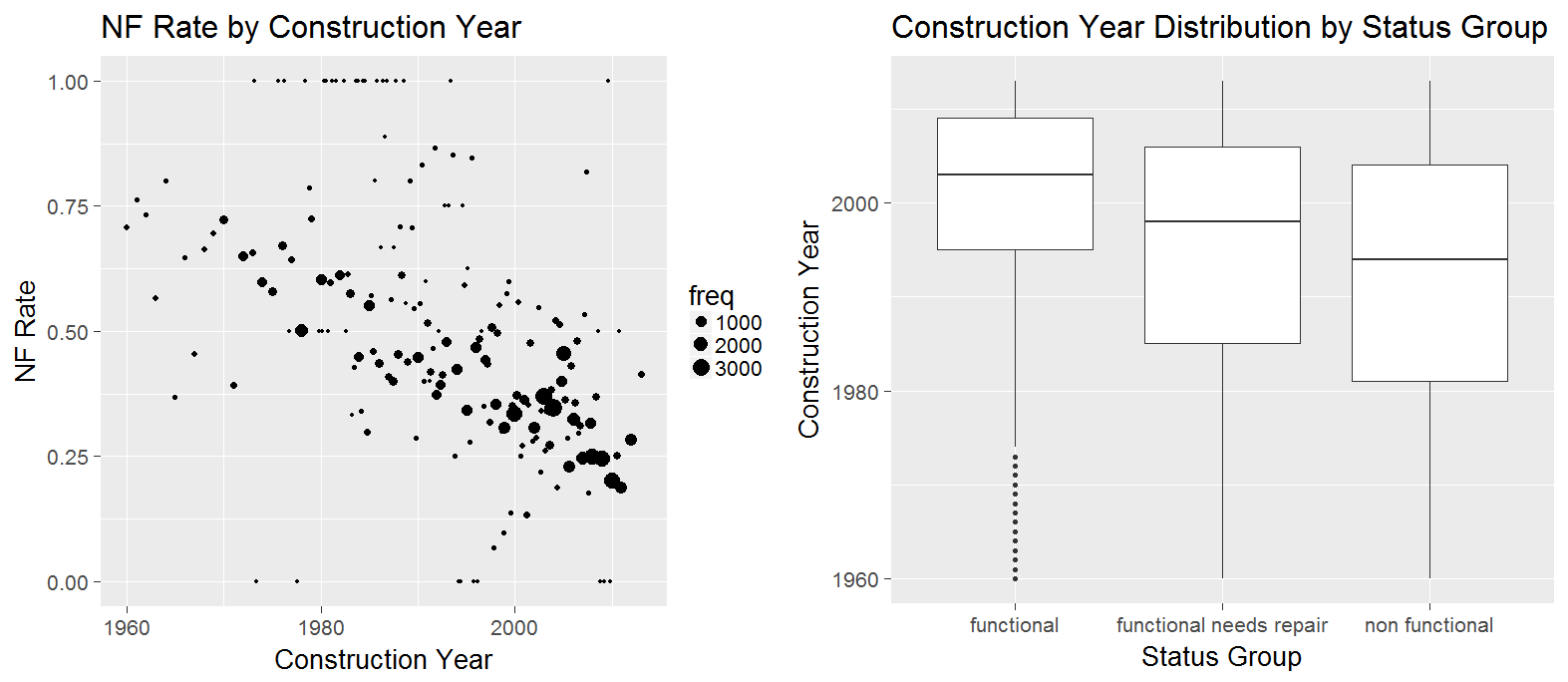
The coordinates are useful for mapping and they have relatively few missing or questionable data points, around 1,800 which I address in the subsequent section. I eliminated subvillage, ward, and lga from the analysis due to the large number of levels. There seemed little reason to try to reduce the variability given that there are already three other geographic grouping variables and the latitude and longitude coordinates. I eliminated region code as it appeared to be duplicative with region despite the fact that it has several additional levels.

Below is a map of the wells based on their coordinates for each of the three geographic groups. You can see that they appear to cover disparate areas and so could each be useful. If they covered similar areas, they might hurt my modelling efforts.



