# PUMP IT UP
# DATA MINING THE WATER TABLE

**Group 1**

Alekhya Akkinepally

Hemanth Kumar Koraboina

Anvesh Kottapelli

Harika Katragadda

Guna Chandrika Paturi

Course: DSBA/ MBAD 6211- Advanced Business Analytics

Instructor: Dr. Mousavi

Spring 2018

05/08/2018

# 1 Executive Summary

## 1.1 Business Problem Statement

Using the data from Taarifa and the Tanzanian Ministry of Water we need to predict which water pumps are faulty i.e. classify each water pump into one of the three classifiers which are functional, needs repair and non functional based on number of variables about what kind of pump is operating, when it was installed and how it is managed. A better understanding of which waterpoint will fail will improve the maintenance operations and reduce the cost which would lead to the better access of water to the people of Tanzania.

## 1.2 Business Goal

The goal is to predict water pumps which are functional but needs repair. Identifying the pumps which needs repair could help the Tanzanian Ministry of water to take an action so as to keep them functioning without interruption. By identifying the problem well in advance could help the ministry to focus their maintenance resources where required and to improve the quality and quantity of the running water to the people of Tanzania.

## 1.3 Data Profile

The data that we collected from Taarifa and Tanzania Ministry of water supply is used to predict the water pumps which are functional, functional but needs repair and non functional.The train dataset has 59,401 observations and 40 features.Out of 40 features, 31 are categorical including the target variable which has three categories , 7 are continues and 2 are date variables. The dataset has features such as population, construction year, payment, water quality and pump locations.

## 1.4 Results

Based on our work we believe that the classification is hard to perform on this dataset as we focus mainly on 'Functional needs repair' class and the number of records of that class are very less in the dataset compared to the rest of the data. We aimed at a model which have high sensitivity, specificity and precision so that the model gives less false positives, high true positives and high true positives compared to false positives. So, based on these factors we have tried to assess our results. We have tried Random Forest and XGBoost models on our data and as the data was collected from a competition we don't have the target labels for the test dataset so we have submitted our results and got an accuracy of 82.16 for Random forest and 81.56 for XGBoost. To submit for the competition we can just choose the model with high accuracy. But just with accuracy we cannot decide on the model as accuracy cannot explain the model performance in detail to satisfy our business statement. So we have trained our model on train data and tested the model performance on the same train data.

Random Forest: Accuracy = 0.9476

| | functional | functional needs repair | non-functional |
|---|---|---|---|
| **Sensitivity** | 0.9850 | 0.74473 | 0.9332 |
| **Specificity** | 0.9135 | 0.99532 | 0.9862 |
| **Pos Pred Value** | 0.9312 | 0.92571 | 0.9768 |

XGBoost: Accuracy = 0.9456

| | functional | functional needs repair | non-functional |
|---|---|---|---|
| **Sensitivity** | 0.9738 | 0.78341 | 0.9366 |
| **Specificity** | 0.9231 | 0.99430 | 0.9774 |
| **Pos Pred Value** | 0.9377 | 0.91504 | 0.9628 |

Here the accuracies are almost similar but we are mainly focused on 'Functional Needs Repair class' if we compare that for both models:

| **Accuracy** | **Random Forest**<br>0.9476 | **XGBoost**<br>0.9456 |
|---|---|---|
| | **functional needs repair** | **functional needs repair** |
| **Sensitivity** | 0.78341 | 0.74473 |
| **Specificity** | 0.99430 | 0.99532 |
| **Pos Pred Value** | 0.91504 | 0.92571 |

If we compare the models to choose one based on above parameters, the accuracy and specificity is almost equal for the both the models so, the models give less false positives i.e. they classify functional, non-functional classes as functional needs repair less. For the model to be good we now need sensitivity and positive predictive value to be good, for the models above

they are 0.78,0.74 and 0.91,0.92 respectively. Here as we cannot afford false positives we tend to select the model with high sensitivity and high Positive Predictive value for us to satisfy our business statement and not to waste the management resources of the ministry on false predictions.

# 2 Project Report

## 2.1 Introduction

Water is critical for a country's development, as it is not only used in agriculture but also for industrial development. Many poorest countries across the world are facing water problems. Through this project we decided to analyse the water problem in Tanzania. Though Tanzania has access to a lot of water, the country still faces the dilemmas of many African countries where many areas have no reliable access to water.The data is provided by the Tanzanian Ministry of Water. If a good model is built by using the data , we will be able to address various business problems.

The economic impact is profound. In poor rural areas, women and girls spend, on average, 15 to 17 hours per week collecting water. Poor children around the world miss 443 million days of school each year because of water-related illnesses. In 60 countries in the developing world, more than half of primary schools have no adequate water facilities and nearly two thirds lack adequate sanitation, close to half of all people in developing countries suffer from health problems resulting from a lack of water and sanitation.

By increasing people's access to drinking water and sanitation is beneficial to the development of individual countries through improvements in health outcomes and the economy. The achievement will depend upon dramatically improving the reliability of handpumps, and therefore the availability of clean water. We can even analyze if the tanzania need more investments and also about how well the funds invested are utilized. In this project we will be able to predict based on the variables about what kind of pump is operating, when it was installed, and how it is managed.

By predicting the pumps which are functional but needs repair, decreases the overall cost for the Tanzanian Ministry of Water we will be able to understand which waterpoints will fail so that they can improve maintenance operations and ensure that clean, potable water is available to communities across Tanzania.

## 2.2 Background

A number of analytical methods are used for data mining.The standard and the common include regression , neural networks and decision trees.Decisions tree models are more flexible than the other two models mentioned, as decision tree model could be built even with the missing values.Different data mining techniques show set of activities performed by the analyst can be exhibited as a progression of coherent advances or undertakings. To start with, the input

dataset are imported from excel database and chose attributes or variables that are appropriate to analyze the output.Preprocessing is done on the datasets, either they remove the missing values or replace it with mean or mode and then the input data is then divided into training and test data sets.

There are two important goals for data mining prediction and description.Prediction is used to predict unknown values of the target variables description would find patterns.Data mining is a process of extracting hidden patterns for large amount of data that is used to take a write decisions.Data mining has become popular in banking field because there is a call for analytical methodology for detecting unknown and useful information in bank data.It can also be in financial sectors like customer segmentation and profitability, marketing, credit analysis, cross selling and many more.

In a research paper cited below have made an analysis on school dropouts, statistical method was used for the study of the student population.Then decision tree was build to analyse the number of the students who have been dropped out from the education at a specific level and the number of students who have continued their higher studies and the reasons like failing in a specific course or for some other reason behind the dropping out from education at certain level .They have also build neural networks and logistic regression and have performed model evaluation. Predicted if the student has dropped out or has not dropped from higher education and monitored the dropout trend.These type of analysis would raise awareness on the possibilities and need to use data mining models at the institution in order to decrease the number of dropouts from school.

Many water related issues are solved using the data science algorithms. The results helped to minimize the immediate damages, and improve resource-allocation decisions for similar water infrastructure crises. In one of the papers cited they developed an ensemble of predictive models to predict which locations are likely to have water with lead contamination. They used random forest, extremely randomized trees, logistic regression, nearest neighbor, and linear discriminant analysis (LDA) classifiers, and a second layer of a single XGBoost classifier for ensembling and made the decision with the error metric.

The other research papers which we have mentioned in the citation have also followed the similar steps from collecting the data till predicting the target variables. That would vary based on the variables that have been used and the target variable. Here for the bank loan analysis, models were implemented using WEKA tool.They have used three different algorithms - j48,bayesNet and naive Bayes was used to build a predictive models that can be used to predict and classify the application of loans that has categorized the customers to good and bad load by investigating customers behaviors and previous pay bank credit.

