INFO-F-422 - Statistical Foundations of Machine Learning

Student 1 - [Kate.Suzanna.Delbeke@vub.be (Kate.Suzanna.Delbeke@vub.be)](mailto:Kate.Suzanna.Delbeke@vub.be) - Student ID 577345

Student 2 - [Somayyeh.Gholami@vub.be (somayyeh.gholami@vub.be)](mailto:somayyeh.gholami@vub.be) - Student ID 562113

Student 3 - [yiming.yan@vub.be (yiming.yan@vub.be)](mailto:yiming.yan@vub.be) - Student ID 573780

Video presentation: [https://www.youtube.com/watch?v=5_F78Vy16yo (https://www.youtube.com/watch?v=5_F78Vy16yo)](https://www.youtube.com/watch?v=5_F78Vy16yo)

Pump it Up: Data Mining the Water Table

1.Introduction

Pump it Up: Data Mining the Water Table

The goal of this project is to simplify the maintenance processing of water pumps and guarantee that clean, potable water is available to every Tanzanian.

This notebook aims to build a predictive model which is able to correctly predict which pumps are functional, which need some repairs, and which don't work at all, using data from Taarifa and the Tanzanian Ministry of Water. The model will be trained using the given Training set values, Training set labels files available on the DrivenData platform, it includes roughly 60000 labeled samples and 40 features. We will be applying 3 different models on the dataset and evaluating them accordingly

The main steps we're going to do in this project:

- **Data preprocessing**
  - **Retrieve data into the DataFrame To easily manipulate**
  - **Handle data types in datasets like missing values, categorical variables**
  - **Design pipelines to enhance the performance of machine learning**
  - **Feature Selections and Feature Engineering**
- **Define a model**
  - **Decision Trees**
  - **KNN Model**

  - **Random Forest Tree**
  - **xgboost**
- **Applying different models of machines learning and examinate the results(scores),To check which the model works better with this datatset**

- **Fit a Model**

- **Make prediction**

- **Validate The model**
  - **Applying the advanced techniques for model validation (for example: cross-validation)**

2.Data preprocessing

2.0 Load Packets(Import necessary libraries)

If packages are not installed, error will occur.
Make sure they are installed.

In [2]:

```
install.packages(c("tidyverse","GoodmanKruskal","rpart","randomForest","lazy","xgboost","Matrix","Ma
```

```
Warning message in install.packages(c("tidyverse", "GoodmanKruskal", "rpart", "rando
mForest", :
 "installation of package 'rpart' had non-zero exit status"
Warning message in install.packages(c("tidyverse", "GoodmanKruskal", "rpart", "rando
mForest", :
 "installation of package 'randomForest' had non-zero exit status"
Warning message in install.packages(c("tidyverse", "GoodmanKruskal", "rpart", "rando
mForest", :
 "installation of package 'lazy' had non-zero exit status"
Warning message in install.packages(c("tidyverse", "GoodmanKruskal", "rpart", "rando
mForest", :
 "installation of package 'Matrix' had non-zero exit status"
Warning message in install.packages(c("tidyverse", "GoodmanKruskal", "rpart", "rando
mForest", :
 "installation of package 'data.table' had non-zero exit status"
Warning message in install.packages(c("tidyverse", "GoodmanKruskal", "rpart", "rando
mForest", :
 "installation of package 'xgboost' had non-zero exit status"
Updating HTML index of packages in '.Library'

Making 'packages.html' ...
 done
```

In [2]:

```
library(tidyverse)
library(GoodmanKruskal)
library(rpart)
library(randomForest)
library(lazy)
```

```
Warning message:
"package 'tidyverse' was built under R version 4.0.5"
-- Attaching packages ------------------------------------------------------------
------------------- tidyverse 1.3.1 --

v ggplot2 3.3.3     v purrr   0.3.4
v tibble  3.1.1     v dplyr   1.0.5
v tidyr   1.1.3     v stringr 1.4.0
v readr   1.4.0     v forcats 0.5.1

Warning message:
"package 'ggplot2' was built under R version 4.0.5"
Warning message:
"package 'tibble' was built under R version 4.0.5"
Warning message:
"package 'tidyr' was built under R version 4.0.5"
Warning message:
"package 'readr' was built under R version 4.0.5"
Warning message:
"package 'purrr' was built under R version 4.0.5"
Warning message:
"package 'dplyr' was built under R version 4.0.5"
Warning message:
"package 'stringr' was built under R version 4.0.5"
Warning message:
"package 'forcats' was built under R version 4.0.5"
-- Conflicts ----------------------------------------------------------------------
------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()

Warning message:
"package 'GoodmanKruskal' was built under R version 4.0.5"
Warning message:
"package 'rpart' was built under R version 4.0.5"
Warning message:
"package 'randomForest' was built under R version 4.0.5"
randomForest 4.6-14

Type rfNews() to see new features/changes/bug fixes.


Attaching package: 'randomForest'


The following object is masked from 'package:dplyr':

    combine


The following object is masked from 'package:ggplot2':
```

margin

2.1 Read the csv File and convert them to dataframe

In [3]:

```
#Read File
Train <- read.csv("TrainingSetValues.csv", header = TRUE)
class_var <- read.csv('TrainingSetLabels.csv', header = TRUE)
Train
```

A data.frame: 59400 × 40

id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_nam
<int>	<dbl>	<chr>	<chr>	<int>	<chr>	<dbl>	<dbl>	<chr
69572	6000	2011-03-14	Roman	1390	Roman	34.93809	-9.85632177	nor
8776	0	2013-03-06	Grumeti	1399	GRUMETI	34.69877	-2.14746569	Zahana
34310	25	2013-02-25	Lottery Club	686	World vision	37.46066	-3.82132853	Kw Mahun
67743	0	2013-01-28	Unicef	263	UNICEF	38.48616	-11.15529772	Zahanati Y Nanyumb
19728	0	2011-07-13	Action In A	0	Artisan	31.13085	-1.82535885	Shule

In [4]:

```
colnames(Train)
```

'id' 'amount_tsh' 'date_recorded' 'funder' 'gps_height' 'installer' 'longitude' 'latitude' 'wpt_name' 'num_private' 'basin' 'subvillage' 'region' 'region_code' 'district_code' 'lga' 'ward' 'population' 'public_meeting' 'recorded_by' 'scheme_management' 'scheme_name' 'permit' 'construction_year' 'extraction_type' 'extraction_type_group' 'extraction_type_class' 'management' 'management_group' 'payment' 'payment_type' 'water_quality' 'quality_group' 'quantity' 'quantity_group' 'source' 'source_type' 'source_class' 'waterpoint_type' 'waterpoint_type_group'

The above cell gives us the all features:

'id''amount_tsh''date_recorded''funder''gps_height''installer''longitude''latitude''wpt_name''num_private' 'basin''subvillage''region''region_code''district_code''lga''ward''population''public_meeting''recorded_by' 'scheme_management''scheme_name''permit''construction_year''extraction_type''extraction_type_group' 'extraction_type_class''management''management_group''payment''payment_type''water_quality''quality_group' 'quantity''quantity_group''source''source_type''source_class''waterpoint_type''waterpoint_type_group'

In [5]:

```
summary(Train)
nrow(Train)#amount of rows
ncol(Train)#amount of columns
```

```
       id          amount_tsh       date_recorded          funder         
 Min.   :    0   Min.   :     0.0   Length:59400       Length:59400      
 1st Qu.:18520   1st Qu.:     0.0   Class :character   Class :character  
 Median :37062   Median :     0.0   Mode  :character   Mode  :character  
 Mean   :37115   Mean   :   317.7                                        
 3rd Qu.:55657   3rd Qu.:    20.0                                        
 Max.   :74247   Max.   :350000.0                                        
   gps_height      installer           longitude        latitude      
 Min.   : -90.0   Length:59400       Min.   : 0.00   Min.   :-11.649  
 1st Qu.:   0.0   Class :character   1st Qu.:33.09   1st Qu.: -8.541  
 Median : 369.0   Mode  :character   Median :34.91   Median : -5.022  
 Mean   : 668.3                      Mean   :34.08   Mean   : -5.706  
 3rd Qu.:1319.2                      3rd Qu.:37.18   3rd Qu.: -3.326  
 Max.   :2770.0                      Max.   :40.35   Max.   :  0.000  
   wpt_name          num_private           basin            subvillage       
 Length:59400       Min.   :   0.0000   Length:59400       Length:59400      
 Class :character   1st Qu.:   0.0000   Class :character   Class :character  
 Mode  :character   Median :   0.0000   Mode  :character   Mode  :character  
                    Mean   :   0.4741                                        
                    3rd Qu.:   0.0000                                        
                    Max.   :1776.0000                                        
    region           region_code    district_code       lga           
 Length:59400       Min.   : 1.0   Min.   : 0.00   Length:59400      
 Class :character   1st Qu.: 5.0   1st Qu.: 2.00   Class :character  
 Mode  :character   Median :12.0   Median : 3.00   Mode  :character  
                    Mean   :15.3   Mean   : 5.63                     
                    3rd Qu.:17.0   3rd Qu.: 5.00                     
                    Max.   :99.0   Max.   :80.00                     
     ward             population      public_meeting     recorded_by       
 Length:59400       Min.   :    0.0   Length:59400       Length:59400      
 Class :character   1st Qu.:    0.0   Class :character   Class :character  
 Mode  :character   Median :   25.0   Mode  :character   Mode  :character  
                    Mean   :  179.9                                        
                    3rd Qu.:  215.0                                        
                    Max.   :30500.0                                        
 scheme_management  scheme_name           permit          construction_year
 Length:59400       Length:59400       Length:59400       Min.   :   0     
 Class :character   Class :character   Class :character   1st Qu.:   0     
 Mode  :character   Mode  :character   Mode  :character   Median :1986     
                                                          Mean   :1301     
                                                          3rd Qu.:2004     
                                                          Max.   :2013     
 extraction_type    extraction_type_group extraction_type_class
 Length:59400       Length:59400          Length:59400         
 Class :character   Class :character      Class :character     
 Mode  :character   Mode  :character      Mode  :character     



  management        management_group     payment          payment_type      
 Length:59400       Length:59400       Length:59400       Length:59400      
 Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character  
```

```
water_quality        quality_group        quantity           quantity_group
Length:59400         Length:59400         Length:59400       Length:59400
Class :character     Class :character     Class :character   Class :character
Mode  :character     Mode  :character     Mode  :character   Mode  :character
```

```
   source            source_type          source_class       waterpoint_type
Length:59400         Length:59400         Length:59400       Length:59400
Class :character     Class :character     Class :character   Class :character
Mode  :character     Mode  :character     Mode  :character   Mode  :character
```

```
waterpoint_type_group
Length:59400
Class :character
Mode  :character
```