## Construction Year

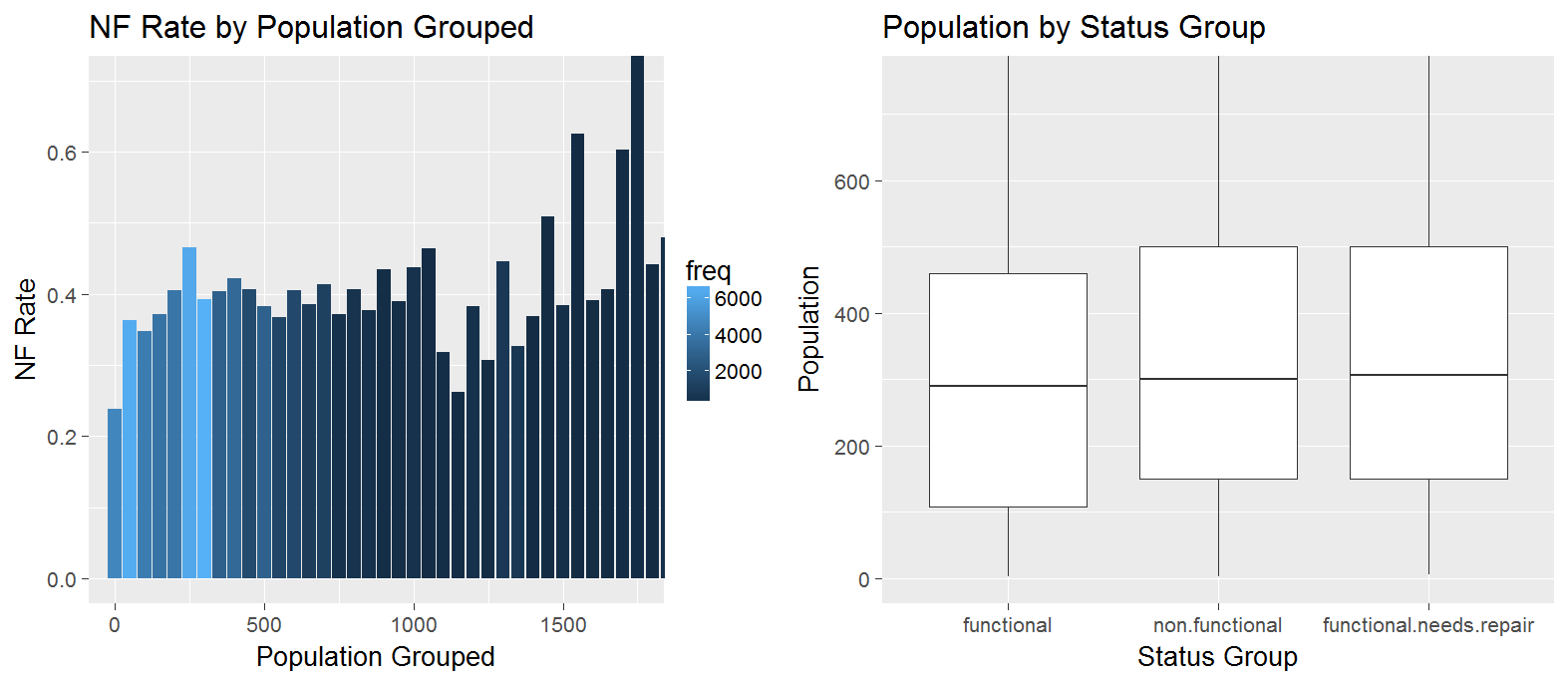
Construction year is one of the most useful features on an intuitive level – the older a well is, the greater the chance is that it will have a problem. That relationship is apparent in the data. I explored the data in a couple of ways. In the chart NF Rate by Construction Year, I looked at the relationship between the NF Rate and construction year. Apparent in that graph is a negative relationship. To illustrate another dimension, the frequency (freq) of wells in that year determines the size of the point. The chart shows that there are more wells built in the latter years.



Similarly, I created boxplots for each class. Generally speaking, this chart too supports the notion that functional wells were built in the later years. The median year for a functional well is above that of the wells that needs repair. Among the wells that need repair, the median year is above that of the wells that are non-functional.