## 2.3 Data

The data comes from the Taarifa waterpoints dashboard, which aggregates data from the Tanzania Ministry of Water. The Taarifa Platform is an open source web API, designed to close citizen feedback loops. Taarifa is an open source platform for the crowdsourced reporting and

triaging of infrastructure related issues. Think of it as a bug tracker for the real world which helps to engage citizens with their local government. They are currently working on an Innovation Project in Tanzania, with various partners.

There are 59400 observations of 41 variables in the training dataset and in the test data there are 14850 observations of 40 variables. The less number of variables in test data is due to the absence of Status_Group which we need to predict.

**Table 1. Attributes**

| Name of the Feature | Data type | Description |
|---|---|---|
| amount_tsh | Numerical | Total static head |
| date_recorded | Factor | The date the row was entered |
| funder | Factor | Who funded the well |
| gps_height | Integer | Altitude of the well |
| installer | Factor | Organization that installed the well |
| longitude | Numerical | GPS coordinate |
| latitude | Numerical | GPS coordinate |
| wpt_name | Factor | Name of the waterpoint if there is one |
| num_private | Numerical | Number of the waterpoint |
| basin | Factor | Geographic water basin |
| subvillage | Factor | Geographic location |
| region | Factor | Geographic location |
| region_code | Integer | Geographic location(coded) |
| district_code | Integer | Geographic location(coded) |
| lga | Factor | Geographic location |
| ward | Factor | Geographic location |
| population | Integer | Population around the well |

| | | |
|---|---|---|
| public_meeting | Factor | True/False |
| recorded_by | Factor | Group entering this row of data |
| scheme_management | Factor | Who operates the waterpoint |
| scheme_name | Factor | Who operates the waterpoint |
| permit | Factor | If the waterpoint is permitted |
| construction_year | Integer | Year the waterpoint was constructed |
| extraction_type | Factor | The kind extraction the waterpoint uses |
| extraction_type_group | Factor | The kind of extraction the waterpoint uses |
| extraction_type_class | Factor | The kind of extraction the waterpoint uses |
| management | Factor | How the waterpoint is managed |
| management_group | Factor | How the waterpoint is managed |
| payment | Factor | What the water costs |
| payment_type | Factor | What the water costs |
| water_quality | Factor | The quality of the water |
| quality_group | Factor | The quality of the water |
| quantity | Factor | The quantity of water |
| quantity_group | Factor | The quantity of water |
| source | Factor | The source of the water |
| source_type | Factor | The source of the water |
| source_class | Factor | The source of the water |
| waterpoint_type | Factor | The kind of waterpoint |
| waterpoint_type_group | Factor | The kind of the waterpoint |

| **Table 2. Number of Missing values for each variable.** | | | |
|---|---|---|---|
| Id | 0 | scheme_name | 28166 |
| gps_height | 0 | extraction_type_class | 0 |
| num_private | 0 | water_quality | 0 |
| dist_code | 0 | source_type | 0 |
| recorded_by | 0 | date_recorded | 0 |
| extraction_type | 0 | latitude | 0 |
| payment | 0 | region | 0 |
| quantity_group | 0 | population | 0 |
| waterpoint_type_group | 0 | management | 0 |
| status_group | 0 | quality_group | 0 |
| installer | 3658 | source_class | 0 |
| basin | 0 | funder | 3635 |
| lga | 0 | wpt_name | 0 |
| scheme_management | 3877 | region_code | 0 |
| extraction_type_group | 0 | public_meeting | 3334 |
| source | 0 | construction_year | 0 |
| amount_tsh | 0 | management_group | 0 |
| longitude | 0 | quantity | 0 |
| subvillage | 371 | waterpoint_type | 0 |
| ward | 0 | waterpoint_type_group | 0 |
| permit | 3056 | payment_type | 0 |

## Pre Processing :

The amount of training data is huge and we have to make sure that our model does not overfit. One way to do this is to discard the variables that are not relevant.There are some set of features which have many levels , many NA's , less correlation and no logical relation with the target variable,hence we can remove such variables. These features are not useful for model creation.

num_private:

one such attribute in num_private where there are around 58643 0's out of 59400 observations in the training set. Hence we can remove the variable num_private.

amount_tsh:

We cannot discard the features just based on the number of missing values and levels because just discarding the attributes based on NA's will reduce the accuracy of our model. One such feature is Amount_tsh , where it has more that 40000 0's , but it is logically related to our target variable. So this attribute is very important in our model creation. Hence we should not discard this attribute.

quality_group**:**

Training data contains many features with redundant data such as the "water_quality" and the "quality_group". We have compared the summary of both the variables, compared their levels and decided to retain the variable quality_group and eliminate the water_quality as both the variables are redundant. Below are a set of features contain similar representation of data in different grains.

extraction_type,extraction_type_group,extraction_type_class.
payment,payment_type.
source,source_class.
subvillage,region,region_code,district_code,lga,ward.
waterpoint_type,waterpoint_type_group.

From the above set of groups we identified one feature from each group which has refined data and used it for model creation.

funder:

The variable "funder" which has 1897 levels, so we have tried grouping the levels which are rare or has low count into a level "other", Now after processing we have total of 16 levels including the level "other".

installer:

The next variable that we have worked on was “installer”, we have noticed that same name was taken as different levels because of a single capitals alphabet “District Council” and “District council” & “Fini Water” and “Fini water”. So we have changed the installer into lower case and now the levels have reduced.

recorder_by:

The variable recorded_by has only one level “GeoData Consultants Ltd” so we are not considering this variable as it is constant .The scheme_name has 28166 number of NA’S, if we process and put those under other we have realised it would of no helps so we have decided not to use this variable.We have removed the feature ”date_recorded” as it is difficult analyse and added two new features num_of_days and date_recorded_month.

public_meeting, scheme_management and permit:

After preprocessing for individual variable there are still some features such as public_meeting , scheme_management and permit variables which have more number of NA’s around 4000 each. we have to replace these NA’s to have a complete data set, that can be then analyzed using a traditional data analysis method. We imputed these NA’s using “hot deck” imputation. In hot deck imputation, each missing value is replaced with an observed response from a similar unit.

**Table 3. Summary Statistics**

| | **Observations** | **Mean** | **std.dev** | **Min** | **Max** |
|---|---|---|---|---|---|
| **amt_tsh** | 59,400 | 317.7 | 2997.574 | 0 | 350000 |
| **gps_height** | 59,400 | 668.3 | 693.116 | -90 | 2770 |
| **longitude** | 59,400 | 34.08 | 6.567 | 0 | 40.35 |
| **latitude** | 59,400 | -5.706 | 2.946 | -11.649 | 0 |
| **population** | 59,400 | 179.9 | 471.482 | 0 | 30500 |
| **construction_year** | 59,400 | 1301 | 951.620 | 0 | 2013 |
| **num_of_days** | 59,400 | 2104 | 334.216 | 1490 | 5558 |

