MINING THE WATER TABLE

Renel Chesak

CSC 529

FALL 2017

Renel Chesak
CSC 529
Fall 2017

Case Study 1: Mining the Water Table

Introduction

Can we predict which water pumps are likely to be functional, functional but in need of repair, or nonfunctional? That is the question for this challenge hailing from Tanzania. A classifier that can do this with high accuracy can allow for prioritization of costly inspections as well as preemptive maintenance of life-giving water pumps, thus preventing or shortening periods of water-outages. After cleaning, the dataset for this challenge includes 21 relevant predictors related to the location, water source, water quality and quantity, government regulators, funders, population served, management authorities, specs of the pumps (e.g. total static head), altitude, payment scheme, and more. Given that the dataset includes a healthy mix nominal, continuous, and ordinal data, a decision tree should work well. However, in the likely case that the accuracy can be improved by an ensemble, we can also modify the dataset to meet the requirements and assumptions of K-Nearest Neighbor, Gaussian Naïve Bayes, and Bernoulli Naïve Bayes. Given that these three approaches output a probability of an instance falling into a particular class, they adequately fulfill our need to classify water pumps.

Decision Trees have no real requirements other than balanced target classes, so the cleaned data can be input with little-to-no problem. However, K-Nearest Neighbor and Naïve Bayes both require categorical input to be binary, so we will need to create dummy variables and keep dimensionality in mind. Given that we have a dataset with 59,400 records, we should have a good deal of flexibility to include features. The Gaussian implementation of Naïve Bayes allows us to include continuous data in our model, following the assumption of normality and the subsequent probabilistic nature of normal distributions. Bernoulli NB is designed to work with binary predictors, and thus will work well with the many dummy variables that will be created. Keeping these requirements in mind, these three classifiers should work well with our dataset and allow us to create an ensemble that performs better than any of them alone.

Dataset Description

This dataset comes from Driven Data, a competition site targeted at social good projects (https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/). The raw data contains 39 features and 59,400 observations. The following feature descriptions come directly from Driven Data:

- 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

As mentioned in the introduction and justified in the data cleaning section of this report, some of the most relevant features include location, water source, water quality and quantity, government regulators, funders, population served, management authorities, specs of the pumps (e.g. total static head), altitude, and payment scheme. Together these features represent important aspects of the water pumps that relate to the functionality. For example, the water source could be corrosive, apply too much pressure on the pipes, clog the pipes with debris, etc. The location could be indicative of local irregularities or patterns in usage, soil type and stability, and/or temperature (thawing and freezing). Government regulators, funders, population served, and management authorities all represent the complex ecosystem of people who work together to make sure the pumps are built well, maintained, and used properly. The total static head (static pressure in a pipe) could affect how often pipes burst or valves wear out. Altitude could indirectly relate to temperature (which can affect pump functionality), and/or how likely it is for maintenance crews to reach the pump (e.g. being high in the mountains could be a barrier to travel). Payment Scheme could represent how much incentive there is for pump owners to make sure pumps are functional.

Though our dataset is rich, it does have limitations. Several variables that could be important contain null values for a large portion of instances, which limits their usefulness. Many of the variables that could be important have a high cardinality. Given that KNN and Naive Bayes require binary dummy variables to be made, high cardinality translates to high dimensionality. We know from the concept of "The Curse of Dimensionality" that as dimensionality increases, the ratio between the nearest and farthest points approaches 1, meaning that points become indistinguishable from each other, which is a problem for KNN in particular. It also increases the probability that correlations found are due to chance alone. High dimensionality also tends to lead to overfitting, and can make decision tree building a very lengthy and computationally-heavy process. For all of these reasons, features will be scrutinized for high cardinality.

Another limitation is that we are relying on several of these features to help us indirectly measure more pertinent predictors that we do not have in the dataset. One example is measurements related to soil temperature, wetness, and makeup, which would indicate the stability of the medium containing the water pump. Other examples include time since the last maintenance inspection, and more specifications of the pump such as materials, designs, and durability. It would be wonderful to have such data, but as is most always the case, we will try to tease out those latent variables given the data we do have.

Data Cleaning

This dataset began with 39 features, many of which were redundant, irrelevant, or had far too high cardinality to be useful.

`ward` contains 2,092 unique values, which justifies throwing it out on the basis of high cardinality. `scheme_name` has 2,696 unique values, and thus has far too high cardinality to be kept.

Several variables came in hierarchical sets, or where 100% redundant with another variable. Those removed due to redundancy include `extraction_type`, `extraction_type_group`, `payment_type`, `water_quality`, `quantity_group`, `source_type`, `source_class`, `waterpoint_type_group`, `region_code`, `district_code`.

`subvillage` has 19,287 unique values and thus is likely too granular to be predictive; it is almost a unique identifier, and the cardinality is also far too high. `date_recorded` is arbitrary, however `construction_year` could be very predictive, thus we will drop the former. `wpt_name` is a unique identifier, and it can be dropped as well. `num_private` is not described in the dataset description, and it is 0 in an overwhelming majority (98.7%) of cases, so we will throw it out as well. `recorded_by` has the same value for all instances, and thus cannot be predictive and can be thrown out. `scheme_name` has 2,696 unique values, and thus has far too high cardinality to be kept.

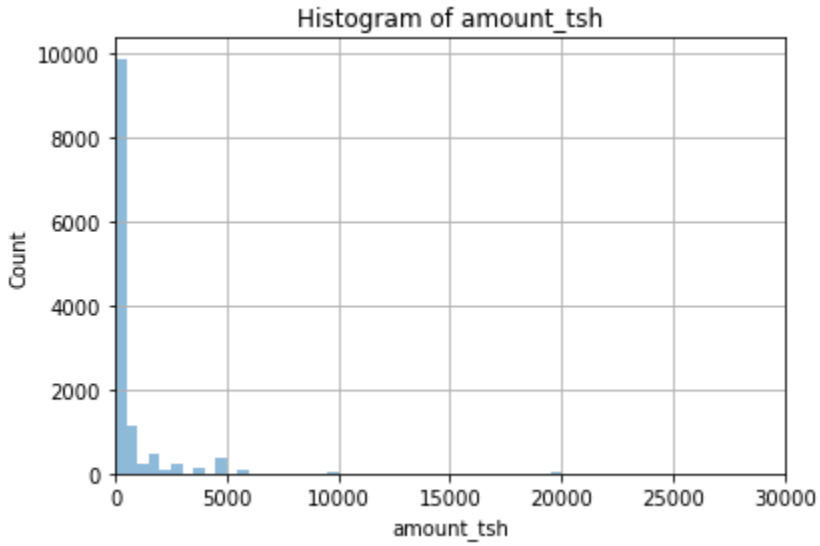
`installer` has 2,145 unique values, which means it's cardinality is out of the question for KNN and Naive Bayes. The same is true for `funder` (1,897 unique values). We can leave them in for the decision tree, as long as the tree can be built in a reasonable amount of time.