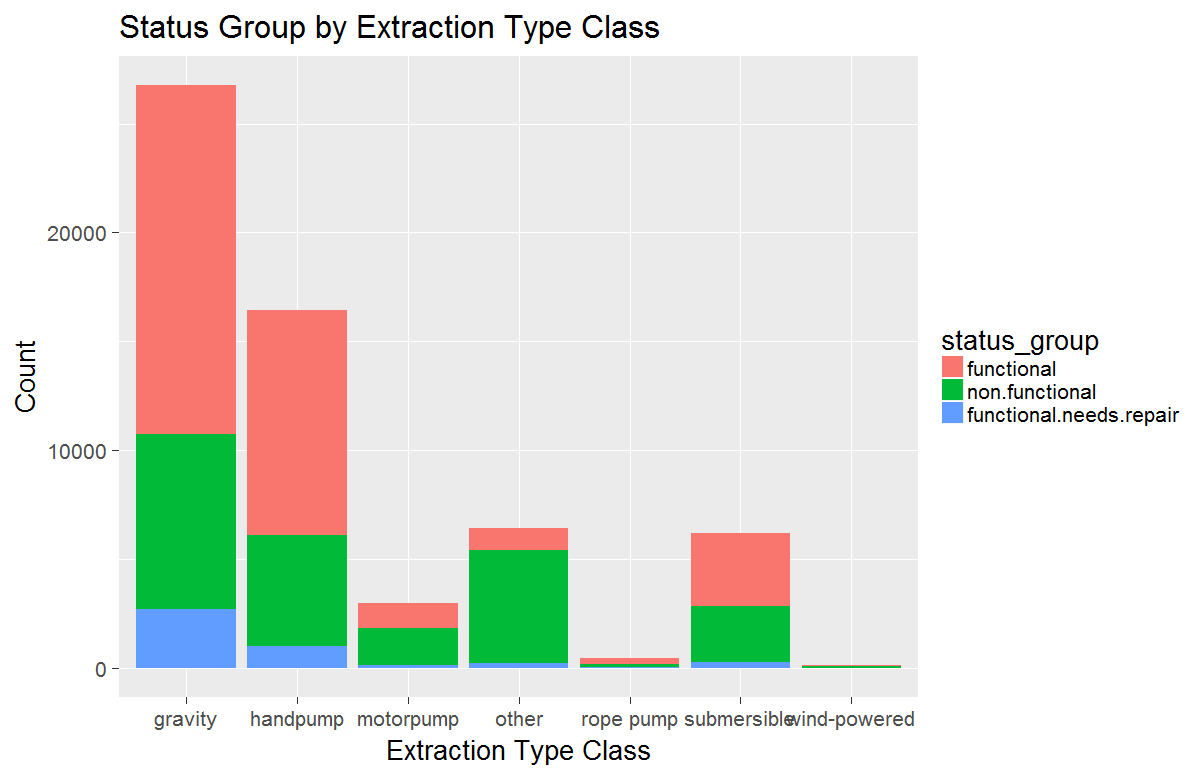
## Population

Population also has an intuitive influence on the functionality of wells – the more people that rely on the well or the more people that draw water from the well, the more likely a pump is to malfunction. This relationship is not quite as strong as it was with the construction year but I do see a positive relationship in the chart *NF Rate by Population Grouped.* Note that the population has been binned and the sizes of the bins are coded by color with darker colors signaling smaller bins. The larger bins are towards the left side of the graph (the smaller populations) and among those the positive relationship is clearer. I also show the boxplots by status group, as I did before. Here again, the boxplots generally support the relationship.



More than half of the values for population are “0” or “1” and while it is not inconceivable that an area would have just one resident, there were roughly 8,000 wells with a population of “1” and 4 wells with a population of “2”. I considered those values missing and address them in the subsequent section.

## Extraction Type

There are 3 different extraction type information: extraction type (18 levels), extraction type group (13 levels), and extraction type class (7 levels). The information in these fields is redundant. On a simple J48 decision tree, the accuracy between these factors is virtually identical and so I will use the extraction type class for simplicity. The chart to the left shows the number of wells for each extraction type class colored by the status group.