59400

40

Above, we find a small summary of the data, together with the size of the pump_data (59400,40). We have 40 features for almost 60000 datapoints.

2.2 Feature selection

Feature selection is the process of reducing the number of input variables for a predictive model, To enhance the performance of the model and reduce the computational cost.

- **Feature selection methods examine the relationship between each input variable and the target variable using statistics and chose the input variables that have the greatest influence on or relationship with the target variable. Features that are highly correlated to the dependant variable are considered to be highly informative.**
- **Choosing appropriate statistical measures is dependant on the data type of both the input and output variables.**
- **The main types of feature selection techniques are supervised and unsupervised techniques**
    - **We will be working with supervised methods since the data to do so is available.**
        - **wrapper techniques**
        - **filter techniques**
        - **intrinsic or embedded techniques**
- Wrapper methods function as a wrapper around a predictive model (and are thus dependant on the predictive method). They are also quite computationally expensive and prone to overfitting, which is why we will decide against them. Embedded methods select features during the model building phase and are thus also dependant on the predictive model. Filter methods use general characteristics such as correlation

to the dependant variable to filter out features that are not relevant. This works based on a heuristic threshold. It is a very simple and fast method that functions independently of a predictive model and usually works well with large numbers of features. They also avoid overfitting a lot better than wrapper methods. For these reasons, we will be opting for a filter method. One thing we need to keep in mind is that filter methods sometimes fail to include important features in the model.

http://www.datasciencesmachinelearning.com/2019/10/feature-selection-filter-method-wrapper.html (http://www.datasciencesmachinelearning.com/2019/10/feature-selection-filter-method-wrapper.html)

Features To Remove(drop):

The following features will be dropped because we do not bellieve they can influence the model.

- **col 2, amount_tsh:** The static head (amount of water available to waterpoint) For this feature, there are too many zero values.
- **col 9, wpt_name :** It seems to be the name of the water point. But could a name influence the functionality?
- **col 10,num_private:** All values are 0. A feature with no variability can not be an informative one.
- **col 12 13, subvillage & region:** Could be identified with region_code
- **col 17, ward:** It seems like the name of person who guard the pump. Too manyunique values.
- **col 20,recordrd_by:** It does not matter who recorded the data.
- **col 22,scheme_name:** Too many missing values and we think it will not influence the functionality.
- **col 25 26,extraction_type & extraction_type_group:** Similar to col 27: extraction_type_class
- **col 31,payment_type: Similar to col 30:** payment
- **col 35,quantity_group : Similar to col 34:** quantity
- **col 39 40,waterpoint_type waterpoint_type_group:** Type of waterpoint should not influence the result.

In [6]:

```
#Removing unnecessary columns
Train[, c(2,9,10,12,13,17,20,21,22,25,26,31,35,39,40)] <- NULL
```

Some other features To Remove:

In [7]:

```
Train$longitude<-NULL
Train$latitude<-NULL
Train$basin<-NULL
Train$lga<-NULL
Train$management<-NULL
Train$management_group<-NULL
Train$water_quality<-NULL
Train$source<-NULL
Train$source_type<-NULL
```

- **longitude and latitude:** Those variables could help us to find where are the pumps, but cannot provide

any help with the model. We assume the region_code variable gives us enough information about how location may influence the working of the pumps.
- **basin:** We think region_code is enough to show where the pump is.
- **lga:** Too many unique values. Features that are too heterogeneous are usually not informative.
- **management & management_group:** Those columns always have the same values but corresponding to different results.It seems not the **"management"** that will have an effect on the maintenance of the pump.
- **water_quality:** Similar to quality_group.
- **And moreover, we have feature source and source_type, but we think source_class is enough.**

In [8]:

```
Train
```

A data.frame: 59400 × 16

id	date_recorded	funder	gps_height	installer	region_code	district_code	population	public_r
<int>	<chr>	<chr>	<int>	<chr>	<int>	<int>	<int>	
69572	2011-03-14	Roman	1390	Roman	11	5	109	
8776	2013-03-06	Grumeti	1399	GRUMETI	20	2	280	
34310	2013-02-25	Lottery Club	686	World vision	21	4	250	
67743	2013-01-28	Unicef	263	UNICEF	90	63	58	
19728	2011-07-13	Action In A	0	Artisan	18	1	0	
9944	2011-03-13	Mkinga	0	DWE	4	8	1	

2.3 Missing value imputation

Most machine learning libraries produce an error while you try to build a model using data included missing values. Therefore it's needed to choose one strategy to deal with them.

In machine learning with python, to deal with missing values and preparing the data for machine learning, better said, to impute the numerical and categorical data; There are some approaches like using the SimpleImputer library from sklearn.impute which replaces NaN valuse by a specified placeholder, extensive imputer, etc. All of which are available in the built-in the SciKit learn. Of course, these need to be applied for both the training and validation sets. Another way of dealing with these values is by dropping the columns with NaN values. This is clearly suboptimal because it can cause us to lost quite a few datapoints.

Then Scikit-learn provided the Pipeline module to the training data and fit the model in a single line of code. This gives a much better score than imputing the train and validation set manually. Although both of them have the same concept, the pipeline (Python-Sklearn) raises the score and reduces MAE better than manual imputation.

In this project with R,we manually fix missing values of “year” feature by replacing them by the median values in blank spaces.

Missing value imputation

After removing some unrelated features, there are about 20 features left. Few of them have missing values. "construction_year" have many missing values and we think this attribute is related to the functionality of pump, therefore we deal with missing values by fill median values in blank spaces.
And insert a new colume "age" to replace "construction_year" and "date_recorded".

In [9]:

```
install.packages("stringr")
```

```
Warning message:
"package 'stringr' is in use and will not be installed"
```

In [10]:

```
install.packages("tidyverse/stringr")
```

```
Warning message:
"package 'stringr' is in use and will not be installed"
```

In [11]:

```
install.packages("tidyverse")
```

```
Warning message:
"package 'tidyverse' is in use and will not be installed"
```

In [12]:

```
library("stringr")
```

In [13]:

```
#create new col to extract year from date recorded
Train$year_recorded <- str_sub(Train$date_recorded, 1, 4)#library(tidyverse)
#Now use year_recorded and construction_year to find age of the well
Train$year_recorded <- as.numeric(Train$year_recorded)
Train$construction_year <- as.numeric(Train$construction_year)
#before using construction year, deal with its null values
#We consider the possibility of loss of data among the pump constructed in different year is the equ
Train$construction_year[Train$construction_year<1960]= median(Train$construction_year[Train$constru
#medianYr <- median(pump_data$construction_year)
#pump_data$construction_year[pump_data$construction_year == 0] <- medianYr
#How old is the pump
Train$age <- Train$year_recorded - Train$construction_year
Train$age[Train$age < 0] <- 0
#Now we have age and year_recorded, and drop construction year and date recorded
Train$construction_year<-NULL
Train$date_recorded<-NULL
```

In [14]:

```
Train
```

A data.frame: 59400 × 16

id	funder	gps_height	installer	region_code	district_code	population	public_meeting	permit
<int>	<chr>	<int>	<chr>	<int>	<int>	<int>	<chr>	<chr>
69572	Roman	1390	Roman	11	5	109	True	False
8776	Grumeti	1399	GRUMETI	20	2	280		True
34310	Lottery Club	686	World vision	21	4	250	True	True
67743	Unicef	263	UNICEF	90	63	58	True	True
19728	Action In A	0	Artisan	18	1	0	True	True
9944	Mkinga	0	DWE	4	8	1	True	True

2.4 Feature engineering

Feature engineering aims to make the data more compatible to the available problem by using domain knowledge.