21,381 records have a `population` of 0, and 7,025 have a `population` of 1. These may or may not have meaning, but since we do not know, we will assume they are NA and code them as such. 20,709 records have 0 as `construction_year`, so we will code those as NaN. It is curious that the majority of instances (41,639) have an `amount_tsh` (amount water available to waterpoint) of 0. Looking at the raw data, we can see that many of these instances do serve a nonzero population, so it seems most likely that a value of 0 corresponds to unknown. For this reason we will code it as NaN. The most common `longitude` and `longitude` pair is (0.0, -2.000000e-08) which is essentially (0,0). This is a point over the ocean and thus it equates to unknown, so we will code those values as such. A large portion (20,438) of records have a `gps_height` of 0. This is curious, but since it falls within the realm of possibility, we will leave them as-is.

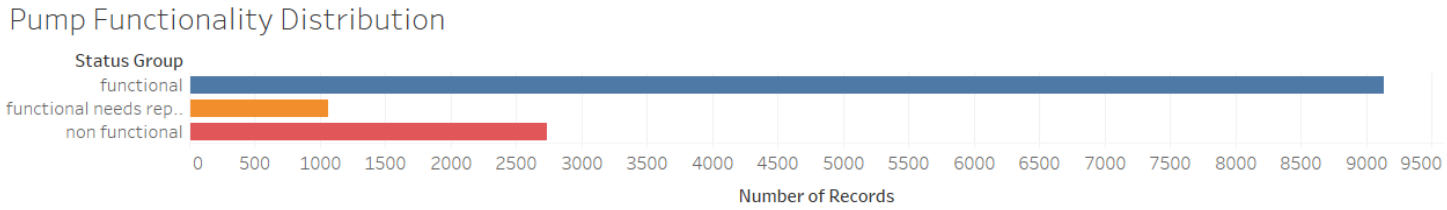
`basin` may be redundant with `region`, given that they are both geographic features, but it is possible that their geographic ranges are not the same and that one is more predictive than the other, so we will keep both for now. `lga` has 125 unique values which is costly in the sense of cardinality, however it refers to the Local Government Authority of the area and thus could represent fiscal inequities which may relate to the functionality of the local water pumps. For this reason, we keep this feature for now.

After culling the features thus far, rows containing NULL values where thrown out to yield approximately 13K observations, which is plenty for our investigation and is probably nearing the maximum dataset size we can run through an algorithm on a single machine with 8GB of RAM. Dummy variables were created from the categorical variables as Naïve Bayes (NB) and KNN both require this, and it doesn't matter for Decision Trees (DT). Creation of dummy variables resulted in 1,276 predictors, which will be culled again during feature selection for each model. The data was then split into a training, validation, and test set (60:20:20 split). Given that we have binary dummy variables, the data was normalized using min-max normalization so that all values will fall into the range between 0 and 1 and thus no feature will carry more weight than another.

Exploratory Data Analysis

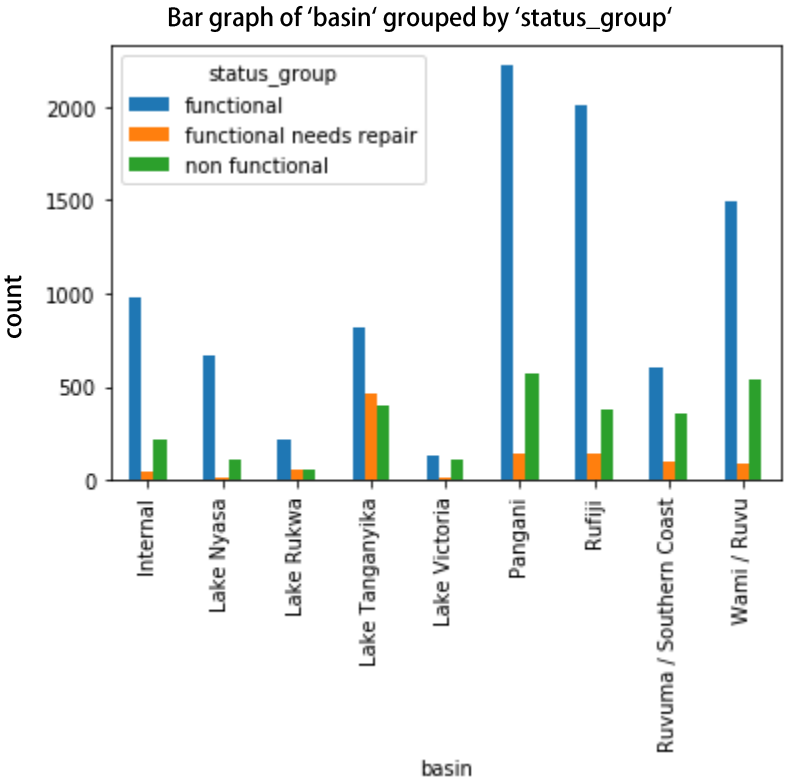


The distributions of numeric variables were all checked for normality, as this is a requirement of Gaussian NB. An example is displayed on the left, and the rest can be found in the appended ipynotebook. None of the numeric features in our dataset are normal, but we will incorporate DTs and KNN into our ensemble to support weakness of NB. This is however a potential area for improvement, if the distributions can be transformed to better fit a normal distribution.



Sum of Number of Records for each Status Group. Color shows details about Status Group.