## Univariate Analysis:

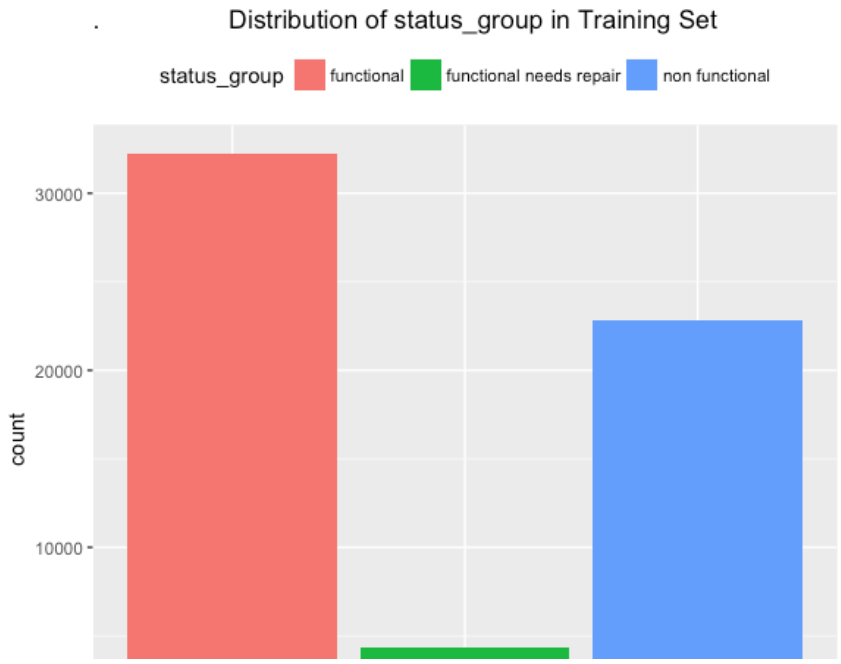


**Figure 1**

Figure 1. shows the count of functional,functional need repair, and non functional pumps in the Training data.We can clearly say that it is a Class Imbalance problem as we have very less observations for the class "functional needs repair" when compared to functional and non-functional classes. It will be a challenge in identifying the ones that need repair, simply because there are not as many observations for that level.

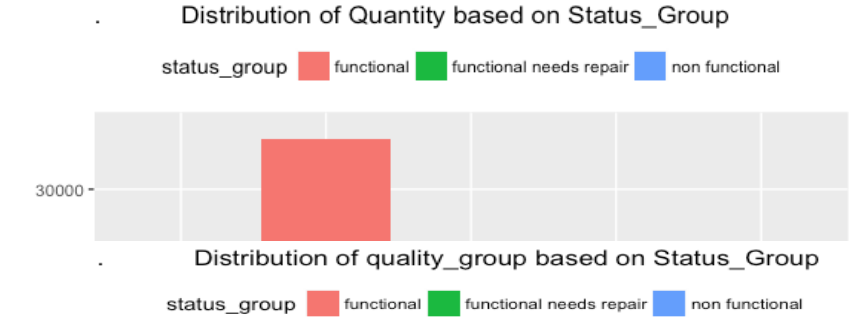


**Figure 2**

From the Figure 2. Plot it is clear that observations that have dry as quantity are classified as non-functional.

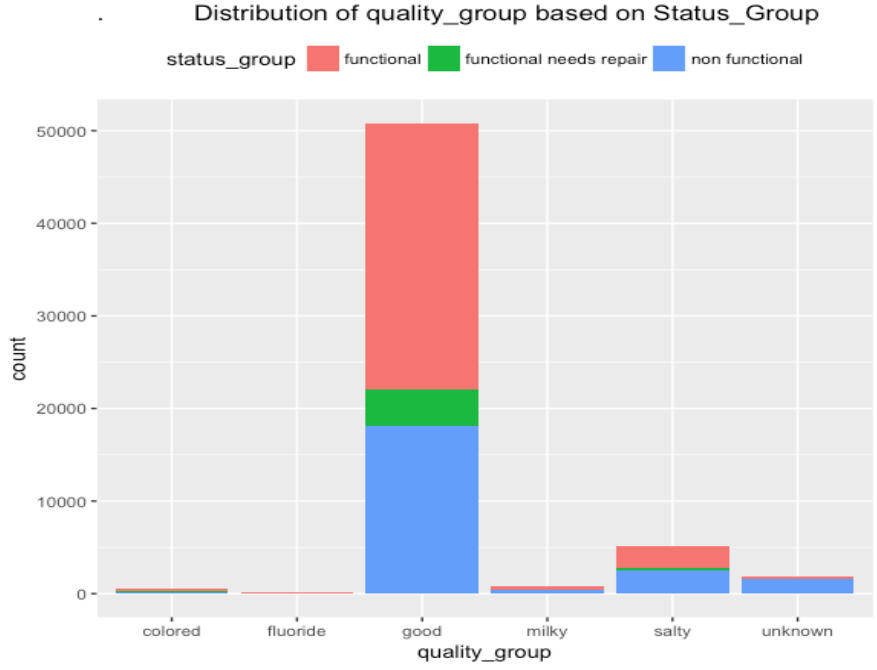


**Figure 3**

From the Figure 3. we can clearly observe that more

than 50,000 of the training observations have their quality as good.

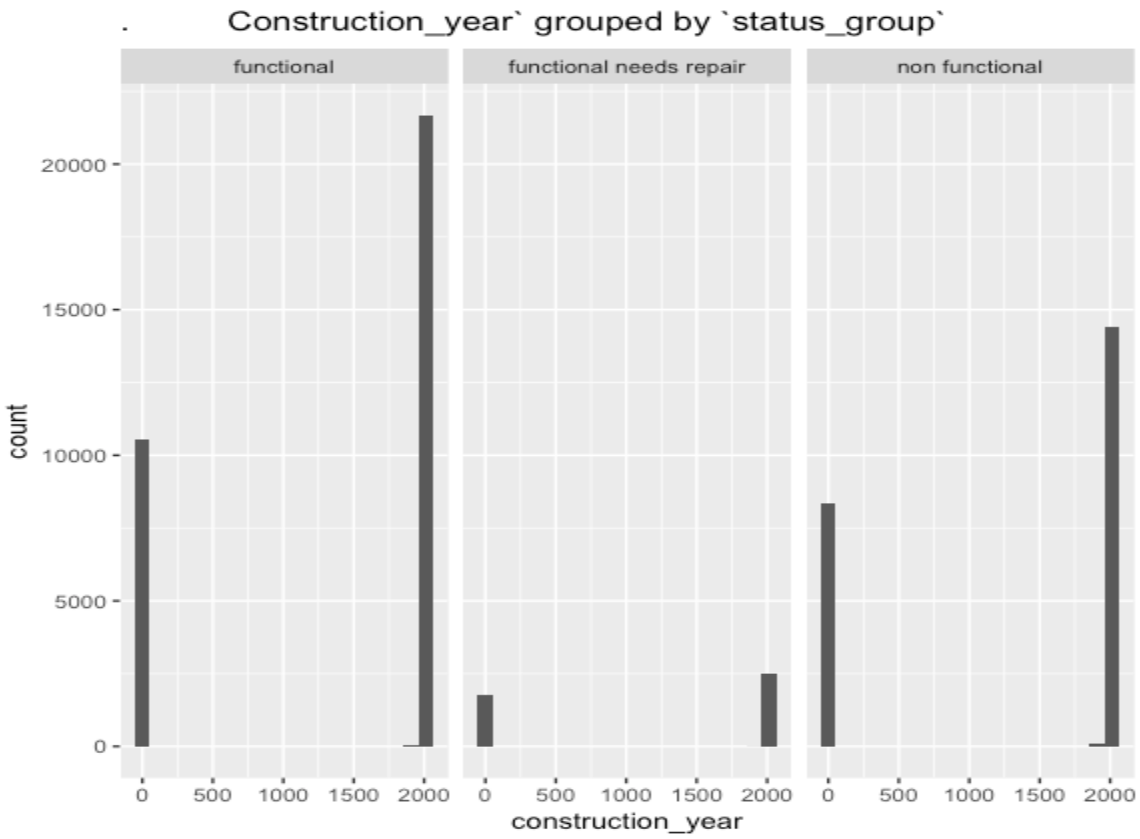


**Figure 4**

From the Figure 4. we can observe that there are some observations in the training set with coustruction_year as 0. So we cannot have the clear plot. So let us take the observations with construction_year greater than 0.