- **Enhances the performance of model's prediction**
- **Lowers need of the computational or data**

- **Enhances the interpretability of the results**

We think funder and installer may influence the result.Funder and installer may determine the quality of facilities. But there are tens of different values.
Find top 15 funder and installer and set others as "other".
This is the part of train set data processing. To guarantee the prediction will work, we should make sure the test set have the same levels as the train set.

In [15]:

```
#funder- subsetted the top 15 funder and considered the rest as "other"

Train$funder <- tolower(Train$funder)
Train$funder[Train$funder %in% c(" ", "", "0", "_", "-")] <- "other"
funder.top <- names(summary(as.factor(Train$funder)))[1:15]
Train$funder[!(Train$funder %in% funder.top)] <- "other"
Train$funder <- as.factor(Train$funder)
#installer

Train$installer <- tolower(Train$installer)
Train$installer[Train$installer %in% c(" ", "", "0", "_", "-")] <- "other"
installer.top <- names(summary(as.factor(Train$installer)))[1:15]
Train$installer[!(Train$installer %in% installer.top)] <- "other"
Train$installer <- as.factor(Train$installer)
Train
```

A data.frame: 59400 × 16

id	funder	gps_height	installer	region_code	district_code	population	public_meeting	permit
<int>	<fct>	<int>	<fct>	<int>	<int>	<int>	<chr>	<chr>
69572	other	1390	other	11	5	109	True	False
8776	other	1399	other	20	2	280		True
34310	other	686	world vision	21	4	250	True	True
67743	unicef	263	other	90	63	58	True	True
19728	other	0	other	18	1	0	True	True
9944	other	0	dwe	4	8	1	True	True

Calculation the Correlation

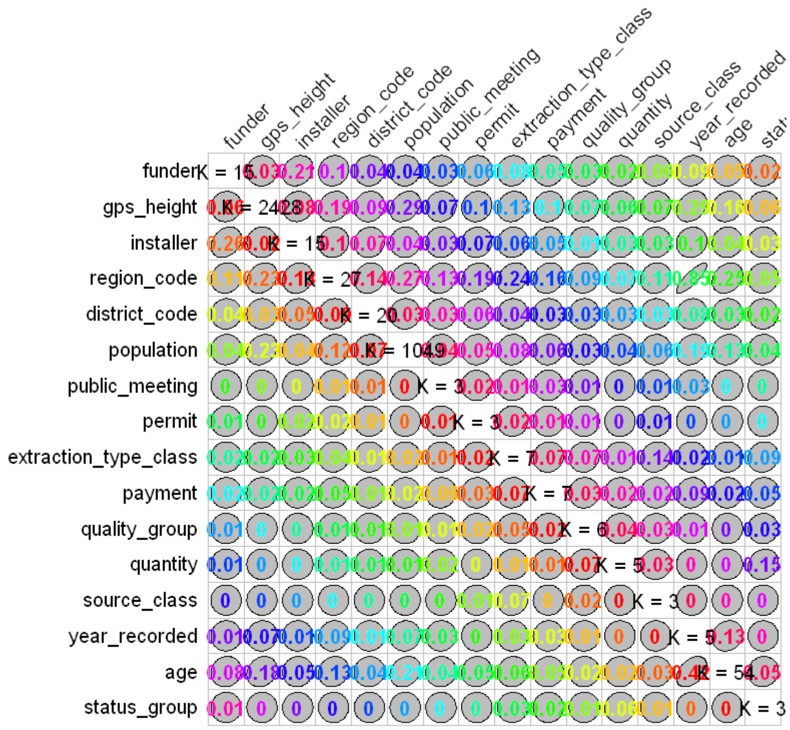
After manually editing several features, we can finally apply a filter method to our data. Association between categorical variables can be measured by the Goodman-Kruskal (GK) statistics. These metrics measure the association between two variables, based on how well you can predict the value of one variable based on the other.

There are a few important measures: GK gamma (γ), GK lambda (λ) and GK tau (τ). We will not further elaborate on GK gamma. Lambda shows the improvement in probability of knowing the dependant variable, given the value of the other variable (in percentage). Tau shows the improvement in predictability of the dependant variable, given the value of the other variable. This is calculated based on random category assignment, with probabilities specified by marginal proportions (and not the probability of the modal category such as with GK lambda). We will focus on the values of GK tau. This value can range from -1 to +1, respectively a perfect negative and positive association. [Source 1 (https://support.minitab.com/en-us/minitab/18/help-and-how-to/statistics/tables/how-to/cross-tabulation-and-chi-square/interpret-the-results/all-statistics-and-graphs/measures-of-association/#goodman-kruskal-lambda-and-tau)](https://support.minitab.com/en-us/minitab/18/help-and-how-to/statistics/tables/how-to/cross-tabulation-and-chi-square/interpret-the-results/all-statistics-and-graphs/measures-of-association/#goodman-kruskal-lambda-and-tau) [Source 2 (https://cran.r-project.org/web/packages/GoodmanKruskal/vignettes/GoodmanKruskal.html)](https://cran.r-project.org/web/packages/GoodmanKruskal/vignettes/GoodmanKruskal.html)

The GKtauDataframe function that we use computes the GK tau score for all pairwise combinations of features.

In [16]:

```
Train <- merge(Train, class_var, by = 'id', all.x = TRUE)
#Before plotting, remove id. ID will not contribute to the model.
Train[,1]<-NULL
#Inspired by:
#https://www.rdocumentation.org/packages/GoodmanKruskal/versions/0.0.3/topics/GKtauDataframe
gkm <- GKtauDataframe(Train)
plot(gkm)
```



Finally Remove those variables with zero association

As we can see in the gkm figure, the public_meeting, permit, source_class and year_recorded features have no correlation with the status. These will be removed.

In [17]:

```
Train[,c(7,8,13,14)]<-NULL
Train
```

A data.frame: 59400 × 12

funder	gps_height	installer	region_code	district_code	population	extraction_type_class	payment
<fct>	<int>	<fct>	<int>	<int>	<int>	<chr>	<chr>
tasaf	0	other	14	3	0	handpump	unknown
other	1978	other	11	4	20	rope pump	never pay
other	0	other	1	4	0	motorpump	pay per bucket
other	1639	ces	3	5	25	gravity	pay per bucket
other	0	other	1	4	0	handpump	unknown
other	28	other	60	43	6922	submersible	pay per bucket

2.5 Dataset:Summary

The status of pump base on different features.

In [18]:

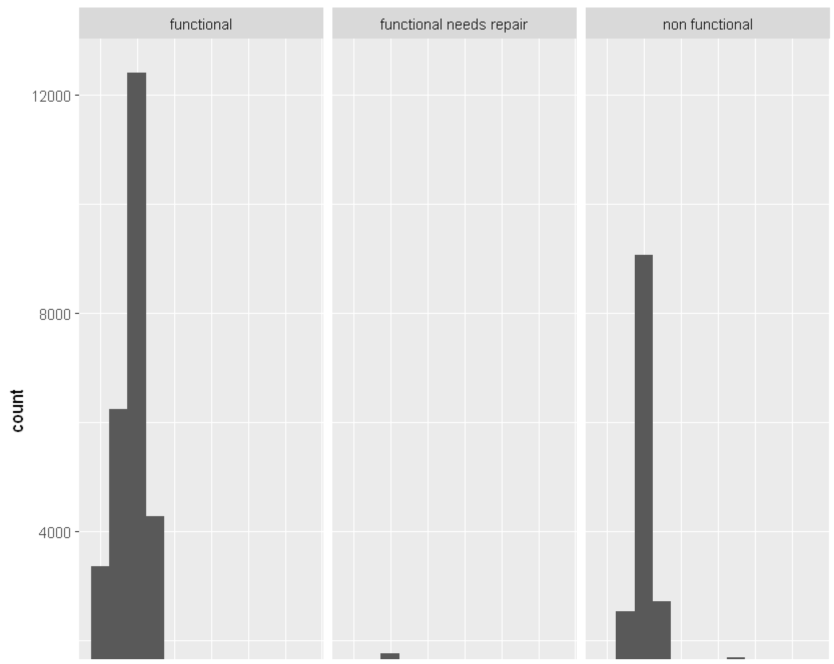
```
#library ggplot to show the status of data
ggplot(subset(Train, age > 0), aes(x = age)) +
  geom_histogram(binwidth = 5) +
  facet_grid( ~ status_group)

ggplot(data=Train, aes(x=extraction_type_class,fill=status_group)) +
  geom_bar()+ggtitle("Status --> Extraction Type class")+scale_fill_manual(values=c("pink","black",'

ggplot(data=Train, aes(x=payment,fill=status_group)) +
  geom_bar()+ggtitle("Status --> Payment")+scale_fill_manual(values=c("cadetblue2","black","cyan4"))

ggplot(data=Train, aes(x=quality_group,fill=status_group)) +
  geom_bar()+ggtitle("Status --> Quality")+scale_fill_manual(values=c("blue4","purple4","red4"))

ggplot(data=Train, aes(x=quantity,fill=status_group)) +
  geom_bar()+ggtitle("Status --> Quantity")+scale_fill_manual(values=c("pink","darkred","yellow"))
```



Splitting the dataset to train and validation set for three models.