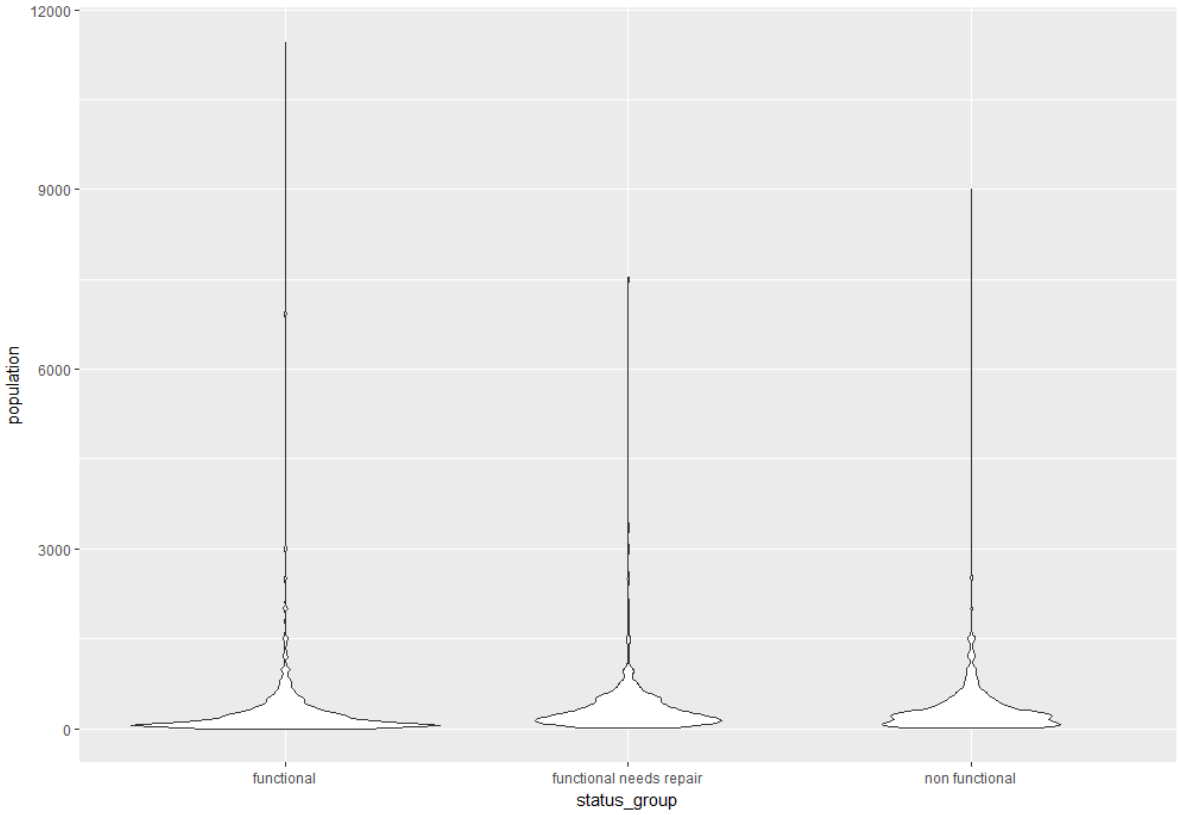
We see above that there is an uneven distribution of pump functionality, which could cause problems for our classifiers, so stratified sampling will be done with cross validation.



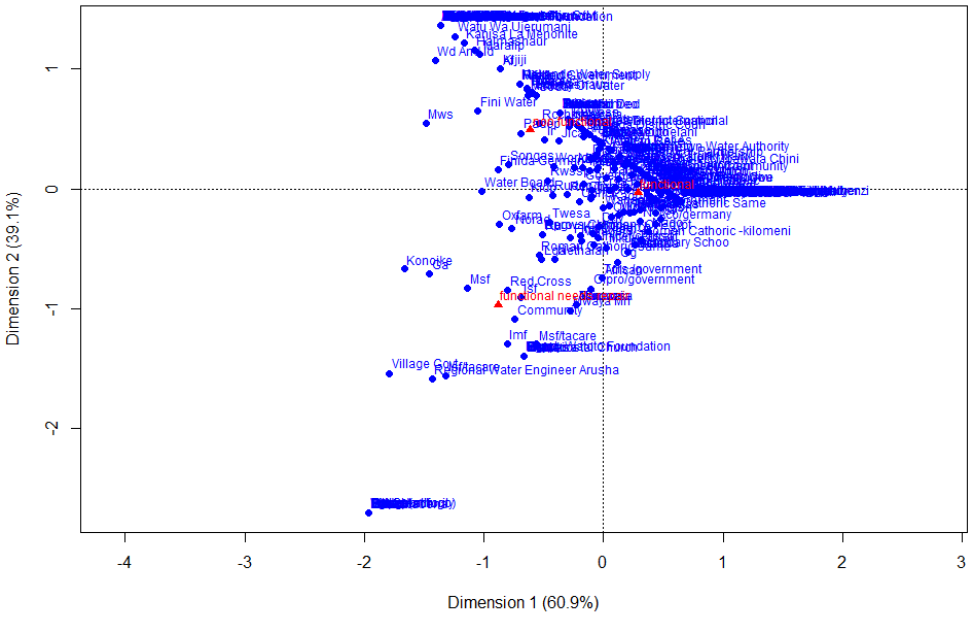
We can see on the left that pumps in the `Lake Tanganyika` basin have a higher likelihood of being in need of repair or nonfunctional. Please note that all visualizations are included in the ipynotebook appended to this PDF. From analysis of several similar bar charts, we can see several relationships and possible strong predictors. It looks like a local government authority of `Kigoma Rural` could be predictive of pumps needing repair. `public_meeting` does not look very predictive of pump functionality, and will likely get thrown out during feature selection. Not surprisingly, it looks like newer pumps tend to have a higher proportion of functionality than older pumps. Water coming from rivers and springs result in the most functional pumps, however water coming from rivers has a disproportionate number of pumps in need of repair. Water coming from lakes seems to be the most indicative of non-functionality, which makes sense because we might expect that water to contain more debris than springs and rivers. Pumps whose quantity is `enough` are by far the most likely to be functional. Interestingly, pumps where people pay only when the scheme fails or pay per bucket have the lowest proportions of pumps that are functional but need repair. This may imply that when people have direct control over whether the pumps get fixed, they are more likely to get fixed.

We can see below, in the violin plot, that population does not appear to be related to pump functionality. We might expect that it would be, as it makes sense that large populations would have more stable access to water, thus this result is a surprise.

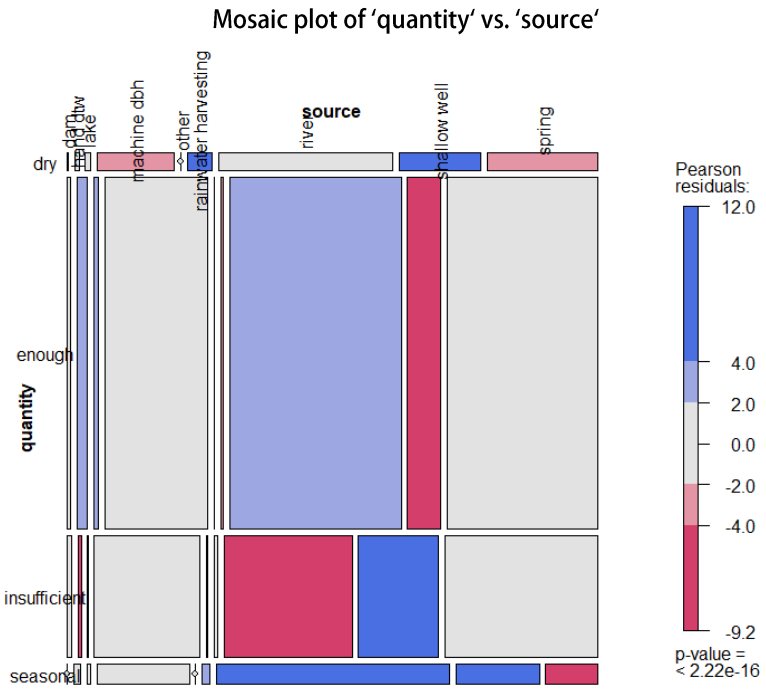
'population' distribution broken down by 'status_group'