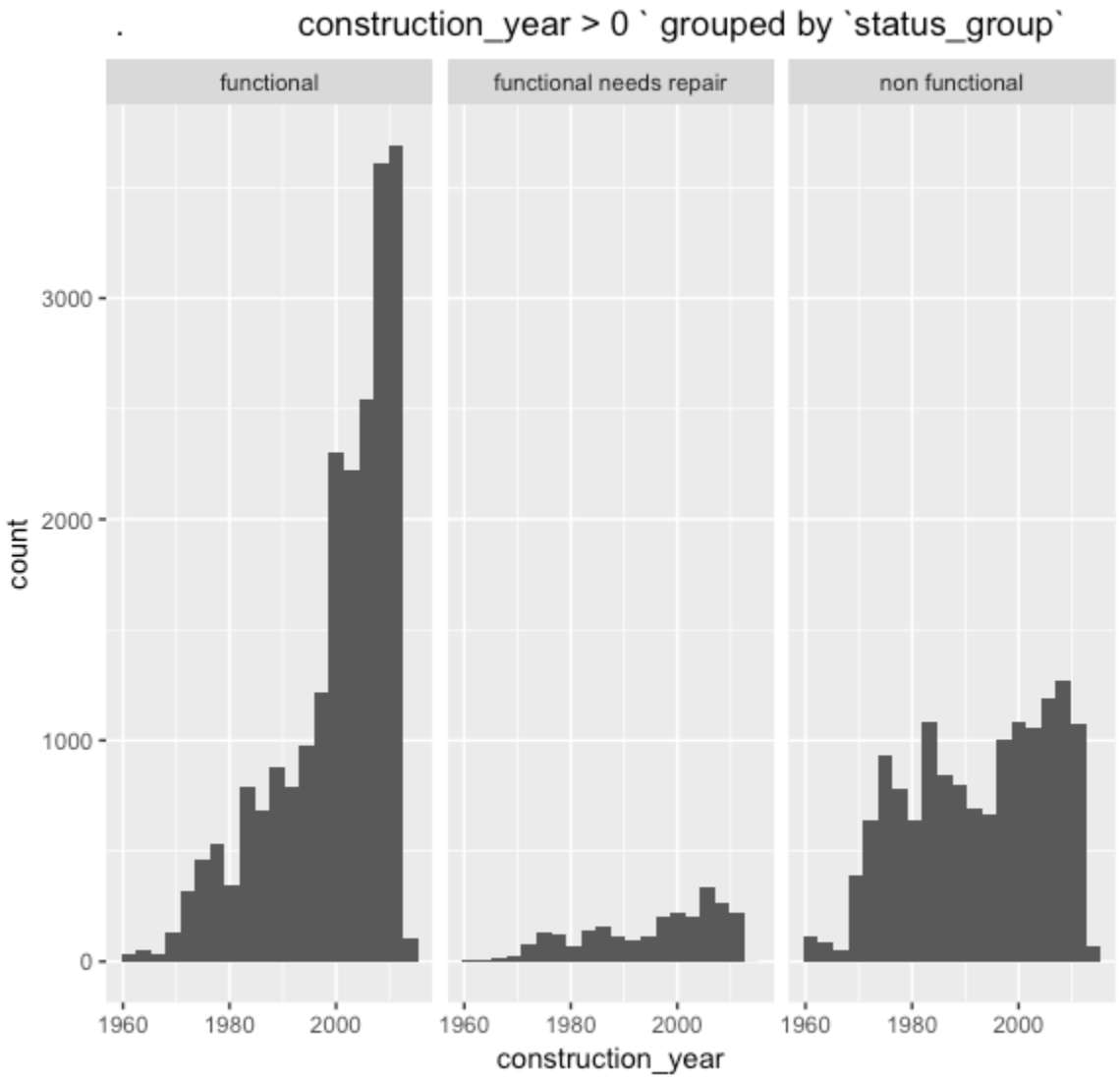


**Figure 5**

Above figure is similar to Figure 4 but with construction_year greater than 0. From the plot we can see that the number of functional pumps increased after 1980's and the number of non-functional pumps are almost same in 1990's through 2010.

## 2.4 Method

**1)Decision Tree:**

The first model we have implemented for our data set is the decision tree model.A decision tree is a tree-like graph that shows the possible consequences, including chance event outcomes. It is a way to represent an algorithm that only contains conditional control

statements.Decision trees are mostly used in decision analysis, to help recognise a strategy most likely to reach a goal.

A decision tree starts with a single node, which branches into possible outcomes. Each of those outcomes leads to additional nodes, which branch off into other possibilities giving it a treelike shape with different type of nodes.There are three different types of nodes: chance nodes, decision nodes, and end nodes. A chance node, is represented by a circle, and shows the probabilities of certain results. An end node shows the final outcome of a decision path and a decision node, represented by a square, shows a decision to be made.

A decision tree is a flowchart-like structure where each internal node indicate a "test" on an attribute,each leaf node represents a class label (decision taken after computing all attributes) and each branch represents the outcome of the test . The paths from root to leaf represent classification rules.Decision rules can be generated by establishing the association rules with the target variable on the right.

After the preprocessing, we have decided to build a Decision tree model. We used recursive partitioning (rpart) package which is usually used to build a classification or regression models of a general structure using a two stage procedure. We have used the rpart function from the rpart library and build a model using the training data set. We have used our final set of variables we have after the preprocessing.To plot the decision tree of our model we used packages “RColorBrewer” and “rpart.plot”.We have used “fancyRpartPlot” function for plotting the decision tree. Once we built the model , we predicted the test dataset using our model.

### 2)Random Forest:

Random forest is a ensemble learning method where average or mode of multiple decision trees is given as output. Multiple decision trees are constructed by training on different parts of data set. The random forest algorithm can be applied for both classification and regression.

Random forest method is mainly used to overcome the problem of overfitting the data by individual trees in other words random forest algorithm is used when model stops learning the patterns and starts learning the training data that it only fits for that individual tree instead of generalizable patterns.

In Random forest, “Random” refers to mainly two processes: 1. random observations to grow each tree and 2. random variables selected for splitting at each node.

Once we have the pre processed data we have split randomly 80% of the data in to training set and the remaining 20% to testing set.

we built our initial model with 10 trees with , status_group as our target variable and the remaining variables as the predictors. We had an OOB error of 23.51%. The we increased the number of trees in increments of 10 and built models until the OOB error is converged. Finally we got least OOB error with 19.25% with 60 trees.

Now we need to tune our model to find optimum number of variables at each split. With number of trees equal to 60 we got the least OOB error for 4 variables at each split.

Now with number of trees(nTree) = 60 and number of variables at each split(mTry)=4 we built our final Random forest Model.

Then we used this model to predict our test data set.

### Variable Importance:

The random forest trees most important feature ability to calculate variable importance which can be done using 2 ways , Gini Importance and Mean Decrease Accuracy. Gini Index is favorable for continuous variables and layered variables so for those cases it's better to use mean decrease in accuracy.

Mean Decrease Accuracy: How much the accuracy of the model decreases if we drop that variable. Mean Decrease Gini: Measure of variable importance based on the Gini impurity index used for the calculation of splits in trees.

We used varImpPlot function plot to plot the Variable Importance plot for the model we built.

### Boruta:

Variable selection also called as feature selection is very important step while performing a predictive modelling project as it is important to remove redundant variables and identify important variables. It helps in improving the accuracy and prevent overfitting. There are lot of packages for feature selection but 'Boruta'is prefered as it works well for both regression and classification.It is an improvement on random forest variable importance measure . By default Boruta uses Random Forest.

In this method shuffling of predictors' values is performed and later join them with the original predictors.A random forest is built on the merged dataset. Then comparison is made of the original variables with the randomised variables to measure the variable importance. Only variables with higher importance than that of the randomised variables are considered important.