There are many different ways of validating a dataset such as train-test split, cross-validation (CV), OOB for bootstrapping techniques, etc. The simplest of these is the train-test split, where the data randomly gets split into a training portion and a testing portion of the dataset to evaluate the model on unseen data. This is commonly done according to a 70/30, 80/20 (or even 90/10) frequency. For fairly large datasets like ours however, a train-test split might offer the best trade-off between computing power and variance reduction. We conclude that an 80/20 split is appropriate for a dataset of our size. If we would decide that our model is overfitting, we can use another validation technique that further reduces variance.

In [41]:

```
#splitting the dataset into training and train data in 80:20 ratio

size<-floor(nrow(Train)*0.8)
index <- sample(1:nrow(Train), size)#Select 80% of the rows randomly with sample() function

pump_train<-Train[index,]
pump_tst<-Train[-index,]
#normalization

pump_train2 <- pump_train %>%
  mutate_if(is.numeric, scale)
pump_tst2 <- pump_tst %>%
  mutate_if(is.numeric, scale)

pump_train3<-pump_train2
pump_tst3<-pump_tst2

pump_tst3
```

A data.frame: 11880 × 12

	funder	gps_height	installer	region_code	district_code	population	extraction_type_class	p
	<fct>	<dbl[,1]>	<fct>	<dbl[,1]>	<dbl[,1]>	<dbl[,1]>	<chr>	
1	tasaf	-0.9632358	other	-0.07106301	-0.26975186	-0.38249412	handpump	u
10	other	-0.9632358	other	0.15939589	0.25616997	-0.38249412	motorpump	
27	kkkt	2.1148380	kkkt	-0.76243972	0.04580124	0.14460125	gravity	
30	other	-0.9632358	other	0.10178117	-0.26975186	-0.38249412	other	u
37	government of tanzania	0.2115020	government	-0.70482500	-0.26975186	0.98795384	submersible	
41	other	0.9632358	other	0.82005445	0.16456749	0.38249412	handpump	

3.Model Selection

The dataset has been divided. We will use the train set to train the model and validation set to check how those three models work.
Prediction will be made on test set and decision will be based on accuracy.

Accuracy

$$A = \frac{T_P + T_N}{N} = \frac{T_P + T_N}{F_P + F_N + T_P + T_N} = \frac{Correct predictions}{Size of dataset.}$$

The accuracy represents the ratio between the number of correctly classified samples (False ...) and the total number of samples.

A note on the accuracy score

Accuracy scores are a good metric to use on balanced data. If there is an imbalance between the frequency of the different categories, we might be better off using the F1-score. A simple example to illustrate this: Imagine our dependant variable can take on the values "yes" and "no". Their respective frequencies are 95% and 5%. It is easy to see how we can make a model that will predict yes all the time and have an accuracy of 95%, while not being a good model at all.

For this reason we will check the distribution of the classes in our dependant variable. We see that there is a mild imbalance. The frequency of our least common category is <15% of our most prevalent category.

In [20]:

```
#table to show the count of each class to look for an imbalance
table(pump_train$status_group)
```

```
             functional functional needs repair          non functional
                  25739                    3445                   18336
```

F1-Score

In imbalanced cases it might be better to use a metric such as F1-score. This score can be calculated as follows:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Here, precision is a measure for how accurately our model predicts a positive. This is useful if the cost of the false positive is high.

$$Precision = \frac{T_P}{T_P + F_P}$$

Recall is a measure for how many of the actual positive values our model captures. This is useful when there is a high cost associated with false negatives.

$$Recall = \frac{T_P}{T_P + F_N}$$

Here we implement functions to calculate F1 Score and Macro F1-Score based on confusion matrix.

In [42]:

```
F1<- function(posTP,posFP,posFN,confusion_matrix,map) {
TP<-sum(confusion_matrix[posTP])
FP<-sum(confusion_matrix[posFP])
FN<-sum(confusion_matrix[posFN])
    FScore<-2*TP/(2*TP+FP+FN)
    return(FScore)
}
getPos<-function(value,Matrix){#Return the position(x,y) of the value in a matrix.
    for (i in 1:nrow(Matrix)){
        for (j in 1:ncol(Matrix)){
         if (Matrix[i,j]==value){
             return(c(i,j))
         }
        }
    }
    return(0)
}
MacroF1<-function(funcPos,nonFuncPos,needRepPos,confusion_matrix){ #Position of TP, for example. 1
    map<- matrix(c(1:9), nrow = 3, byrow = FALSE)
    order<-c(funcPos,nonFuncPos,needRepPos)
    sum<-0
    for (i in 1:3){
        posTP<-order[i]
        location<-getPos(posTP,map)
        row<-location[1]
        col<-location[2]
        SameCol<-map[,col]
        SameRow<-map[row,]
        posFN<-SameCol[-row]#The items in the same column are FN
        posFP<-SameRow[-col]#Items in the same row are FP
        sum=sum+F1(posTP,posFP,posFN,confusion_matrix,map)
    }
    return (sum/3)
}
```

Model 1 :Decision Trees

One of the Supervised Machine Learning algorithms is the Decision Tree uses a set of binary rules to calculate a target value. In this Tree, **the Root** is our first question and it performs the first split, **other nodes(Internal nodes)** in the Decision Tree represent a predictor variable (feature), **the edge between the nodes represents a Decision**, and the **Terminal nodes or leaves** represents an outcome (response variable).

Model 1:Decision Trees(Advantages vs. disadvantages)

Advantages of Decision Tree:

- Easy to understand and interpreted.
- It is used as a classifier (for the categorical target variable) and also as a regressor (for the continuous target variable) so it applied for the classification and regression problems
- Better efficient performance with non-linear data, comparing with other Machine Learning algorithms
- Fast process since it uses only one feature per node to split the data.

Disadvantages of Decision Tree:

- Decision trees are not suitable for difficult decisions.
- A deep tree with lots of leaves will overfit because each prediction is coming from historical data from only a few houses at its leaf.
- A shallow tree with few leaves will perform poorly because it fails to capture as many distinctions in the raw data.
- It faces the tension between underfitting and overfitting.

Reference : [kaggle machine learning (https://www.kaggle.com/dansbecker/random-forests)](https://www.kaggle.com/dansbecker/random-forests)

In [22]:

```
pump_train
```

A data.frame: 47520 × 12

	funder	gps_height	installer	region_code	district_code	population	extraction_type_class	pa
	<fct>	<int>	<fct>	<int>	<int>	<int>	<chr>	
47731	other	0	other	14	5	0	other	
51910	other	1302	other	13	1	145	gravity	
52988	other	379	other	5	6	1	submersible	p
21874	other	1350	other	16	1	1	submersible	un
35043	other	1669	other	13	2	1	submersible	un
17633	government of tanzania	1088	community	2	7	130	gravity	un

In [23]:

```
pump_tst
```

A data.frame: 11880 × 12

	funder	gps_height	installer	region_code	district_code	population	extraction_type_class	pa
	<fct>	<int>	<fct>	<int>	<int>	<int>	<chr>	
10	other	0	other	18	8	0	motorpump	
11	other	64	other	4	5	150	submersible	p
16	other	0	commu	1	3	0	motorpump	p
17	other	226	other	99	1	450	motorpump	p
19	government of tanzania	0	other	1	4	0	motorpump	un
20	other	76	other	7	2	250	other	p

Train the decision tree model.

In [24]:

```
start_time <- Sys.time()
dt_model<-rpart(status_group~.,data=pump_train)
dt_time <- Sys.time() - start_time
dt_time
dt_model
```

```
Time difference of 1.031241 secs

n= 47520

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 47520 21781 functional (0.541645623 0.072495791 0.385858586)
  2) quantity=enough,insufficient,seasonal 41856 16422 functional (0.607654817 0.081
350344 0.310994839)
    4) extraction_type_class=gravity,handpump,motorpump,rope pump,submersible,wind-p
owered 38026 13390 functional (0.647872508 0.085388944 0.266738547)
      8) age< 33.5 35316 11645 functional (0.670262770 0.084777438 0.244959792) *
      9) age>=33.5 2710  1218 non functional (0.356088561 0.093357934 0.550553506) *
    5) extraction_type_class=other 3830   956 non functional (0.208355091 0.04125326
4 0.750391645) *
  3) quantity=dry,unknown 5664   345 non functional (0.053848870 0.007062147 0.93908
8983) *
```

Importance of variables in decision trees:

In [25]:

```
data.frame(dt_model$variable.importance)
```

A data.frame: 5 × 1

	dt_model.variable.importance
	<dbl>
quantity	3525.76332
extraction_type_class	1492.87472
quality_group	654.85576
age	494.23986
installer	75.94335