CA plot of 'funder' vs. 'status_group'

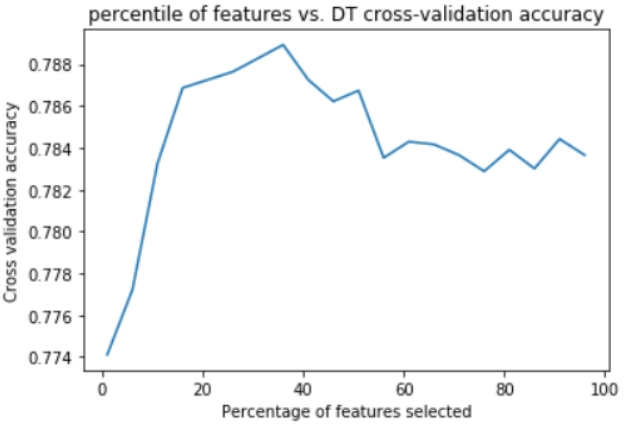


Correspondence Analysis (CA) is a technique that computes dimensions of highest variance, similarly to PCA, but for categorical values of variables. From the CA plots of `funder` and the target variable (seen on the left), as well as `installer` and the target variable, it is clear there are some relationships that could be predictive. The CA plots can be read by drawing a line between a red point and the origin, drawing a perpendicular line from it to a blue point, and measuring the distance along the first line from the perpendicular intersection to the red point. Smaller distances imply stronger, positive relationships, and further distances opposite the origin imply stronger negative relationships. The patterns seen here justify leaving `funder` and `installer` in the feature set, despite their high cardinality.



Mosaic plots are also built on the concept of Correspondence Analysis. The mosaic plot on the left shows us that several values of `quantity` and `source` occur together more often than we would expect if the features where completely independent. For instance, we can see that shallow wells are significantly likely to have an insufficient water quantity, which makes sense. Rivers are significantly likely to not have an insufficient water source, which also makes sense. From a similar mosaic plot found in the ipynotebook, we can see that `management_group` and `payment` are also clearly not independent, as we can see that there is a strong positive relationship between commercial management and the pay-per-bucket scheme, and a strong negative relationship between user-group management and the pay-per-bucket scheme. Though these features will fail the assumption of independence that we find in Naïve Bayes, we expect that our ensemble, which will include a Decision Tree, can make up for this failing, since the dependencies are inherently built into decision trees.

From our exploratory data analysis, we can conclude that our dataset is well-suited to our goal of predicting water pump functionality, since there appear to be several strong predictors in the dataset. We have discovered that we have a good mix of numeric and categorical data, some of which fit the assumptions of our models, and others that don't, which supports the use of an ensemble model with varying assumptions and requirements of the data.



Experimental Results

Feature selection was done for DT, NB, and KNN algorithms via iterating over different percentiles of the training features. For each percentile, Chi2 testing is used to pick the best features for that percentile; 10 Stratified 10-Fold Cross Validation models are built using the given algorithm; all percentile's 10 CV accuracy scores are compared for significant differences using an F-test; the winning percentile is chosen based on having the highest accuracy, and its 10 CV scores are used to compute a 95% C.I. for a final accuracy report. Finally, we graph the average accuracy scores found for each percentile (seen on the left). We use the Chi2 test because we are looking for relationships between predictors which are majority non-negative categorical and a non-negative categorical target feature. Also note that the 10-Fold CV being done is using stratified sampling of the target class, which is important because we have an uneven distribution of classes. A summary of the results is in the table on the left, and the full details can be found in the ipynotebook at the end of this report.

	Gauss_NB	Bern_NB	DT	KNN
features	14	12	102	89
F-value	66.75	13.66	2.4	4.81
p-value	5.43E-71	8.04E-26	0.0064	7.30E-05
Accuracy 95% C.I.	71.36%, 73.49%	72.82%, 74.18%	77.58%, 78.73%	78.35%, 79.64%

Both Gaussian and Bernoulli Naïve Bayes were performed in the hopes that they could exploit different features of the data, since Gaussian NB is designed to work with normally-distributed continuous predictors, and Bernoulli NB is designed to work with binary predictors.

Parameter tuning was done for DT and KNN algorithms via iterating over different ranges of possible parameters on the validation data sets, which were then used in 10-Fold Stratified CV to produce accuracy scores. The ANOVA F-value was computed over all sets of CV scores to test for significant differences between parameter choices. The optimal schema was chosen based upon the highest accuracy, and the 10 CV accuracy scores were used to compute the 95% C.I. for a final accuracy report. The parameters for NB are the class prior probabilities, and since we do not know them, we will allow the algorithm to learn them from the data, and thus have no manual parameter tuning to perform for NB.

For DTs, we iterated over a range of maximum tree depths, and again over a range of minimum splitting values to create a classifier object. The final results for the DT were as follows:

```
results, cv_scores = find_DT_parameters(DT_minmax_validation, class_validation, kfolds=10, max_depth_range = range(1, 8, 1),
                                        min_samples_split_range = range(5, 35, 5))
```

```
Optimal schema:
    max_depth  min_samples_split  average CV accuracy
38        7.0               15.0             0.768946


--------------------------------------------------------------------------------
ANOVA F-value for the all the schema's CV scores:
F_onewayResult(statistic=5.1182059627213263, pvalue=1.6636700964892584e-18)
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
95% Confidence Interval for the highest accuracy score:
(0.7594043209586403, 0.77848798234222327)
--------------------------------------------------------------------------------
```

For KNN, we iterated over a range of n_neighbors and both uniform and weighted distance measures to create a classifier object. The Jaccard distance is a measure of dissimilarity that is well suited for binary variables, and since the vast majority of the features are binary dummy variables, it seems a better choice than a distance metric only suited for continuous data and is used for all iterations. The final results for the KNN were as follows:

```
results, cv_scores = find_KNN_parameters(KNN_minmax_validation, class_validation, kfolds=10, n_neighbors_range=range(10, 30, 1))
```