### Difference between Boruta and Random Forest Importance Measure:

In random forest, the Z score is calculated by dividing the average accuracy loss by its standard deviation. For all the variables it is used as the importance measure. This Z score is not directly related to the statistical significance of the variable importance as we cannot use Z Score which is calculated in random forest, as a measure for finding variable importance.The boruta package runs random forest on both random and original attributes to compute the importance of all variables to overcome this problem. We repeat random permutation procedure to get statistically robust results,since the whole process is dependent on permuted copies.

## 3)XGBOOST

XGBoost (Extreme Gradient Boosting) is an optimized distributed gradient boosting library.

XGBoost belongs to a family of boosting algorithms that convert weak learners into strong learners. A weak learner is one which is slightly better than random guessing.

Boosting is a sequential process; i.e., trees are grown using the information from a previously grown tree one after the other. This process slowly learns from data and tries to improve its prediction in subsequent iterations

XGBoost parameters can be divided into three categories :

- General Parameters: Controls the booster type in the model which eventually drives overall functioning.
- Booster Parameters: Controls the performance of the selected booster.
- Learning Task Parameters: Sets and evaluates the learning process of the booster from the given data.

### *Parameters for Tree Booster*

1. nrounds[default=100]
    - It controls the maximum number of iterations. For classification, it is similar to the number of trees to grow.
    - Should be tuned using CV
2. eta[default=0.3][range: (0,1)]
    - It controls the learning rate, i.e., the rate at which our model learns patterns in data. After every round, it shrinks the feature weights to reach the best optimum.
    - Lower eta leads to slower computation. It must be supported by increase in nrounds.
    - Typically, it lies between 0.01 - 0.3
3. gamma[default=0][range: (0,Inf)]
    - It controls regularization (or prevents overfitting). The optimal value of gamma depends on the data set and other parameter values.
    - Higher the value, higher the regularization. Regularization means penalizing large coefficients which don't improve the model's performance. default = 0 means no regularization.
    - *Tune trick:* Start with 0 and check CV error rate. If you see train error >>> test error, bring gamma into action. Higher the gamma, lower the difference in train and test CV. If you have no clue what value to use, use gamma=5 and see the performance. Remember that gamma brings improvement when you want to use shallow (low max_depth) trees.
4. max_depth[default=6][range: (0,Inf)]
    - It controls the depth of the tree.
    - Larger the depth, more complex the model; higher chances of overfitting. There is no standard value for max_depth. Larger data sets require deep trees to learn the rules from data.
    - Should be tuned using CV

5. min_child_weight[default=1][range:(0,Inf)]
    - In regression, it refers to the minimum number of instances required in a child node. In classification, if the leaf node has a minimum sum of instance weight (calculated by second order partial derivative) lower than min_child_weight, the tree splitting stops.
    - In simple words, it blocks the potential feature interactions to prevent overfitting. Should be tuned using CV.
6. subsample[default=1][range: (0,1)]
    - It controls the number of samples (observations) supplied to a tree.
    - Typically, its values lie between (0.5-0.8)
7. colsample_bytree[default=1][range: (0,1)]
    - It control the number of features (variables) supplied to a tree
    - Typically, its values lie between (0.5,0.9)
8. lambda[default=0]
    - It controls L2 regularization (equivalent to Ridge regression) on weights. It is used to avoid overfitting.
9. alpha[default=1]
    - It controls L1 regularization (equivalent to Lasso regression) on weights. In addition to shrinkage, enabling alpha also results in feature selection. Hence, it's more useful on high dimensional data sets.

## 2.5 Results & Discussions

### Decision Tree:

The first model we built the decision tree model. Here we will discuss the performance of the decision tree model we built. Below is the overall statistics for the decision tree model. The accuracy of the model in classifying the test set in 70.98%. So the model does not perform much better in classifying the test data correctly. This might be due to less data for the class functional needs repair.

### Overall Statistics:

Accuracy : 0.7098

95% CI : (0.7016, 0.718)

No Information Rate : 0.5467

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.4147

Mcnemar's Test P-Value : < 2.2e-16

Below is the confusion matrix for the decision tree model. The model does not able to predict the class “functional needs repair”. Model is classifying all the observations as either functional or non-functional and makes 0 predictions for the class “functional needs repair”. Model performs better for the class functional , does some job for the class non-functional and does not work at all for the class functional needs repair. We can understand more about the model by analysing the tree that our model is built.

**Table 4. Confusion Matrix and Statistics-Decision Tree**

| | | Reference/Actual | | |
|---|---|---|---|---|
| | | **functional** | **functional needs repair** | **non functional** |
| **Prediction** | **functional** | 6008 | 693 | 2116 |
| | **functional needs repair** | 0 | 0 | 0 |
| | **non functional** | 487 | 151 | 2425 |

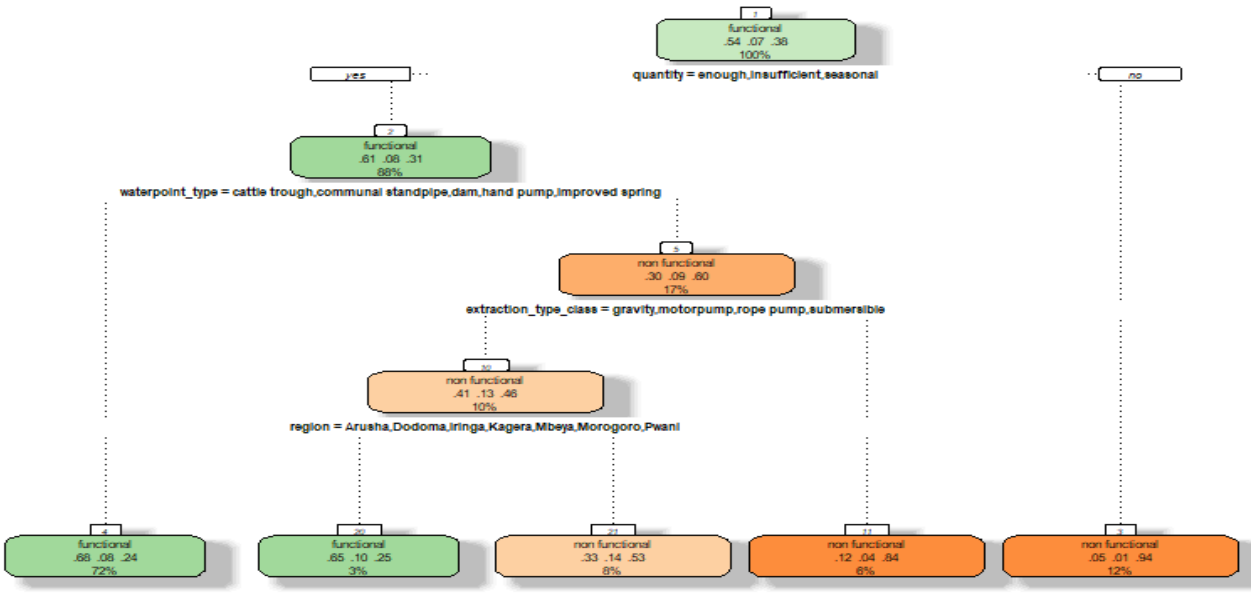


Figure 6-Decision Tree For Model

Figure 6 is the decision tree for the model we built. When we observe carefully the leaf nodes of the decision tree, there are three probabilities where the first probability refers to the

probability that the observation is functional , second probability refers to the probability that the observation is functional needs repair and third probability refers to the probability that the observation is non functional. The final class of the observation is the class for which the probability is high. We can see that all the observations in the training set are classified as either functional or non-functional but the model is not able to classify for the class " functional needs repair". Thus even though the model has an accuracy of around 71%, model is not able to classify one class out of three class. So we rejected the decision tree model.