Make predictions

In [43]:

```
predict_dt <- predict(dt_model, pump_tst)
predict_dt
```

A matrix: 11880 × 3 of type dbl

	functional	functional needs repair	non functional
1	0.67026277	0.084777438	0.2449598
10	0.67026277	0.084777438	0.2449598
27	0.67026277	0.084777438	0.2449598
30	0.20835509	0.041253264	0.7503916
37	0.67026277	0.084777438	0.2449598
41	0.67026277	0.084777438	0.2449598
44	0.67026277	0.084777438	0.2449598
53	0.67026277	0.084777438	0.2449598
58	0.67026277	0.084777438	0.2449598
67	0.67026277	0.084777438	0.2449598
69	0.20835509	0.041253264	0.7503916
70	0.67026277	0.084777438	0.2449598
71	0.67026277	0.084777438	0.2449598
74	0.67026277	0.084777438	0.2449598
76	0.67026277	0.084777438	0.2449598
85	0.67026277	0.084777438	0.2449598
94	0.67026277	0.084777438	0.2449598
99	0.67026277	0.084777438	0.2449598
102	0.67026277	0.084777438	0.2449598
104	0.67026277	0.084777438	0.2449598
111	0.05384887	0.007062147	0.9390890
112	0.67026277	0.084777438	0.2449598
113	0.67026277	0.084777438	0.2449598
114	0.05384887	0.007062147	0.9390890
119	0.67026277	0.084777438	0.2449598
120	0.67026277	0.084777438	0.2449598
130	0.67026277	0.084777438	0.2449598
133	0.67026277	0.084777438	0.2449598
137	0.67026277	0.084777438	0.2449598
142	0.20835509	0.041253264	0.7503916
...	...	...	...
59280	0.67026277	0.084777438	0.2449598
59283	0.20835509	0.041253264	0.7503916
59296	0.20835509	0.041253264	0.7503916

	functional	functional needs repair	non functional
59302	0.67026277	0.084777438	0.2449598
59304	0.67026277	0.084777438	0.2449598
59305	0.67026277	0.084777438	0.2449598
59309	0.05384887	0.007062147	0.9390890
59311	0.05384887	0.007062147	0.9390890
59312	0.67026277	0.084777438	0.2449598
59314	0.67026277	0.084777438	0.2449598
59321	0.67026277	0.084777438	0.2449598
59323	0.67026277	0.084777438	0.2449598
59326	0.67026277	0.084777438	0.2449598
59330	0.67026277	0.084777438	0.2449598
59332	0.67026277	0.084777438	0.2449598
59338	0.67026277	0.084777438	0.2449598
59342	0.05384887	0.007062147	0.9390890
59346	0.35608856	0.093357934	0.5505535
59349	0.20835509	0.041253264	0.7503916
59357	0.67026277	0.084777438	0.2449598
59361	0.67026277	0.084777438	0.2449598
59365	0.67026277	0.084777438	0.2449598
59367	0.67026277	0.084777438	0.2449598
59373	0.35608856	0.093357934	0.5505535
59376	0.67026277	0.084777438	0.2449598
59381	0.67026277	0.084777438	0.2449598
59387	0.67026277	0.084777438	0.2449598
59392	0.67026277	0.084777438	0.2449598
59393	0.05384887	0.007062147	0.9390890
59398	0.67026277	0.084777438	0.2449598

In [44]:

```
Y_pred<-c(1:nrow(predict_dt))
for (i in 1:nrow(predict_dt)){
if (predict_dt[i,1]>0.6){
Y_pred[[i]]<-"functional"
}
}
for (i in 1:nrow(predict_dt)){
if(predict_dt[i,3]>0.75){Y_pred[i]<-"non functional"
}}
for(i in 1:length(Y_pred)){
if(Y_pred[i]!="functional"&&Y_pred[i]!="non functional"){
    Y_pred[i]<-"functional needs repair"
}
}
#Y_pred
```

In [45]:

```
tstSet<-pump_tst$status_group
confusion_matrix1 <- table(Y_pred,tstSet)
confusion_matrix1
```

```
                         tstSet
Y_pred                    functional functional needs repair non functional
  functional                    5927                     743           2117
  functional needs repair        231                      71            404
  non functional                 268                      52           2067
```

In [46]:

```
size<-length(Y_pred)

t<-confusion_matrix1[c(1,5,9)]
accuracy<-sum(t)/size

sprintf('The accuracy in total for decision tree: %f ',accuracy)

#TP<-sum(confusion_matrix1[1])
#TN<-sum(confusion_matrix1[c(5,6,8,9)])
#FP<-sum(confusion_matrix1[c(4,7)])
#FN<-sum(confusion_matrix1[c(2,3)])
#FScore=2*TP/(2*TP+FP+FN)
#sprintf('If we divide the data to two classes, then the FScore for decision tree: %f ',FScore)
MacroScore<-MacroF1(1,5,9,confusion_matrix1)
sprintf('The Macro F1 for decision tree: %f ',MacroScore)
```

'The accuracy in total for decision tree: 0.678872 '

'The Macro F1 for decision tree: 0.487407 '

Model 2: KNN(K Nearest Neighbor)

A Supervised Machine Learning algorithm and a classifier that determines a new data point, **based on the**

features of its neighborhood, belongs to which **target class**.

- **The main concept relies on feature similarity. KNN examines the similarity of a data point to its neighbors and classifies the data point into the class where there are the most similarities. (KNN algorithm can be used for regression problems as well.)**
- **The input is the labeled data set to predict the output of the data points**
- **A non-parametric model which uses a flexible number of parameters to build the model.**
- **KNN is a lazy algorithm, lazy models have less training time but more time in predicting while it memorizes the training data set instead of learning a discriminative function from the training data.**
- **Easy implementation**

In [47]:

```
#library("lazy")
#Those features need to convert to factor or otherwise error will occur when training (and predictin
pump_train2$status_group<-factor(pump_train2$status_group,levels=c('functional','non functional','f
pump_train2$quantity<-factor(pump_train2$quantity,levels=c('dry','unknown','insufficient','enough',
pump_train2$quality_group<-factor(pump_train2$quality_group,levels=c('colored','fluoride','good','m
pump_train2$payment<-factor(pump_train2$payment,levels=c('never pay','unknown','pay annually','pay 
pump_train2$extraction_type_class<-factor(pump_train2$extraction_type_class,levels=c('gravity','han
pump_train2$funder<-factor(pump_train2$funder,levels=c('danida','dhv','district council','dwsp','go

pump_tst2$status_group<-factor(pump_tst2$status_group,levels=c('functional','non functional','funct
pump_tst2$quantity<-factor(pump_tst2$quantity,levels=c('dry','unknown','insufficient','enough','sea
pump_tst2$quality_group<-factor(pump_tst2$quality_group,levels=c('colored','fluoride','good','milky
pump_tst2$payment<-factor(pump_tst2$payment,levels=c('never pay','unknown','pay annually','pay mont
pump_tst2$extraction_type_class<-factor(pump_tst2$extraction_type_class,levels=c('gravity','handpum
pump_tst2$funder<-factor(pump_tst2$funder,levels=c('danida','dhv','district council','dwsp','govern
tstStatus<-pump_tst2$status_group
pump_train2$status_group<-factor(pump_train2$status_group,levels=c('functional','non functional','f
for (i in 1:ncol(pump_train2)){
    pump_train2[,i]<-as.numeric(pump_train2[,i])
    pump_tst2[,i]<-as.numeric(pump_tst2[,i])
}
pump_train2
```

A data.frame: 47520 × 12

	funder	gps_height	installer	region_code	district_code	population	extraction_type_cl
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<c
48722	1	1.76544097	5	-0.24442946	-0.06684327	-0.29426374	
21864	8	1.75967567	12	-0.01773330	-0.17030882	2.25257351	
58367	5	1.29412821	9	-0.30110349	-0.27377436	-0.23691211	
26866	9	0.12377359	8	-0.01773330	-0.27377436	-0.37922912	
19665	5	1.57806892	8	-0.75449581	-0.37723990	-0.06273308	
39900	2	-0.53346989	13	-0.58447369	-0.27377436	0.25588709	
55464	9	-0.75831635	13	-0.58447369	-0.37723990	-0.02025040	
47202	9	-0.96442560	8	0.20896286	-0.37723990	-0.38135326	
3101	1	0.95253456	6	-0.30110349	-0.27377436	-0.27514653	
22600	9	1.12693469	8	-0.75449581	-0.37723990	0.04347364	
30816	5	0.06756197	7	-0.69782177	-0.37723990	-0.16893981	
16064	9	1.27539100	12	0.32231094	-0.48070545	0.16879757	
43971	9	1.45988040	12	-0.64114773	-0.48070545	-0.37922912	
39276	11	-0.96442560	8	0.09561478	-0.27377436	-0.38135326	
30080	9	0.73777737	12	0.03894074	-0.37723990	2.80484848	
4018	9	-0.08089434	12	-0.58447369	-0.48070545	-0.06273308	
31625	9	-0.96442560	8	-0.81116985	-0.48070545	-0.38135326	
56223	4	-0.96442560	8	0.09561478	-0.48070545	-0.38135326	
33345	9	1.16008513	12	0.26563690	-0.37723990	0.89312744	