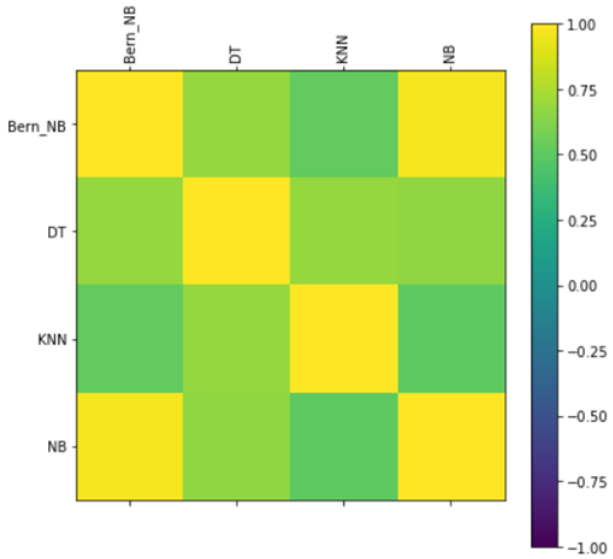
```
Optimal schema:
  n_neighbors   weights  average CV accuracy
8          14  distance             0.767453


--------------------------------------------------------------------------------
ANOVA F-value for the all the schema's CV scores:
F_onewayResult(statistic=2.0738008255995908, pvalue=0.00030259823681448486)
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
95% Confidence Interval for the highest accuracy score:
(0.75208796687740542, 0.7828172761601333)
--------------------------------------------------------------------------------
```

The final models where built on the training data, and the correlations between their predictions on the training data are shown in the correlation plot on the right. It turns out that the Bernoulli NB correlates very highly with the Gaussian NB, which makes sense because the large majority of features that went into each model where binary, so the algorithms operated largely the same (with the exception of a few numeric variables). There is moderate correlation between the other models (between 50% to 68%), which demonstrates a good deal of independence between the models and a larger capacity for synergy in an ensemble.

Correlation of Model Predictions



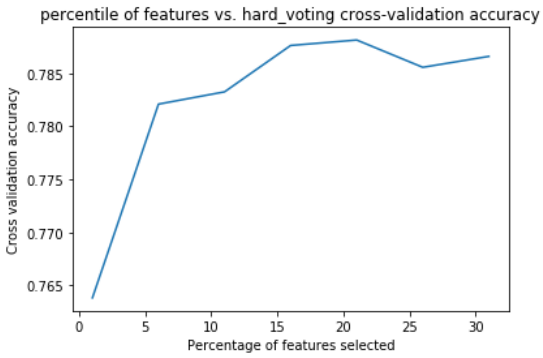
Ensembles were built using a few different methods: Stacking with Hard Voting, Stacking with Soft Voting, and XGBoost. The first two methods required that all base models be trained on the same feature set, so we implemented the same iterative process for feature selection, using the ensemble classifier object trained on the training set. The results are as follows:

Stacking with Hard Voting

```
Optimal percentile of features:[21]
Optimal number of features:[ 267.96]
Optimal schema:
    classifier percentile  average CV accuracy
4  hard_voting         21             0.788155

---------------------------------------------------------------------------
ANOVA F-value for the all the percentile's CV scores:
F_onewayResult(statistic=6.1428705405640063, pvalue=4.1162135183126686e-05)
---------------------------------------------------------------------------

---------------------------------------------------------------------------
95% Confidence Interval for the highest accuracy score:
(0.78038542924914123, 0.79592518831887582)
---------------------------------------------------------------------------
```

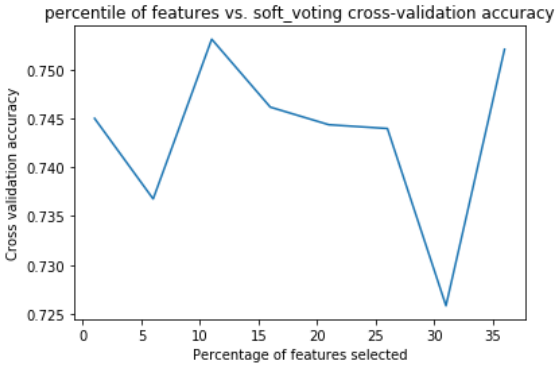


Stacking with Soft Voting

```
Optimal percentile of features:[11]
Optimal number of features:[ 140.36]
Optimal schema:
    classifier percentile  average CV accuracy
2  soft_voting         11             0.753128

---------------------------------------------------------------------------
ANOVA F-value for the all the percentile's CV scores:
F_onewayResult(statistic=9.4066343853363215, pvalue=3.1360520817609959e-08)
---------------------------------------------------------------------------

---------------------------------------------------------------------------
95% Confidence Interval for the highest accuracy score:
(0.74612084998431794, 0.76013472784159575)
---------------------------------------------------------------------------
```



The XGBoost implementation allows for the base models to be trained on different feature sets, so those chosen during feature selection were used to train the base models on the training set. Then, the predictions were used as vector representations of each base model, and were used to build an ensemble XGBoost classifier. This classifier treats each base model's predictions as a feature, and trains gradient boosted decision trees to create an ensemble that performs better than any of the base models on their own. The algorithm is boosting in the sense that it uses the errors of previous models to create iteratively better models, but on top of that it uses a gradient descent algorithm to minimize prediction error overall. Since there are many parameters to set on a gradient boosted decision tree, we will do

parameter optimization in the same way as I did with DTs and KNN. The parameter ranges are as follows: **n_estimators_range**=range(50, 100, 5) (number of boosted trees to fit), **max_depth_range**=range(4,15,1) (maximum depth of the trees), **min_child_weight_range**=range(1,5,1)(minimum number of child instances required for a split), **gamma_range**=np.arange(.5,2,.5)(minimum loss reduction required to make a split). As one can guess, the ranges of these parameters required about an hour to run. From the results we saw that the optimal accuracy was 84.73% and this was achieved with many different combinations of parameters. We can also see that the F-test failed (F=5.7E-23, p=1.0), and thus none of the models were significantly different. Thus, we chose the model with the least complexity:

```
Optimal schema:
   n_estimators  max_depth  min_child_weight  gamma  average CV accuracy
0          10.0        4.0               4.0    0.5             0.847258


--------------------------------------------------------------------------------
95% Confidence Interval for the highest accuracy score + least complex model:
(0.83846468098201077, 0.85605130038964072)
--------------------------------------------------------------------------------
```