In this case the proportions are more obvious. The larger categories of “gravity” and “handpump” have similar ratios of functional wells but “motorpump” and “other” have a much lower percentage of functional wells. This could be a very useful data element for a model in that these groups seem to divide the non-functional wells. I considered combining some of the groups here to reduce dimensionality but ultimately decided to keep the additional information.

## Others

There were many other features in this data set. Many of the features were subsets of other features and for those, I looked for unique information (i.e., only one variable that is based on extraction type) with a limited number of levels (generally less than 20) and with the fewest number of very small groups (generally under 200 wells). A summary table is provided as *Appendix A. Feature List* with a detailed list of variables and a note as to why they were excluded.

# Data Transformations

Latitude, longitude, construction year and population all had missing or questionable data points. To deal with those, I used Caret’s pre-processing functionality which imputes the missing data points using the average value of the five nearest neighbors (kNN regression) in the case of numeric values which these were. Many of the categorical variables had blanks or missing data points. For those, I typically added a group called “other” or added the missing values to a smaller group currently identified in the data. With more time, I would have liked to explore and test these methods a little bit further. I could have used other imputation methods such as using means or medians, various levels of kNN or even excluding the missing values and compared the results of each. Ultimately, I chose the simplicity of the Caret method “knnImpute” to save time.

One element, date recorded, defaulted to a character string. I converted that to a date and calculated the number of weeks that have passed between that date and January 1st, 2017. This makes the variable continuous.

# Model Development

To classify the well type, I explored and evaluate four models: J48, kNN, random forest, adaBoostM1.

## k-Nearest Neighbors (kNN)

kNN identifies a set number of records, k, that are most similar to the row in question and selects the most frequently found observation in the case of classifier or the average value in the case of regression. Using the IBK algorithim in the RWeka package in R, I ran all levels of k from 1 to 20 and compared the model accuracy between each iteration. The best performing model had a k of six. This means that the model performs best when I considered the six nearest records to each record in the test set based on the euclidean distance.

## J48

The J48 model found in the RWeka package generates C4.5 decision trees. I experimented with several parameters and chose the model with the highest overall accuracy when using 10-fold cross validation.

The “C” parameter which controls the confidence factor impacts the pruning of the model. Lower “C” parameters result in less pruning. I experimented with three different levels here – 0.05, .25 (which is the default), and 0.5. I used three different levels for the “M” control which specifies the minimum instances per leaf. I used 1, 3 and 5 and 5 had the best result by a small margin. The final model reflected a “C” of .25 and an “M” of 5.

I also tried different combinations of the geographic variables. My concern initially was that this model had five different features that were based on geographic location and that maybe that was over emphasizing that aspect of the data. However, when I took out the different geographic characteristics one at a time, model accuracy suffered. The final model includes all of the geographic characteristics.

## AdaBoostM1

Also from the RWeka package, the AdaBoostM1 model works with other algorithms to emphasize incorrectly classified observations. I started with the adaBoostM1 with a decision stump. This resulted in no observations being classified as “functional needs repair”.[[2]](#footnote-3) I was able to change the underlying algorithm to “J48” and this model predicted all three classes. Oddly enough though, this model performed worse than the J48 model without boosting. The description of this model suggests that this model is subject to overfitting which might explain why it underperforms compared to the J48 model.

## Random Forest

The random forest model is an ensemble learner that uses variations on decision trees together to classify records. I used the “caret” and “randomForest” packages to create the random forest model. For this work, I have stuck with the default parameters, primarily due to the length of time that it takes to run the random forest model.

# Model Evaluation

To evaluate the models consistently, I split the data into a training set and a test set. 75% of the data went to the training set and 25% went to the test set. For each set, the data contained roughly equal proportions of each of the status groups, the class for this problem. Each final model was used to predict the values in the test set which allowed me to have an apples-to-apples comparison.