	funder	gps_height	installer	region_code	district_code	population	extraction_type_cl
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<c
49491	5	-0.96442560	9	0.09561478	-0.37723990	-0.38135326	
7100	9	0.45527798	8	-0.30110349	-0.37723990	-0.38135326	
24855	14	-0.96442560	12	-0.58447369	-0.48070545	0.08808046	
16968	5	-0.07224640	8	0.32231094	-0.17030882	-0.06273308	
25173	9	0.83722868	12	-0.13108138	-0.48070545	0.43643852	
16542	5	-0.96442560	8	0.15228882	-0.48070545	-0.38135326	
18126	12	-0.96442560	12	-0.18775542	0.03662227	-0.38135326	
1017	9	0.55617062	12	-0.75449581	-0.48070545	-0.37922912	
17205	9	1.68472686	8	-0.75449581	-0.37723990	0.04347364	
35161	9	-0.96442560	12	0.09561478	-0.48070545	-0.38135326	
31474	5	-0.96442560	7	-0.18775542	-0.37723990	-0.38135326	
...	...	...	...	...	...	...	
21726	9	0.79687163	14	0.03894074	-0.48070545	0.68071399	
39686	9	-0.96442560	8	-0.64114773	0.24355336	-0.31762922	
32906	5	0.08197521	9	-0.69782177	-0.17030882	-0.37922912	
9318	9	1.32439600	12	-0.75449581	0.03662227	1.21174761	
9942	9	0.56914253	12	-0.01773330	-0.48070545	2.80484848	
58305	9	0.50860694	12	0.32231094	-0.06684327	0.46830054	
33961	5	1.12405204	8	0.03894074	-0.48070545	0.12843902	
17161	6	-0.96442560	10	0.15228882	-0.37723990	-0.38135326	
35481	9	-0.96442560	12	0.20896286	0.03662227	-0.38135326	
5512	2	-0.54932445	8	-0.58447369	-0.27377436	0.25588709	
19002	13	-0.95433633	12	2.53259848	4.89950287	-0.37922912	
17052	9	-0.90965531	8	-0.52779965	-0.37723990	-0.37922912	
32942	9	-0.96442560	12	-0.81116985	-0.17030882	-0.38135326	
13384	9	1.55933172	8	-0.01773330	-0.27377436	0.57450726	
11538	9	-0.96442560	12	-0.07440734	-0.17030882	-0.38135326	
59030	9	-0.96442560	8	0.09561478	-0.06684327	-0.38135326	
29187	2	-0.53346989	8	-0.58447369	-0.17030882	0.17729411	
26778	9	0.85596589	12	-0.58447369	-0.48070545	0.14968036	
54440	9	1.25809512	12	0.26563690	-0.37723990	0.04347364	
15846	6	-0.96442560	10	0.15228882	0.24355336	-0.38135326	
6124	9	-0.96442560	3	-0.81116985	-0.06684327	-0.38135326	
34991	1	0.88911633	14	0.03894074	-0.48070545	0.68071399	
21910	6	1.10387352	10	0.26563690	-0.37723990	0.46830054	
55131	9	1.24224056	12	-0.13108138	-0.37723990	-0.37922912	
15445	6	-0.96442560	10	0.20896286	0.14008782	-0.38135326	
23773	9	-0.96442560	3	-0.81116985	-0.06684327	-0.38135326	

	funder	gps_height	installer	region_code	district_code	population	extraction_type_cl
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<c
26264	9	-0.96442560	12	-0.64114773	-0.37723990	-0.37922912	
7085	1	1.10819749	5	-0.24442946	0.14008782	-0.27514653	
45434	14	-0.96442560	12	-0.81116985	0.03662227	-0.38135326	
35687	9	-0.96442560	8	0.09561478	-0.48070545	-0.38135326	

In [48]:

```
start_time <- Sys.time()
lz_model<-lazy(status_group~.,data=pump_train2)
lz_time <- Sys.time() - start_time
lz_time
lz_model
```

```
Time difference of 0.01595712 secs

Call:
lazy(formula = status_group ~ ., data = pump_train2)

Number of observations: 47520
```

The output is the prediction on possibility that this object belongs to this class.

In [49]:

```
lz_pred<-predict(lz_model, pump_tst2)
lz_pred
```

$h =

1.72935576419117 1.87955383726914 1.44003597276575 1.13323515889551
1.46999770685489 1.12116769653466 1.37740928866377 1.34431782960821
1.06780915838982 1.45028111214201 1.95965798601431 2.07893438942719
0.993574490778691 2.52636758066599 1.12369535758264 1.05523427778035
1.14270913780912 1.03466600724763 1.26160694633013 1.06727824022953
1.99993340176878 0.93447225746289 1.00253926941506 1.99966538971159
1.29330284721225 1.64527274659801 1.92367526176841 1.3611803873287
1.62394860199191 1.65160206473952 1.25693896449089 1.99971997576692
1.9999996928378 0.999895892519251 1.78909922444156 1.04489250428155
1.40452640173666 1.44641018582501 1.06743984139894 1.30452591915486
1.89983057724106 1.9995252866385 1.7458079801392 1.96005603146945
1.04331822028734 1.06434635502223 0.987769920080457 1.16457022077373
2.20081177238762 1.19172239442907 1.23879653691921 1.99997508430438
1.91943307811313 0.840695766648422 0.999830864019085 0.969974840644625
1.05332927684418 1.72739136553045 1.23766334575035 1.23697101004341
1.59410578431052 1.88795906900022 1.99955414166947 1.33164789007948
0.999966712280607 1.15225609601292 1.18647286615609 1.78909922444156
1.40029199722494 2.16304670265422 0.999999642801523 1.84850212275651
1.07755130840013 2.10587975349889 1.49021041741806 1.98371633368377
2.2691216295717 1.95660532351487 2.97148749312019 2.44980546869524
1.68667052503416 1.15805420134941 1.97114084080186 1.99999948764773
1.12663010240668 1.65497881953001 1.55947723738168 0.99395796365354
1.07452569885132 2.09130449284611 1.78909922444156 1.79577986559126
2.08729767892151 1.46074565116952 0.999999808692061 1.73261039751296
2.17259016877702 1.66568422506436 1.19711359666046 0.999862445013486
1.95172210551891 1.39843830993919 1.67899368849092 2.03478874162328
1.99995593000648 1.45370377876444 1.2165177910929 0.724526599997424
1.4052754856586 1.40876606009231 0.990941824507032 2.67367006995793
1.60769774458774 1.59894533372825 0.790916741995973 2.43707993527004
2.05968824434397 2.3179925691735 1.62157994803243 1.53309223424206
0.880390183901511 1.0099356884374 1.87981423869903 1.26118872769779
1.03570865423124 1.08447245463902 2.00209729773433 1.9999750675992
1.28043520085182 0.968014028731403 1.80030635659827 1.34880646415218
1.46167771941528 1.26009511663087 1.2707402215361 1.9578581980354
2.03298846801662 1.14270913780912 1.9708579709093 1.61950376178577
1.4232752307707 1.79914285177486 1.4345371790532 1.97809537082848
1.40372130549114 1.2051106561029 1.10550252395534 2.01157251820661
1.47728403174683 1.18885501285037 1.35841830214428 1.17553037690149
1.52632829706374 0.940160007561464 1.60947477678781 2.25939170906177
1.99999987738267 1.89604211073783 3.00015115224844 1.21534491739899
1.47512123463934 1.08457108016824 1.00748361986233 1.17750286893499
1.15264333205678 1.82847785443635 1.9816858342974 2.25322441287486
1.14927616823076 1.83909654715169 0.99999767285932 -1.02638895309395
1.39793919982255 1.96195590974046 1.2165177910929 1.56639288829294
1.38751066029307 1.33220648257194 1.47857434834334 1.9999302917715

```
2.07344853901601 1.78311313305327 1.50412816441295 2.02268997077528
1.74141209819944 1.00893252952874 1.05616892716362 2.02311325455102
0.96699538408429 1.26386846511758 1.52064553937602 1.56639288829294
2.13175753072464 1.00651351417748 1.29830129705195 1.55116309117434
2.06788715331272 2.04923760023295 1.23796775453698 0.996762955761114 ...
1.2873765173738 1.52600432750912 1.92129178777396 1.04088015426281
1.41019813139729 2.09828154602815 1.25368491066871 1.243500473493
1.78497596122695 1.99978660323617 1.20210445901492 0.877902887337168
1.47026806278276 1.57823108966277 2.12672488149275 0.999999631689731
1.46474841011577 1.00103907594465 0.999830861819071 1.0645083740969
1.22210505351911 1.13773181004976 1.22369175148756 1.8928334870405
1.79587441177834 0.999999684450156 0.819638502692076 2.04794825635989
1.77241747407898 1.08313174279172 1.7325592647395 1.00399902233216
1.40926360597094 0.99988257009941 1.97507574512654 1.78909922444156
0.999599060976076 1.001001512984 2.16609542213802 2.09421287079058
1.60660575038744 1.9996694676407 1.37813206067852 0.999966586666586
1.1164403287273 0.999818515018843 1.89419369820187 1.37126113537748
2.49500772039447 2.27625678782731 1.14498463829505 0.999879840768559
2.70524646860969 1.43298206231757 1.99978660323617 1.95558749927889
1.46979858059622 1.28178300480101 1.27987733715556 1.71291088561227
1.19098733061785 1.99955414166947 1.35554013812433 1.45015001206407
2.0342977812741 1.1776150655501 1.49936473902872 1.91872396741398
1.32061132061735 1.19319990684272 1.57990614184046 1.99999419991624
1.98276612738777 1.00050254298587 1.26888877711117 1.42448234551117
0.927337623860383 1.08313174279172 1.99996724052693 1.99831686065952
1.40926360597094 1.95566813111127 1.4495504412355 1.47134011201299
1.55556448778896 2.33113967734376 1.9999943206022 1.12997952515657
0.999893301618083 1.39358063739049 1.9999998999944 0.0909773245244809
1.04087792165367 1.99978660323617 1.86598235560919 1.09365744284852
1.99999972807453 1.11015082550263 1.75238786520721 1.09563400026987
1.17030074723956 2.42426791830183 0.999999625226865 1.52481168518593
1.13664664465609 1.26634014280035 1.16006191711028 2.09369409931226
1.40774620884348 1.55383259658971 1.78866142822312 1.2433663263442
0.991274251764088 1.41525473336463 1.53370464470607 2.13494110807583
1.31141806656107 1.15673582304658 1.16234609556874 2.11629817841951
0.917341880017081 1.61514779074727 1.83831345476608 1.08019400361748
1.900050390277 1.52491891639981 0.997279350263371 0.959113357015198
0.976341846245599 2.1064346133871 1.5546645959499 1.0064060015417
1.25877509777296 1.56115141488026 2.9971534859391 1.97409505907468
1.41538917985842 2.4568920956705 1.41698417602483 1.07628758556621
2.05257527410599 1.10612156224763 0.729229273286676 1.14442941849469
0.99976264241387 1.27951666700249 2.21827546783699 0.914210852890751
1.60947477678781 1.98135262809898 1.136119962111 1.99972218190336
1.08882235554892 1.27987733715556 1.05867130551985 1.40926360597094
1.16824752032856 1.6372233414891 1.1100045285049 1.99996718863163
1.04279145214492 1.97823022408063 1.17750286893499 1.99464985850928
0.93345301579829 1.22460728401654 1.99977180924529 1.98521116277268
1.05144421939941 1.31944462503232 1.85345960991685 1.64376076200512
1.0086362460645 1.10677467912302 0.999999625334603 2.99972718700454
1.97577441806201 1.98276620629939 1.20889655506096 2.47500429216328
1.02637281315733 0.976210432276005 1.33220648257194 1.31879864949671
1.8891092712511 1.70066829588674 1.99993030557413 1.69041417834811
```