Finally, a Random Forest Decision Tree Ensemble was performed. Using the same iterative process as above, we iterated over several parameters for optimization, ranges including the following: n_estimators_range=range(10, 25, 5), max_depth_range =range(4,15,1), min_samples_split_range=range(50,100,5). The results were as follows:

```
Optimal schema:
    n_estimators  max_depth  min_samples_split  average CV accuracy
80          10.0       12.0               50.0             0.791374


--------------------------------------------------------------------------------
ANOVA F-value for the all the schema's CV scores:
F_onewayResult(statistic=13.344183341271734, pvalue=0.0)
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
95% Confidence Interval for the highest accuracy score + least complex model:
(0.78511475670022546, 0.79763352091369888)
--------------------------------------------------------------------------------
```

A summary of all ensemble final training results is below:

	Stacking, Hard	**XGBoost**	**RandomForest**
features	267	per base model	102
F-value	6.14	5.70E-23	13.34
p-value	4.12E-05	1.0	0.000
Accuracy 95% C.I.	78.04%, 79.60%	83.85%, 85.61%	78.51%, 79.76%

The Stratified 10-Fold CV accuracy scores for each winning model were used in pairwise t-tests as well as an F-test to look for significant differences in the means of the accuracy scores. The results are as follows:

```
XGBoost vs. Stacked Ensemble with Hard Voting:
Ttest_indResult(statistic=11.3687036094459, pvalue=1.1984851708036782e-09)

Random Forest vs. Stacked Ensemble with Hard Voting:

Ttest_indResult(statistic=0.70057562450302047, pvalue=0.49252519881466461)
```

```
Random Forest vs. XGBoost:

Ttest_indResult(statistic=-11.712298126758002, pvalue=7.451130249869516e-10)

ANOVA F-value for the all the schema's CV scores:
F_onewayResult(statistic=95.622956447326359, pvalue=5.5929148104928943e-13)
```

From these results, we have a clear winner: XGBoost with [83.85%, 85.61%] accuracy.

The ensembles were built on the training + validation sets, and the accuracy scores on the test set are summarized in the table below:

	Stacking, Hard	**XGBoost**	**RandomForest**
Test Accuracy	80.19%	79.37%	80.30%

Experimental Analysis

Though we cannot be sure due to a lack of ability to t-test, it appears that all three final models have similar performance on the test set. Given the accuracy scores on the training set, it appears that XGBoost is overfitting the most ([83.85%, 85.61%] vs. 79.37%, which makes sense given that it iteratively does gradient descent to minimize error. Thus, in terms of accuracy, I would say that all three ensembles perform relatively equally. Accuracy is the best measure for this problem set because it is equally important to correctly predict each class so that maintenance and other resources can be allocated efficiently and effectively.

Since accuracy is not a deciding factor, we can look at other aspects of the ensembles. The following table summarizes some important model factors:

	XGBoost	**RandomForest**	**Stacking, Hard**
accuracy	excellent	excellent	excellent
time to build	Long; each base model must be built with optimized parameters and feature sets, and their output transformed to input to train the gradient boosted trees. The process of training a gradient boosted tree can be computationally expensive because it is performing gradient descent on multiple decision trees to create an ensemble of trees.	Fast; decision trees are made by picking features at random (not computing GINI values).	Long; each base model must be built with optimized parameters and feature sets, and their output using in simple voting.
time to query	Long; predictions must be made for each base model and their output transformed to input for the gradient boosted trees.	Fast; you only need to go along the branches of the aggregated tree, answering binary questions.	Moderate; predictions must be made for each base model and their output used in simple voting. While this process could be long, it is automated and can be easily parallelized by having predictions made for each model on its own node.
scalability	Moderate; you need to train and predict with several models of varying complexity, in order to scale you would almost certainly need parallel computing. However, parallelizing is easy because you can have each base model run on its own node and train different gradient boosted trees on randomized subsets of the predictions on different nodes to produce a single aggregated tree.	Excellent; not only is the time to build and query fast, but it is easily parallelized by building random trees on randomized subsets of the data on different nodes and aggregating the results into a final tree.	Moderate; you need to train and predict with several models of varying complexity, in order to scale you would almost certainly need parallel computing. However, parallelizing is easy because you can have each base model run on its own node, and then compute simple voting.
scrutability	Moderate; you have to explain each base model, which would be lengthy and potentially inscrutable to executives. Explaining the gradient boosted trees would not be so hard as long as you only mention that it uses a decision tree to know which model to trust most for each class.	Moderate; the process is difficult to explain and visualize, however you can print out a bar graph of the most influential features, which helps to explain a system to executives.	Moderate; you have to explain each base model, which would be lengthy and potentially inscrutable to executives. Explaining the simple voting process is very simple and easy for anyone to understand.
interpretability	Good; you are given a final class prediction that is actionable. However, it could be improved if you could get feature importances as well as prediction probabilities.	Excellent; you are given a final class prediction that is actionable and can even retrieve prediction probabilities.	Good; you are given a final class prediction that is actionable. However, it could be improved if you could get feature importances as well as prediction probabilities.
updatability	Difficult; you need to train several models of varying complexity and transform their output as input to train computationally-heavy gradient boosted trees.	Excellent; time to train is fast and can be easily parallelized.	Good; you need to train several models of varying complexity, however the process is easily parallelized.

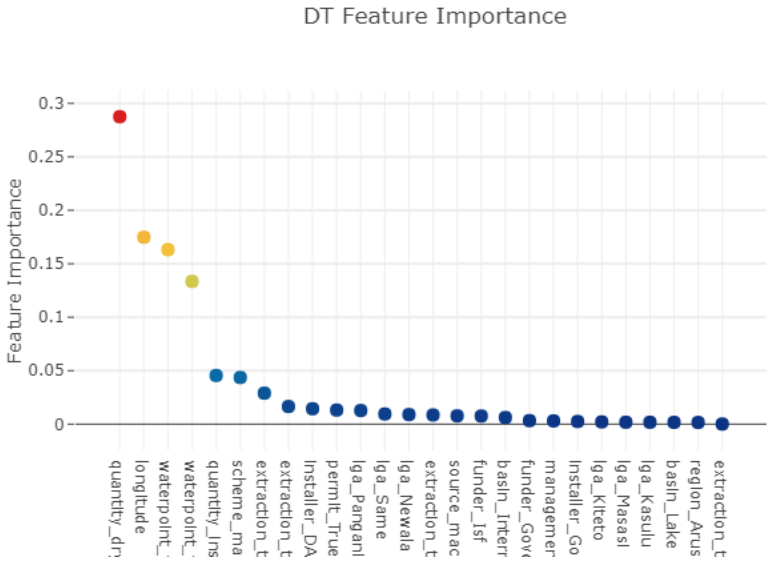
Given the reasons summarized in the table above, Random Forest seems to be the best choice since it has fantastic results in all areas of model evaluation.