I considered four different metrics when evaluating the different models: Overall model accuracy, the true positive rate or sensitivity for the “non functional”(NF) class, the area under the curve (AUC) for the “non functional” class, and the Kappa statistic. The overall model accuracy measures how many of the predictions were correct, which is ultimately what I am trying to do. I also considered the true positive rate for the “non functional” wells because there is an emphasis on correctly identifying those wells. If, for example, I had two models with similar model accuracy but one had a higher true positive rate for the class that I am most interested in, I would likely want to choose the model with the higher true positive rate. The AUC is another way of assessing how well the model fits with test data, with a value from 0 to 1 where better-fitting models have higher values. Finally, I used the kappa value which is a comparison of observed accuracy to the expected accuracy. Here again, the higher the value, the better the model fits the data.

A summary table of the selected results for each model is provided below.

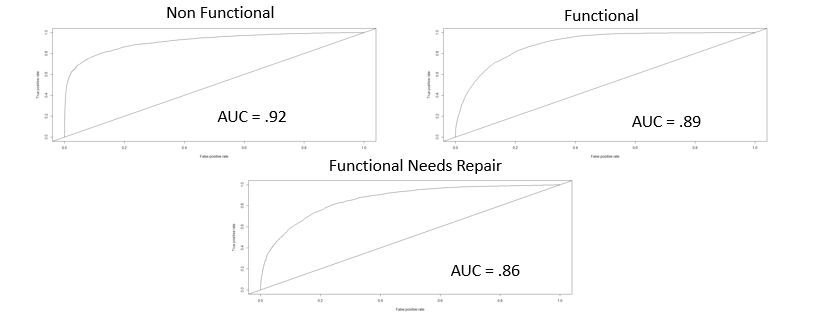
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | NF true positive | NF AUC | Kappa |
| J48 | .7809 | .7404 | .8677 | .5868 |
| kNN | .7646 | .6817 | .8769 | .5503 |
| Random Forest | .7988 | .7748 | .9170 | .6208 |
| AdaBoostM1 | .7654 | .7688 | .8920 | .5704 |

The random forest model outperformed the other models in each of these metrics. With the random forest model nearly 80% of the wells were accurately classified and of the wells in the test set that were non-functional, 77% were classified correctly. The area under the ROC curve (shown below) was 92% and the kappa statistic was .62. All of these metrics point to the random forest as the best of these four models and a fairly high functioning model in general.

Below is the confusion matrix showing the performance of the random forest on the test data.

|  |  |  |  |
| --- | --- | --- | --- |
|  | functional | Non functional | Functional needs repair |
| functional | 7,098 | 1,206 | 560 |
| Non functional | 736 | 4,421 | 176 |
| Functional needs repair | 230 | 79 | 343 |

The ROC graph shows the true positive rate on the y-axis and the false positive rate on the x-axis. The curved line represents the values for those metrics at different decision-point probabilities. The ROC curve is meant to compare true and false and so with mulit-class problems such as this one, I have taken a one-versus-all approach and I am showing a separate ROC curve and corresponding AUC for each class. Each curve evaluates the true positive rate and false positive rate for each class compared to the other two classes. [[3]](#footnote-4) Those curves are shown below.



# Conclusion

The best performing model was clearly the random forest. By all of the metrics evaluated above, the random forest outperformed its competitors but not by a huge margin. The random forest correctly identified 34 additional “non-functional” wells compared to the J48 decision tree. That is an increase of less than 1%. The better performance may be welcomed, however small. Certainly, if you are one who relies on one of those 34 wells, you are happy about the improvement but it is probably worth considering one additional metric – run time.

The J48 decision tree executes in about seven minutes while the random forest took over four and one half hours at last run. Some of the difference is the result of the package, I’m sure.[[4]](#footnote-5) That difference in time was pretty limiting. It was difficult to experiment with the random forest package when it takes that long to run. I would have liked to spend more time trying different variations of features and different categories within features. If I had to start again, I would choose a smaller data set. I found myself doing a lot of waiting for models to run when it would have been beneficial to be exploring the modelling a little bit more.