**Table 5. Statistics by Class**

| | functional | functional needs repair | non functional |
|---|---|---|---|
| **Sensitivity** | 0.9250 | 0.0000 | 0.5340 |
| **Specificity** | 0.4784 | 1.0000 | 0.9131 |
| **Pos Pred Value** | 0.6814 | NaN | 0.7917 |
| **Neg Pred Value** | 0.8410 | 0.9289 | 0.7600 |
| **Prevalence** | 0.5467 | 0.0710 | 0.3822 |
| **Detection Rate** | 0.5057 | 0.0000 | 02041 |
| **Detection Prevalence** | 0.7422 | 0.0000 | 0.2578 |
| **Balanced Accuracy** | 0.7017 | 0.5000 | 0.7235 |

### Random Forest:

Here we discuss the performance of the Random forest model we built. From Table 6 we can observe the prediction for true functional class is 31775 and for true functional needs repair is 3215 and for true non functional is 21300. It is difficult to say how the model is performing just by looking at these numbers , we need to look at the parameters sensitivity, specificity to say if the model is doing a good job or not.

| Table 6. Confusion Matrix and Statistics-Random Forest | | | | |
|---|---|---|---|---|
| | Reference/Actual | | | |
| Prediction | | functional | functional needs repair | non functional |
| | functional | 31775 | 933 | 1414 |
| | functional needs repair | 148 | 3215 | 110 |
| | non functional | 336 | 169 | 21300 |

The True Positive rate or Recall or Sensitivity is 0.9850, 0.74473, 0.9332 for the classes functional, functional needs repair, non functional respectively. Therefore, the model has predicted 98% of the functional labels as functional, the model has predicted 75% of the functional needs repair labels as functional needs repair and model has predicted 93% of the non - functional labels as non - functional. Here as we can see the sensitivity is best for functional class, good for non-functional and better for the class functional needs repair. Similarly we can see the specificity is 0.9135, 0.99532, 0.9862 for functional, functional needs repair and non functional classes respectively. We can say that model is predicting better for functional class and non-functional classes and does a better job predicting the functional needs repairs.

| Table 7. Statistics by class –Random forest | | | |
|---|---|---|---|
| | Functional | functional needs repair | non-functional |
| Sensitivity | 0.9850 | 0.74473 | 0.9332 |
| Specificity | 0.9135 | 0.99532 | 0.9862 |

| | | | |
|---|---|---|---|
| **Pos Pred Value** | 0.9312 | 0.92571 | 0.9768 |
| **Neg Pred Value** | 0.9809 | 0.98030 | 0.9595 |
| **Prevalence** | 0.5431 | 0.07268 | 0.3842 |
| **Detection Rate** | 0.5349 | 0.05412 | 0.3586 |
| **Detection Prevalence** | 0.5744 | 0.05847 | 0.3671 |
| **Balanced Accuracy** | 0.9493 | 0.87002 | 0.9597 |

**Overall Statistics:**

Accuracy : 0.9476

95% CI : (0.9458, 0.9494)

No Information Rate : 0.5431

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9035

Mcnemar's Test P-Value : < 2.2e-16

The accuracy of the model is 94.76. The model has high accuracy than decision tree. So as far as we are concerned between decision tree and random forest, the random forest model is performing well in terms of accuracy, sensitivity and specificity.Decision tree was unable to classify instances of functional needs repair at all whereas Random forest has predicted 74% of them correctly.

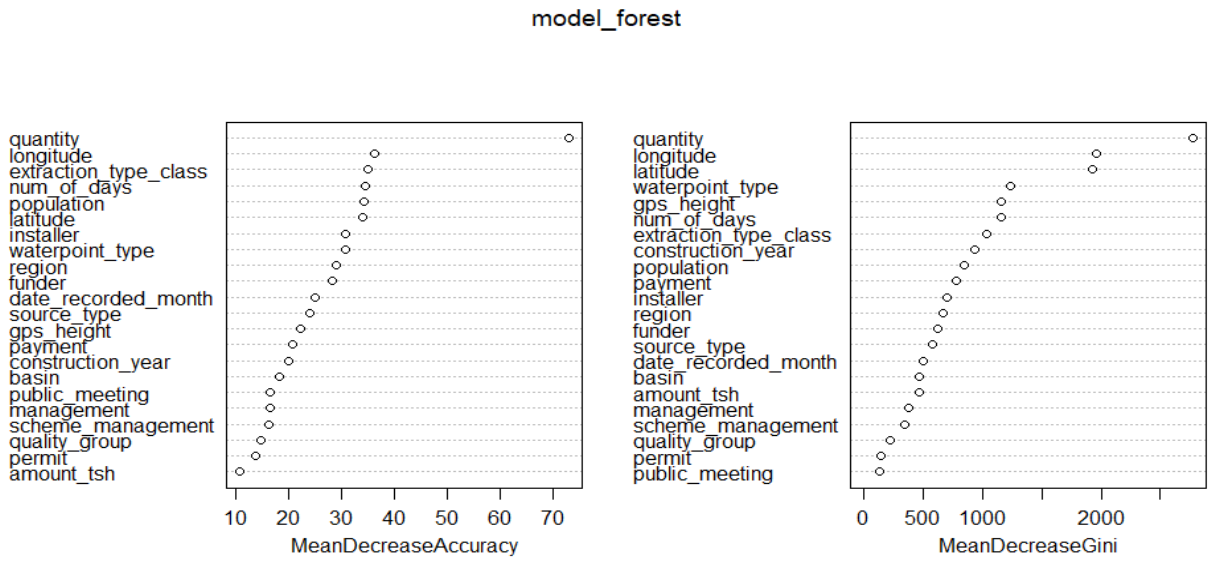


Figure 7. Variable Importance plot

Figure 7 is the variable importance plot for the random forest model we built. From the plot it is clear that "quantity " is the most important feature using both the techniques. Order of importance of features is almost same using both the techniques.Public_meeting, permit, amount_tsa are the least important variables but we cannot decide if we can remove any of the variables from our model. We can conclude that Variable importance plot can only say which feature is most important and which feature is not but , it cannot say if we can exclude the least important features.

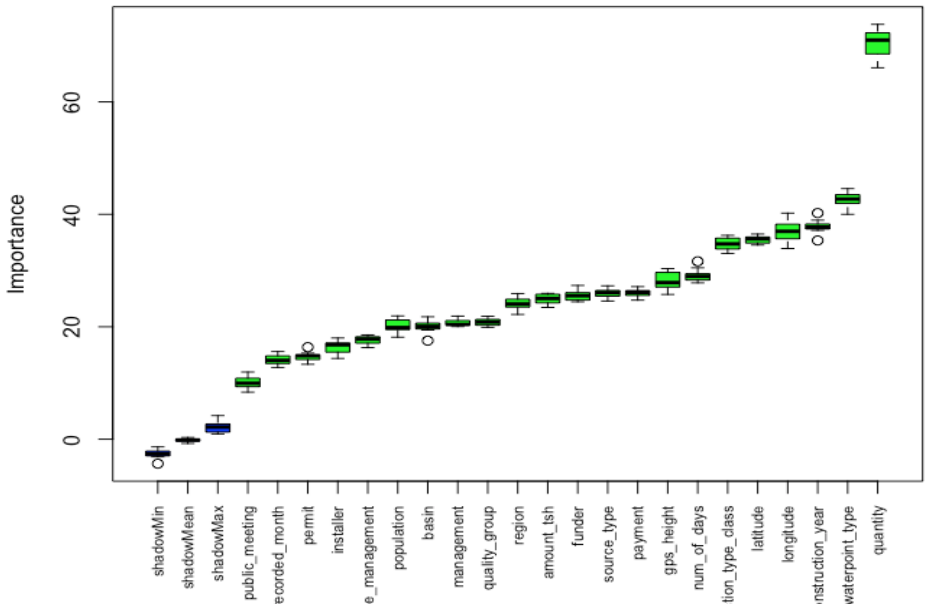


Figure 8 . Plot for feature selection using Boruta.