1.97877583960993 1.56284078478868 2.42747050595664 1.53739557428878
1.02438441166324 1.45318761417912 1.37674436011776 1.81525163689543
1.12997952515657 0.999832669423914 1.99978660323617 1.99966976999161

Set a threshold as a criterion of classification.

In [50]:

```
Y_pred_lz<-c(1:lengths(lz_pred))

for (i in 1:lengths(lz_pred)){
    if(lz_pred$h[i]>2.20){
        Y_pred_lz[i]<-"functional needs repair"
    }else if(lz_pred$h[i]<=1.55){
        Y_pred_lz[i]<-"functional"
    }else{
        Y_pred_lz[i]<-"non functional"
    }
}

confusion_matrix2 <- table(Y_pred_lz,tstStatus)
confusion_matrix2
summary(as.factor(Y_pred_lz))
summary(as.factor(tstStatus))
```

```
                         tstStatus
Y_pred_lz                 functional non functional functional needs repair
  functional                    5046           1128                     296
  functional needs repair        130            212                     187
  non functional                1250           3248                     383
```

functional
6470
functional needs repair
529
non functional
4881

functional
6426
non functional
4588
functional needs repair
866

In [51]:

```
size<-length(Y_pred_lz)
t<-confusion_matrix2[c(1,6,8)]
accuracy<-sum(t)/size
sprintf('The accuracy in total for nearest neighbor: %f ',accuracy)
TP<-sum(confusion_matrix2[1])
FP<-sum(confusion_matrix2[c(4,7)])
FN<-sum(confusion_matrix2[c(2,3)])
FScore2=2*TP/(2*TP+FP+FN)
sprintf('If we divide the data into two classes, then the FScore for nearest neighbor: %f ',FScore2)
MacroScore<-MacroF1(1,6,8,confusion_matrix2)
sprintf('The Macro F1 for KNN: %f ',MacroScore)
```

'The accuracy in total for nearest neighbor: 0.713889 '

'If we divide the data into two classes, then the FScore for nearest neighbor: 0.782568 '

'The Macro F1 for KNN: 0.578899 '

Model 3: Random Forest

One of the supervised learning algorithms is the Random Forest approach.

- It constructs various decision trees called forest and bind them together to provide a more accurate and stable prediction.
- The random forest approach looks like the ensemble technique called as Bagging.
  - We refer to the random forest method as an "ensemble method".
  - The ensemble methods combine the predictions of several models (e.g., several trees, in the case of random forests).
- Here multiple trees are produced by bootstrap samples from training data and then the model reduces the correlation between the trees.
- Applying this approach enhances the performance of decision trees and to some extent prevents overriding.

Random Forest specifications:

It's compound of many decision trees: A random forest is a **set of decision trees** and therefore **<u>it doesn't based on a single feature and combines multiple predictions from each decision tree.</u>**

Advantages of Random Forest:

- **Avoids overfitting**: Each tree, in the multiple decision trees, draws a random sample of data providing the random forest more randomness to produce much better accuracy than decision trees.

- **Efficiency**: Run on the large databases, Random forests are much more efficient comparing with decision trees.
- **Accuracy**: Random forests are set of the decision trees and each decision tree draws sample random data and therefore, random forests give us higher accuracy on prediction.
- **Estimator of the test error**: Efficient use of all features that contribute to the prediction and keep accuracy even if the data is missing.

Random Forest's drawbacks:

- Random Forest is the set of decision trees, so **it needs the various number of levels and biased prediction of the training model** to be much accurate.
- **High memory consumption**: Training a big collection of trees might ask for a high need of memory.

In [53]:

```
pump_train3$status_group<-factor(pump_train3$status_group,levels=c('functional','functional needs r
pump_train3$quantity<-factor(pump_train3$quantity,levels=c('dry','unknown','insufficient','enough',
pump_train3$quality_group<-factor(pump_train3$quality_group,levels=c('colored','fluoride','good','m
pump_train3$payment<-factor(pump_train3$payment,levels=c('never pay','unknown','pay annually','pay 
pump_train3$extraction_type_class<-factor(pump_train3$extraction_type_class,levels=c('gravity','han
pump_train3$funder<-factor(pump_train3$funder,levels=c('danida','dhv','district council','dwsp','go
pump_tst3$status_group<-factor(pump_tst3$status_group,levels=c('functional','functional needs repai
pump_tst3$quantity<-factor(pump_tst3$quantity,levels=c('dry','unknown','insufficient','enough','sea
pump_tst3$quality_group<-factor(pump_tst3$quality_group,levels=c('colored','fluoride','good','milky
pump_tst3$payment<-factor(pump_tst3$payment,levels=c('never pay','unknown','pay annually','pay mont
pump_tst3$extraction_type_class<-factor(pump_tst3$extraction_type_class,levels=c('gravity','handpum
pump_tst3$funder<-factor(pump_tst3$funder,levels=c('danida','dhv','district council','dwsp','govern
```

Train random forest.

In [54]:

```
start_time <- Sys.time()
rf_model<-randomForest(status_group~.,data=pump_train3)
rf_time <- Sys.time() - start_time
rf_time
rf_model
```

```
Time difference of 2.530446 mins


Call:
 randomForest(formula = status_group ~ ., data = pump_train3)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 21.01%
Confusion matrix:
                        functional functional needs repair non functional
functional                   23317                     429           2087
functional needs repair       2158                     839            454
non functional                4666                     191          13379
                        class.error
functional               0.09739481
functional needs repair  0.75688206
non functional           0.26634130
```

Prediction

In [55]:

```
predict_rf <- predict(rf_model, newdata = pump_tst3)
confusion_matrix3 <- table(predict_rf,pump_tst3$status_group)
confusion_matrix3
```

```
predict_rf                functional functional needs repair non functional
  functional                    5822                     526           1224
  functional needs repair        100                     223             48
  non functional                 504                     117           3316
```

In [56]:

```
pump_tst3
```

A data.frame: 11880 × 12

	funder	gps_height	installer	region_code	district_code	population	extraction_type_class
	<fct>	<dbl[,1]>	<fct>	<dbl[,1]>	<dbl[,1]>	<dbl[,1]>	<fct>
1	tasaf	-0.9632358	other	-0.07106301	-0.26975186	-0.38249412	handpump
10	other	-0.9632358	other	0.15939589	0.25616997	-0.38249412	motorpump
27	kkkt	2.1148380	kkkt	-0.76243972	0.04580124	0.14460125	gravity
30	other	-0.9632358	other	0.10178117	-0.26975186	-0.38249412	other
37	government of tanzania	0.2115020	government	-0.70482500	-0.26975186	0.98795384	submersible
41	other	0.9632358	other	0.82005445	0.16456749	0.38249412	handpump

In [57]:

```
size<-length(predict_rf)
t<-confusion_matrix3[c(1,5,9)]
accuracy<-sum(t)/size
sprintf('The accuracy in total for random forest: %f ',accuracy)
TP<-sum(confusion_matrix3[1])
FP<-sum(confusion_matrix3[c(4,7)])
FN<-sum(confusion_matrix3[c(2,3)])
FScore3=2*TP/(2*TP+FP+FN)
sprintf('If we divide the data into two classes, then the FScore for random forest: %f ',FScore3)
MacroScore<-MacroF1(1,5,9,confusion_matrix3)
sprintf('The Macro F1 for random forest: %f ',MacroScore)
```

'The accuracy in total for random forest: 0.787963 '

'If we divide the data into two classes, then the FScore for random forest: 0.831833 '

'The Macro F1 for random forest: 0.656777 '

In [58]:

```
sprintf('The accuracy of "functional" : %f ',confusion_matrix3[1]/sum(confusion_matrix3[c(1,4,7)]))
sprintf('The accuracy of "functional nees repair": %f ',confusion_matrix3[5]/sum(confusion_matrix3[c
sprintf('The accuracy of "non functional" : %f ',confusion_matrix3[9]/sum(confusion_matrix3[c(3,6,9)
```

'The accuracy of "functional" : 0.768885 '

'The accuracy of "functional nees repair": 0.601078 '

'The accuracy of "non functional" : 0.842266 '

Summary: Comparasion on three different models.