In the end, I was able to classify nearly 80% of the wells correctly which I would consider a success. I submitted the results of the model to the website where I am currently ranked 304 out of over 2,600 competitors. I feel pretty good about that ranking for my first competition. The one drawback with using the random forest model, though, is that I am not able to provide concrete rules regarding the functionality of wells. I could provide a list of wells that have the highest probability of being non-functional and with the EDA that I have done, I could provide some insights, but if what the Tanzanian water authority needed was hard rules about the classification, then another model, such as the J48 decision tree, would have been a better choice.

# Appendix A. Feature List

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Feature | Data Type | Number of levels | Notes | Used in the Model? | Exclusiong Reason | Imputation Method |
| longitude |  | N/A |  | Yes |  | kNN |
| latitude |  | N/A |  | Yes |  | kNN |
| region | categorical | 21 |  | Yes |  |  |
| district\_code | numeric | 20 |  | Yes |  |  |
| population | numeric | N/A |  | Yes |  | kNN |
| construction\_year | numeric |  |  | Yes |  | kNN |
| extraction\_type\_class | categorical | 7 | Tried reducing number of categories | Yes |  |  |
| basin | categorical | 9 |  | Yes |  |  |
| gps\_height | numeric | N/A |  | Yes |  | Left in 0's |
| public\_meeting | categorical | 3 |  | Yes |  | NA's as "Unknown" |
| amount\_tsh | numeric | N/A | A lot of 0's, no real distinguishable difference in groups | Yes |  | Left in 0's |
| date\_recorded | date | N/A |  | Yes |  | Transformed to Weeks from Current date |
| scheme\_management | categorical | 13 |  | Yes |  |  |
| permit | categorical | 3 |  | Yes |  |  |
| management | categorical | 7 |  | No | Repeat of Scheme Management | |
| payment | categorical | 7 |  | Yes |  |  |
| quality\_group | categorical | 6 |  | Yes |  |  |
| quantity | categorical | 5 |  | Yes |  |  |
| source\_type | categorical | 7 |  | Yes |  |  |
| waterpoint\_type\_group | categorical | 6 |  | Yes |  |  |
| waterpoint\_type | categorical | 7 |  | No | Redundant with Waterpoint type group |  |
| region\_code | numeric | 27 |  | No | Primarily a repeat of region |  |
| lga | categorical | 125 |  | No | Too many categories |  |
| ward | categorical | 2092 |  | No | Too many categories |  |
| subvillage | categorical | 19288 |  | No | Too many categories |  |
| id | character | N/A |  | No | Not Useful |  |
| extraction\_type | categorical | 18 |  | No | Redundant with extraction type class |  |
| extraction\_type\_group | categorical | 13 |  | No | Redundant with extraction type class |  |
| wpt\_name | categorical |  |  | No | No Information Provided |  |
| recorded\_by | categorical | 1 |  | No | Not useful |  |
| funder | categorical | 1897 |  | NO | Too many categories - no clear way to group them |  |
| installer | categorical | 2146 |  | No | Too many categories - no clear way to group them |  |
| num\_private | numeric |  |  | No | Primarily NA |  |
| scheme\_name | categorical | 2697 |  | No | Too Many Categories |  |
| management\_group | categorical | 5 |  | No | Redundant with management |  |
| payment\_type | categorical | 7 |  | No | Redundant with payment |  |
| water\_quality | categorical | 8 |  | No | Redundant with quality group and worse groupings |  |
| quantity\_group | categorical | 5 |  | No | Redundant with quantity |  |
| source | categorical | 10 |  | No | Redundant with source type |  |
| source\_class | categorical | 3 |  | No | Redundant with source type |  |

1. <https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/>. Accessed Jan. 10, 2016. [↑](#footnote-ref-2)
2. I must confess that I don’t really know why but my guess is that it doesn’t work with multi-class problems. [↑](#footnote-ref-3)
3. While Weka handles multi-class ROC curves well, showing a one-vs-all graph of 3 lines, I could not get the same functionality out of R and I could not get Weka to complete a random forest on this large a data set. [↑](#footnote-ref-4)
4. Running the J48 through the Caret package takes much longer than running it straight through the RWeka package. I am not sure what causes the differences in run time. [↑](#footnote-ref-5)