Using Boruta package we can say if we can remove any features that are least important. Figure 8. is the plot using the Boruta package for feature selection. It is clear that all our final set of variable are above shadowMax. We can conclude that all the final set of features we used to build our model are useful in predicting.

### XGBoost :

The third model we applied on our data set is XGBoost algorithm.The results from the model we built are as below. From Table 8 we can observe the prediction for true functional class is 31413 and for true functional needs repair is 3382 and for true non functional is 21376.

**Table 8. Confusion Matrix and Statistics-XGBOOST**

| | | Reference/Actual | | |
|---|---|---|---|---|
| **Prediction** | | **functional** | **functional needs repair** | **non functional** |
| | **functional** | 31413 | 750 | 1338 |
| | **functional needs repair** | 204 | 3382 | 110 |
| | **non functional** | 642 | 185 | 21376 |

### Overall Statistics

Accuracy : 0.9456

95% CI : (0.9438, 0.9474)

No Information Rate : 0.5431

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9004

Mcnemar's Test P-Value : < 2.2e-16

The Recall or Sensitivity is 0.9738, 0.78341, 0.9366 for the classes functional, functional needs repair, non functional respectively. Therefore, the model has predicted 97% of the functional labels as functional, the model has predicted 78% of the functional needs repair labels as functional needs repair and model has predicted 93% of the non - functional labels as non - functional. We can observe that both the models Random forest and XGBoost has similar accuracy but sensitivity for functional need repair class is higher for XGBoost. So as per our problem statement which is to identify the pumps that need repairs , XGBoost does a better job than Random Forest.

**Table 9. Statistics by class -XGBoost**

| | **Functional** | **functional needs repair** | **non-functional** |
|---|---|---|---|
| **Sensitivity** | 0.9738 | 0.78341 | 0.9366 |
| **Specificity** | 0.9231 | 0.99430 | 0.9774 |
| **Pos Pred Value** | 0.9377 | 0.91504 | 0.9628 |
| **Neg Pred Value** | 0.9673 | 0.98321 | 0.9611 |
| **Prevalence** | 0.5431 | 0.07268 | 0.3842 |
| **Detection Rate** | 0.5288 | 0.05694 | 0.3599 |
| **Detection Prevalence** | 0.5640 | 0.06222 | 0.3738 |
| **Balanced Accuracy** | 0.9484 | 0.88886 | 0.9570 |

## 2.6 Conclusion

As we focus mainly on 'Functional needs repair' class and the number of records of that class are very less in the dataset compared to the rest of the data. We aimed at a model which have high sensitivity, specificity and precision so that the model gives less false positives, high true positives and high true positives compared to false positives. So, based on these factors we have tried to assess our results. We have tried Random Forest and XGBoost models on our data and as the data was collected from a competition we don't have the target labels for the test dataset so we have submitted our results and got an accuracy of 82.16 for Random forest and 81.56 for XGBoost.

The obvious contenders for the final algorithm are XGBoost and Random Forest. Random Forest is a widely accepted as a strong learner and has a good theoretical base for prediction. Boosting also has many followers who bet on its supremacy as a strong learner. But, a good learner should also be efficient whilst giving out good accuracy. Considering our case, Random Forest and XG Boost both have given similar accuracy.

To choose one model based on above parameters we obtained, the accuracy and specificity is almost equal for the both the models so, the models give less false positives i.e. they classify functional, non-functional classes as functional needs repair less. For the model to be good we now need sensitivity and positive predictive value to be good, for the models above they are 0.78,0.74 and 0.91,0.92 for XGBoost and Random Forest respectively. Here as we cannot afford false positives we tend to select the model with high Positive Predictive value for us to satisfy our business statement and not to waste the management resources of the ministry on false predictions so, we chose XGBoost with high sensitivity and high positive predictive value for Functional need repair class.

## 2.7 References


1) Aboobyda Jafar Hamid,Tarig Mohammed Ahmed. (2016), "Prediction Model Of Loan Risk in Banks using Data Mining", Department of Computer Sciences ,University Khartoum, Sudan.
2) Analytics Vidhya, "A complete tutorial for tree based modelling" https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/
3) Analytics Vidhya, "Feature selection using Boruta package in R" https://www.analyticsvidhya.com/blog/2016/03/select-important-variables-boruta-package
4) Deepanshu Bhalla "RANDOM FOREST IN R : STEP BY STEP TUTORIAL" https://www.listendata.com/2014/11/random-forest-with-r.html
5) Hackerearth, https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/beginners-tutorial-on-xgboost-parameter-tuning-r/tutorial/
6) Jacob Abernethy,Cyrus Anderson,"Flint Water Crisis: Data-Driven Risk Assessment Via Residential Water Testing ",University of Michigan ,MI.
7) Leo Breiman and Adele Cutler-"Random Forests" https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

8) Rebecca R.Andrige, “A Review of Hot Deck Imputation for Survey Non Response” https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3130338/
9) Xiaodong Yang “Titanic – Machine Learning From Disaster”, Northwestern University.
10) Zeljko Garaca, Maja Cukusic, Mario jadric (2010), “Student Dropout Analysis with application of data mining methods”,Vol 1,pp. 31-46.