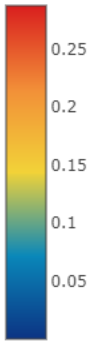
It is interesting that all three ensembles performed similarly on the test set. This makes the case that the data is somewhat trivial in that it can be used in multiple models with widely varying abilities to exploit particular features, as well as varying model assumptions about the data, and still predict with relatively the same level of accuracy. Perhaps this is due to the large number of features, which provide ample opportunity for each model to find and exploit the features that it works best with. The luxury of using large feature sets was made possible by the large number of

observations; perhaps this same performance could not be achieved with fewer observations and the subsequent pruning of features.

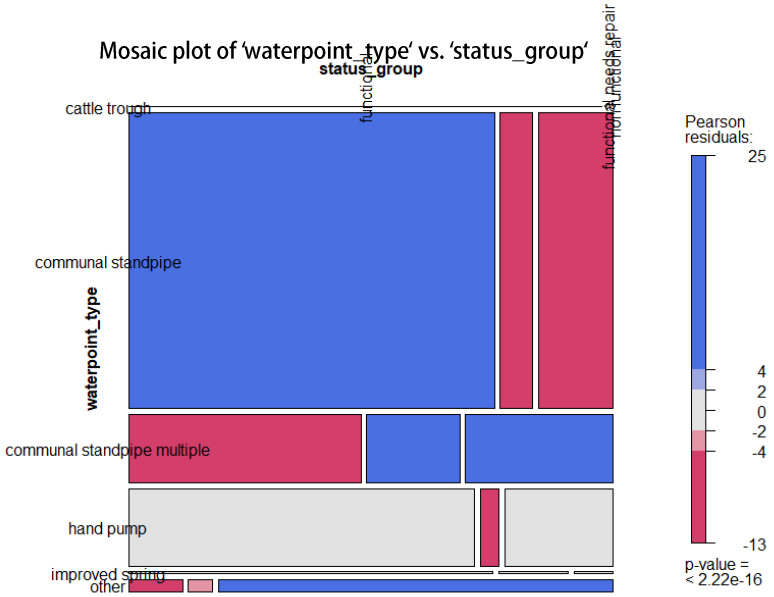
Since DTs are the only classifier that output meaningful feature importance values, we will examine the importance of features, measured in GINI values, in the decision tree trained on the training data.

We can see below that a dry pump was most influential, which makes sense because, for obvious reasons, a dry well is likely to lead to a nonfunctional pump. The longitude was the second most important feature, which could be indicative of differences in urban/rural resources. Perhaps there is something happening geographically that means that different pumps receive different maintenance, funding, or use. It could also be differing climates, for example in semi-arid regions where there are rainy seasons, there could be large fluctuations in water turbidity and quality which could clog the pumps.

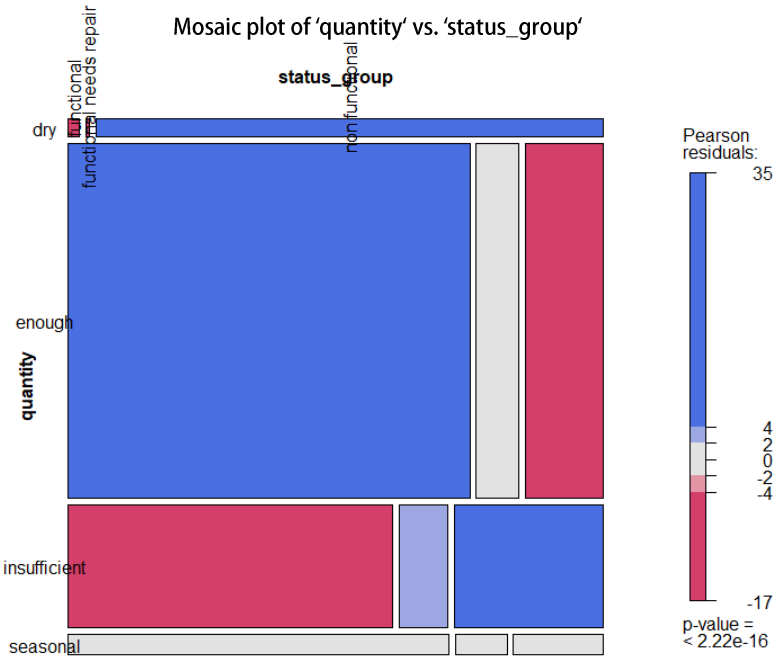




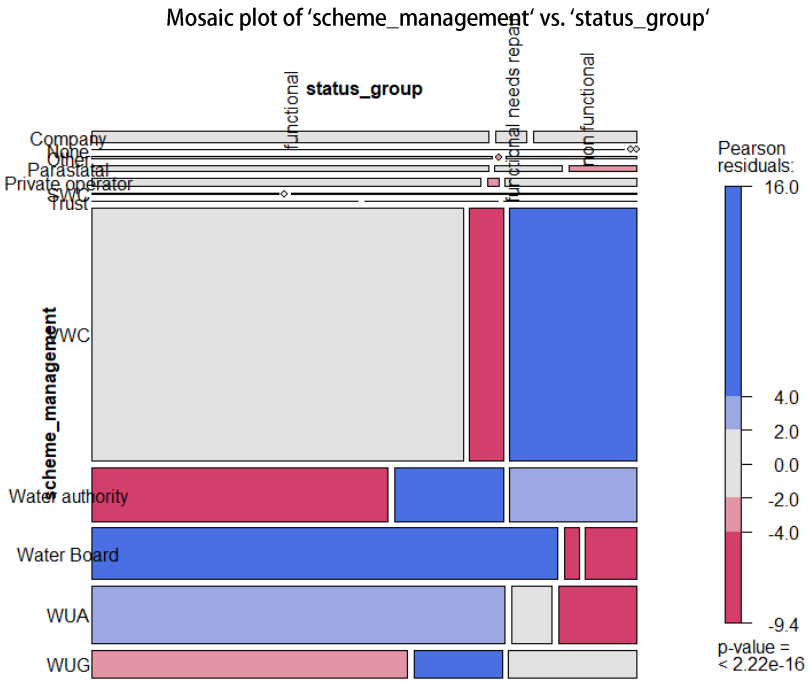
DT_feature_importances	features
0.287568	quantity_dry
0.174779	longitude
0.163201	waterpoint_type_other
0.133458	waterpoint_type_communal standpipe multiple
0.045505	quantity_insufficient
0.043614	scheme_management_Water authority



Whether the waterpoint type was `other` or `communal standpipe multiple` were the third and fourth most important indicators. We can see from the mosaic plot on the left that `communal standpipe multiple` corresponds strongly and negatively with functional pumps. We can see that on the other hand, `communal standpipes` correlate strongly and positively with functional pumps. Perhaps a lesson to be learned from this data is that communal standpipe functionality is put at risk when there are multiple standpipes relying on the same water pump.

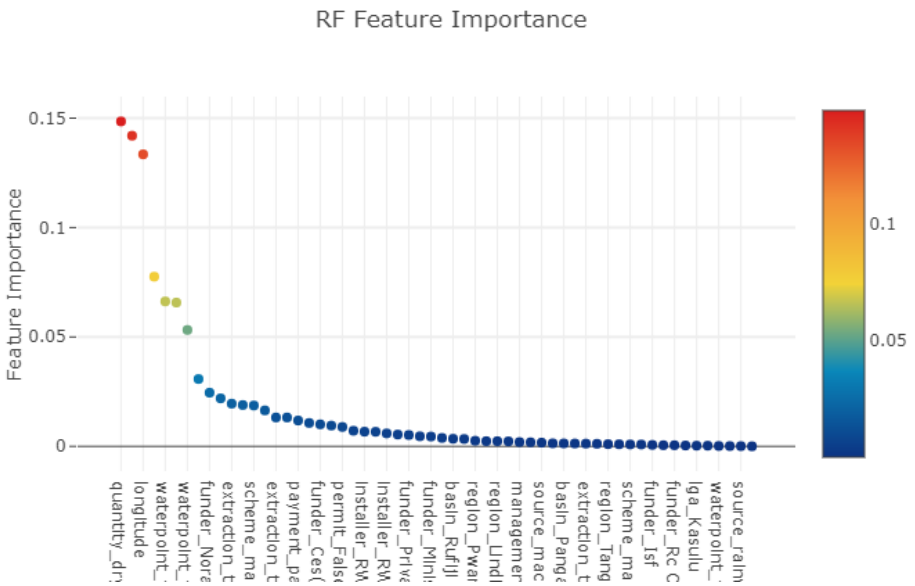


Whether the quantity was `insufficient` was the fifth most important indicator, and we can see from the mosaic plot on the left that it corresponds strongly and negatively to functional pumps. Perhaps pumps that have insufficient water supply are more susceptible to many intermitent periods of wet and dry, and thus rust more frequently. Or perhaps they are susceptible to use when the water source is dry, stagnant, and contaminated, leading to clogged pipes.



Whether or not the scheme management was `water_authority` was the sixth most important indicator, and we can see from the mosaic plot to the right that it strongly and negatively corresponds to functional pumps. Perhaps this information can be useful to a regulatory government agency, which can conduct an inquiry into the operations of that particular management group. It seems from the mosaic plot that it is the only group that has significantly poor results, so that is definitely an area for improvement.

It is important to keep in mind that the DT was working with 102 features and a maximum tree depth, which likely limited the number of features it even looked at when building the model. Thus, there could be more important features that never got assigned a GINI value. This is part of the benefit of random forests, since it builds many trees with random subsets of the data and aggregates the results. Thus analyzing the feature importances of such an ensemble would yield greater information.



RF_feature_importances	features
0.148492	quantity_dry
0.141973	waterpoint_type_other
0.133455	longitude
0.077545	basin_Lake Tanganyika
0.066232	waterpoint_type_communal standpipe
0.065714	quantity_enough
0.053200	waterpoint_type_communal standpipe multiple
0.030774	region_Kigoma
0.024528	funder_Norad
0.021929	lga_Kigoma Rural
0.019519	extraction_type_class_handpump
0.018896	quantity_insufficient
0.018667	scheme_management_Water Board
0.016440	scheme_management_Water authority
0.013199	extraction_type_class_gravity

Above, we can see the feature importances, as decided by the Random Forest Ensemble. The RF model agrees with all of the DT feature importances, but in addition, it has found several other important features to provide a much richer understanding of the problem.

It found that pumps in the `Lake Tanganyika` basin have a higher likelihood of being in need of repair or nonfunctional, which confirms our finding in the exploratory data analysis section. This finding could be due to poor water quality and/or turbidity in Lake Tanganyika, which could lead to clogged pumps.

It also found that a local government authority of `Kigoma Rural` is predictive of pumps needing repair, which confirms our finding in the exploratory data analysis section. Perhaps this local government authority hasn't enough revenue to properly maintain and service their pumps. In such a situation it would be worth calculating whether they would actually save money in the long term by properly maintaining the pumps, and perhaps the pumps could be kept functional as a long-term investment. The `region` `Kigoma` was also found to be important, which makes sense because the geographic area is likely the same or similar to the area where the local government authority of `Kigoma Rural`.

`quantity_enough` was found to be predictive, which was confirmed by our mosaic plot which showed that it corresponded strongly and positively with functional pumps. As speculated above, it is possible that pumps with continuous water flow could lead to less rusting and clogging of the pipes with debris-filled water.

`funder` being `Norad` was a slightly important predictor; it has a slightly higher proportion of functional pumps than other funders. However, this is quite low on the list of importance with a small GINI value (GINI=0.025), and thus must be taken with a grain of salt, as do all of the other features found to be important at or below this GINI level.

Conclusion:

The analysis done here demonstrates that the data routinely collected on water pumps is highly predictive of pump functionality, with an accuracy around 80%, even when used in multiple differing algorithms. Some of the most predictive features and insights include whether the water source is dry or contains a steady flow of water; longitude points to important regional and climate-related differences that affect water pump functionality; communal standpipe functionality is put at risk when there are multiple standpipes relying on the same water pump; the Water Authority scheme management consistently has trouble maintaining the functionality of pumps; pumps in the Lake Tanganyika basin have a higher likelihood of being in need of repair or nonfunctional; and pumps in the Kigoma region have a higher likelihood of being in need of repair or nonfunctional.

Though the dataset is rich, perhaps its ability to produce more accurate models could be improved by more complete and clean data. It is also limited by high cardinality in some features that could have proven to be predictive, but it is difficult to know due to limits in computational power and statistical theories. For example, if we tested all tens-of-thousands of values for particular feature, a large number of those values found to be important could more easily be due to chance alone. Though we are relying on many of these features to help us indirectly measure more pertinent predictors that we do not have in the dataset, it seems that they do a good job of illuminating potential latent variables. However, this will always be a limitation of the dataset that could be improved with more pertinent data.

Future work could use a feature selection that is based on the performance of the model (e.g. forward or backward selection) instead of a Chi2 test, because the latter will return the same data subsets for each algorithm to use. Different algorithms exploit different features and thus should receive a subset of data that maximizes the subsequent model's performance. However, when we tried forward selection on for DTs on all 1,276 variables, it took an extraordinary amount of time and thus would best be suited for a parallel computing setup. Another area for improvement would be using parallel computing to iterate over much larger ranges of parameters to find those that are the best, and perhaps implementing gradient descent to that end.