## 2.8 Appendix

```
library(data.table)

library(caret)

library(randomForest)

#Reading the Data Set

train_labels <- read.csv("Training_set_labels.csv", sep=",", header=T, strip.white = T, na.strings = c("NA","NaN","","?"))

train_data <- read.csv("Training_set_values.csv", sep=",", header=T, strip.white = T, na.strings = c("NA","NaN","","?"))

train_data$status_group <-as.factor(train_labels$status_group)

Original_train_data<-train_data

test_set_orginal <- read.csv("Test_set_values.csv", sep=",", header=T, strip.white = T, na.strings = c("NA","NaN","","?"))

test_set_orginal_copy<-test_set_orginal

#Constructing a table that shows the distribution of status values  in  training set

table(train_data$status_group)

#Shows the proportion of status values  in  training set

prop.table(table(train_data$status_group))

#Constructing a table that shows the distribution of quantity vs status group  in  training set

table(train_data$quantity , train_data$status_group)

# Shows the proportion of quantity vs status group  in  training set

prop.table(table(train_data$quantity, train_data$status_group), margin = 1)

#Using the library ggplot for the plots

library(ggplot2)
```

```
# Plot for distribution of Quantity based on Status_Group
qplot(quantity, data=train_data, geom="bar", fill=status_group) +
  theme(legend.position = "top")+
  ggtitle("Plot showing distribution of Quantity based on Status_Group")
# Plot for distribution of quality_group based on Status_Group
qplot(quality_group, data=train_data, geom="bar", fill=status_group) +
  theme(legend.position = "top")+
  ggtitle("Plot showing distribution of quality_group based on Status_Group")
# Plot for construction_year` grouped by `status_group
ggplot(train_data, aes(x =construction_year)) +
  geom_histogram(bins = 20) +
  facet_grid( ~ status_group)+
  ggtitle("Plot showing construction_year` grouped by `status_group`")
# Plot for construction_year > 0 ` grouped by `status_group
ggplot(subset(train_data, construction_year > 0), aes(x = construction_year)) +
  geom_histogram(bins = 20) +
  facet_grid( ~ status_group)+
  ggtitle("Plot showing construction_year > 0 ` grouped by `status_group`")
train_data<-subset(train_data,select  =  -c(id,water_quality,wpt_name,waterpoint_type_group,
num_private,extraction_type,extraction_type_group,payment_type,source,source_class,region
_code,district_code,subvillage,lga,ward,quantity_group,recorded_b,scheme_name,manageme
nt_group))
test_set_orginal<-subset(test_set_orginal,select=                                    -
c(id,water_quality,wpt_name,waterpoint_type_group,
num_private,extraction_type,extraction_type_group,payment_type,source,source_class,region
_code,          district_code,          subvillage,lga,          ward,          quantity_group,
recorded_by,scheme_name,management_group))
num_of_days <- as.numeric(as.Date("2014-01-01") - as.Date(train_data$date_recorded))
date_recorded_month <- factor(format(as.Date(train_data$date_recorded), "%b"))
train_data <- train_data[, -which(names(train_data) == "date_recorded")]
```

```
train_data <- cbind(train_data, num_of_days)
train_data <- cbind(train_data, date_recorded_month)

num_of_days_test <- as.numeric(as.Date("2014-01-01") - as.Date(test_set_orginal$date_recorded))
date_recorded_month_test <- factor(format(as.Date(test_set_orginal$date_recorded), "%b"))
test_set_orginal <- test_set_orginal[, -which(names(test_set_orginal) == "date_recorded")]
test_set_orginal$num_of_days <- num_of_days_test
test_set_orginal$date_recorded_month <- date_recorded_month_test

#Reduce the number of levels in the Funder(15 Levels)
train_data$funder<-replace(train_data$funder,train_data$funder=='0',NA)
funder_names<-names(summary(train_data$funder)[1:15])
funder <- factor(train_data$funder, levels=c(funder_names, "Other"))
funder[is.na(funder)] <- "Other"
train_data$funder <- funder
#summary(train_data$funder)
#nlevels(train_data$funder)
test_set_orginal$funder<-replace(test_set_orginal$funder,test_set_orginal$funder=='0',NA)
funder_test <- factor(test_set_orginal$funder, levels=c(funder_names, "Other"))
funder_test[is.na(funder_test)] <- "Other"
test_set_orginal$funder <- funder_test

#Converting to lower case installer column
#Reduce the number of levels to 15 + other
train_data$installer<-replace(train_data$installer,train_data$installer=='0',NA)
train_data$installer <- substr(tolower(train_data$installer),1,3)
train_data$installer<-as.factor(train_data$installer)
```

```
installer_names<-names(summary(train_data$installer)[1:15])
installer <- factor(train_data$installer, levels=c(installer_names, "Other"))
installer[is.na(installer)] <- "Other"
train_data$installer<-as.factor(installer)
#summary(train_data$installer)
#nlevels(train_data$installer)
#nlevels(as.factor(tolower(train_data$installer)))
test_set_orginal$installer<-
replace(test_set_orginal$installer,test_set_orginal$installer=='0',NA)
test_set_orginal$installer <- substr(tolower(test_set_orginal$installer),1,3)
test_set_orginal$installer<-as.factor(test_set_orginal$installer)
installer_test <- factor(test_set_orginal$installer, levels=c(installer_names, "Other"))
installer_test[is.na(installer_test)] <- "Other"
test_set_orginal$installer<-as.factor(installer_test)

train_data$scheme_management<-
replace(train_data$scheme_management,train_data$scheme_management=='None','Other')
train_data$scheme_management<-factor(train_data$scheme_management)
library(VIM)
train_data<-hotdeck(train_data)
train_data<-train_data[,1:23]
test_set_orginal<-hotdeck(test_set_orginal)
test_set_orginal<-test_set_orginal[,1:22]
trainIndex = sample(1:nrow(train_data),
                    size = round(0.8*nrow(train_data)),
                    replace=FALSE)
train_sample = train_data[trainIndex,]
test_sample = train_data[-trainIndex,]
```

```
#Random Forest
# Load the randomForest library
library(randomForest)
model_forest <- randomForest(as.factor(status_group) ~ ., data = train_sample,
                             importance = TRUE, ntree = 100,proximity=F, na.action=na.exclude)

pred_forest_test_sample <- predict(model_forest, test_sample)
library(caret)
confusionMatrix(pred_forest_test_sample, test_sample$status_group)
#Predicting the original Test set
pred_forest_test_original<-predict(model_forest, test_set_orginal)

final_op_forest<-
data.frame(id=test_set_orginal_copy$id,status_group=pred_forest_test_original)

write.csv(final_op_forest, file = "pump1.csv")



#Random Forest Model 2
#Load the randomForest library
library(randomForest)
model_forest_1 <- randomForest(as.factor(status_group) ~ ., data = train_data,
                             importance = TRUE, ntree = 60,mtry=4,proximity=F, na.action=na.exclude)
pred_forest_test_sample_1 <- predict(model_forest_1, train_data)
library(caret)
confusionMatrix(pred_forest_test_sample_1, train_data$status_group)
#Predicting the original Test set
pred_forest_test_original_1<-predict(model_forest_1, test_set_orginal)
```

```
final_op_forest_1<-
data.frame(id=test_set_orginal_copy$id,status_group=pred_forest_test_original_1)

levels(final_op_forest_1$status_group)<-c('functional','functional needs repair','non functional')

write.csv(final_op_forest_1, file = "pump2.csv")

#Sampling

library(DMwR)

smote_over<-SMOTE(status_group ~ .,train_data,perc.over = 200)

table(smote_over$status_group)

prop.table(table(smote_over$status_group))

model_forest_smote_over <-randomForest(as.factor(status_group) ~ ., data=smote_over, mtry=4, importance=TRUE, ntree=60)

confusionMatrix(predict(model_forest_smote_over, train_data), train_data$status_group)

smote_under<-SMOTE(status_group ~ .,train_data,perc.under = 200)

table(smote_under$status_group)

prop.table(table(smote_under$status_group))

model_forest_smote_under <-randomForest(as.factor(status_group) ~ ., data=smote_under, mtry=4, importance=TRUE, ntree=60)

confusionMatrix(predict(model_forest_smote_under, train_data), train_data$status_group)

smote_over_under<-SMOTE(status_group ~ .,train_data,perc.over = 100,perc.under = 100)

table(smote_over_under$status_group)

prop.table(table(smote_over_under$status_group))

model_forest_smote_over_under <-randomForest(as.factor(status_group) ~ ., data=smote_over_under, mtry=4, importance=TRUE, ntree=60)

confusionMatrix(predict(model_forest_smote_over_under, train_data), train_data$status_group)

#XGBoost

library(caret)

library(xgboost)

ControlParamteres <- trainControl(method = "cv",
```

```
                    number = 5,
                    savePredictions = TRUE,
                    classProbs = TRUE
)
levels(train_data$status_group) <- make.names(levels(factor(train_data$status_group)))
parameterGrid <-  expand.grid(eta = 0.7,
                    colsample_bytree=1,
                    max_depth=20,
                    nrounds=15,
                    gamma=1,
                    min_child_weight=2,
                    subsample=0.7
)
#Building XGBoost on the model
modelxgboost <- train(status_group~.,
                data = train_data,
                method = "xgbTree",
                trControl = ControlParamteres,
                tuneGrid=parameterGrid,
                metric = "Accuracy",
                verbose = 1,
                num_class = 3
)
confusionMatrix(predict(modelxgboost, train_data), train_data$status_group)
pred_xgboost <- predict(modelxgboost, test_set_orginal)
final_op<-data.frame(id=test_set_orginal_copy$id,status_group=pred_xgboost)
levels(final_op$status_group)
levels(final_op$status_group)<-c('functional','functional needs repair','non functional')
```

```
write.csv(final_op, file = "xgboost.csv")

final_data <- cbind(test_data, predicted_values)

colnames <- c(colnames(test_data),"prob.one")

write.table(final_data, file="ann_predictions.csv", sep=",", row.names=F, col.names = colnames)

test_op<-data.frame(id=test_data$id,status_group=pred_forest_test)

write.csv(test_op, file = "pump.csv")
```