	Decision Tree	Nearest Neighbor	Random Forest
Classification Accuracy	0.670286	0.716751	0.787710
F-Score	0.782278	0.787944	0.833603
Time	1.029108 secs	0.595443 secs	2.525385 mins
MacroF1	0.429795	0.570179	0.640276

According to the requirement of the project, **random forest** will be chosen to be the model for prediction for it has a **higher accuracy**.
We prefer to choose some other standards to judge the performance of a model. If we consider "functional needs repair" as a kind of "non functional", then we could calculate the model's F-score. (Or otherwise it is hard to define "True" and "False" for we have three classes.)
Training random forest take much longer time than other two models, but to attain higher accuracy random forest seems to be a better choice.
The reason why we got macro F1 scores that are not high, is because of the score is average of three F1 scores, when we seperately consider three classes as ture positive. And the success rate of prediction on "functional needs repair" is much lower than other two classes. And as a result it affects the mean value.

Alternative models :XGBoost

XGBoost (Extreme Gradient Boosting) is an a gradient boosting method, based on decision trees. This ensemble method is similar to random forest in its building block, but differs considerably in the way the learning process happens. The random forest algorithm trains every tree in parallel, while in gradient boosting methods, the trees are trained sequentially. Every tree can thus learn from the error in the previous tree, but could possibly also exemplify noise. Due to the sequential nature of XGBoost it is not scalable, unlike the random forest method.

Considering well-tuned parameters and a dataset with little noise, gradient boosting algorithms could potentially outperform the random forest algorithm.

In [59]:

```
library(xgboost)
library(Matrix)
library(MatrixModels)
library(data.table)
pump_train_alt<- pump_train3
pump_tst_alt<-pump_tst3
for (i in 1:12){
   pump_train_alt[,i]<-as.numeric(pump_train_alt[,i])
   pump_tst_alt[,i]<-as.numeric(pump_tst_alt[,i])
}
#col 12 is the Label
trainLabel<-pump_train_alt[12]
pump_train_alt<-pump_train_alt[,-12]
valLabel<-pump_tst_alt[12]
pump_tst_alt<-pump_tst_alt[,-12]

pump_train_alt
```

```
Warning message:
"package 'xgboost' was built under R version 4.0.5"

Attaching package: 'xgboost'


The following object is masked from 'package:dplyr':

    slice



Attaching package: 'Matrix'


The following objects are masked from 'package:tidyr':

    expand, pack, unpack
```

In [60]:

```
pump_train_alt<-xgb.DMatrix(data=as.matrix(pump_train_alt),label=as.matrix(trainLabel))
pump_tst_alt<-xgb.DMatrix(data=as.matrix(pump_tst_alt))
```

In [61]:

```
pump_train_alt
```

```
xgb.DMatrix  dim: 47520 x 11  info: label  colnames: yes
```

In [62]:

```
start_time <- Sys.time()
xgb <- xgboost(data = pump_train_alt,max_depth=6, eta=0.5,  objective="multi:softmax", booster = "g
                 evaluation = "merror", eta = .2, max_depth = 12, colsample_bytree = .4)
xg_time <- Sys.time() - start_time
```

```
Warning message in check.booster.params(params, ...):
"The following parameters were provided multiple times:
        eta, max_depth
  Only the last value for each of them will be used.
"
```

In [64]:

```
predict <- predict(xgb,pump_tst_alt)
```

In [65]:

```
predict[predict==1]<-"functional"
predict[predict==2]<-"functional needs repair"
predict[predict==3]<-"non functional"
thread<-as.matrix(valLabel)
thread[thread==1]<-"functional"
thread[thread==2]<-"functional needs repair"
thread[thread==3]<-"non functional"

mtrx<-table(predict,thread)
mtrx
```

```
                         thread
predict                   functional functional needs repair non functional
  functional                    5708                     523           1218
  functional needs repair        141                     232             72
  non functional                 577                     111           3298
```

In [66]:

```
rf_time
t2<-mtrx[c(1,5,9)]
accuracy<-sum(t2)/size
sprintf('The accuracy in total for xgboost: %f ',accuracy)
TP<-sum(mtrx[1])
FP<-sum(mtrx[c(4,7)])
FN<-sum(mtrx[c(2,3)])
FScore4=2*TP/(2*TP+FP+FN)
sprintf('If we divide the data into two classes, then the FScore for xgboost: %f ',FScore4)
MacroScore<-MacroF1(1,5,9,mtrx)
sprintf('The Macro F1 for xgboost %f ',MacroScore)
```

```
Time difference of 2.530446 mins
```

'The accuracy in total for xgboost: 0.777609 '

'If we divide the data into two classes, then the FScore for xgboost: 0.822775 '

'The Macro F1 for xgboost 0.648669 '

Prediction

In last chapter we compared the classification accuracy of three candidates. According to this metric we determine to select Random Forest as the model for prediction.

In [67]:

```
test <- read.csv('TestSetValues.csv', header = TRUE)
test
```

A data.frame: 14850 × 40

id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_nam
<int>	<dbl>	<chr>	<chr>	<int>	<chr>	<dbl>	<dbl>	<chr
50785	0	2013-02-04	Dmdd	1996	DMDD	35.29080	-4.05969643	Dinamu Secondar Schoo
51630	0	2013-02-04	Government Of Tanzania	1569	DWE	36.65671	-3.30921425	Kimnya
17168	0	2013-02-01		1567		34.76786	-5.00434437	Puma Secondar
45559	0	2013-01-22	Finn Water	267	FINN WATER	38.05805	-9.41867222	Kwa Mze Pang
49871	500	2013-03-27	Bruder	1260	BRUDER	35.00612	-10.95041200	Kwa Mze Turuk

Do data processing on test set again.

As what we did on training set (and test set split from original train set).

In [68]:

```
test[, c(2,9,10,12,13,17,20,21,22,25,26,31,35,39,40)] <- NULL

test$longitude<-NULL
test$latitude<-NULL
test$basin<-NULL
test$lga<-NULL
test$management<-NULL
test$management_group<-NULL
test$water_quality<-NULL
test$source<-NULL
test$source_type<-NULL
test$public_meeting<-NULL
test$permit<-NULL

test$year_recorded <- str_sub(test$date_recorded, 1, 4)#library(tidyverse)
test$year_recorded <- as.numeric(test$year_recorded)
test$construction_year <- as.numeric(test$construction_year)
test$construction_year[test$construction_year<1960]= median(test$construction_year[test$constructio
test$age <- test$year_recorded - test$construction_year
test$age[test$age < 0] <- 0
test$construction_year<-NULL
test$date_recorded<-NULL
test$source_class<-NULL
test$year_recorded<-NULL

#Attention: We have to keep the levels of the test set the same as the training set or otherwise the
#For example: installer.top <- names(summary(as.factor(pump_tst3$installer)))[1:15]
#Here we select the top 15 installers in training(and validation) dataset
test$funder <- tolower(test$funder)
test$funder[test$funder %in% c(" ", "", "0", "_", "-")] <- "other"
funder.top <- names(summary(as.factor(pump_tst3$funder)))[1:15]
test$funder[!(test$funder %in% funder.top)] <- "other"
test$funder <- as.factor(test$funder)
test$installer <- tolower(test$installer)
test$installer[test$installer %in% c(" ", "", "0", "_", "-")] <- "other"
installer.top <- names(summary(as.factor(pump_tst3$installer)))[1:15]
test$installer[!(test$installer %in% installer.top)] <- "other"
test$installer <- as.factor(test$installer)
id<-test[,1]
test$id<-NULL
#test
```

In [69]:

```
test$quantity<-factor(test$quantity,levels=c('dry','unknown','insufficient','enough','seasonal'))
test$quality_group<-factor(test$quality_group,levels=c('colored','fluoride','good','milky','salty',
test$payment<-factor(test$payment,levels=c('never pay','unknown','pay annually','pay monthly','pay 
test$extraction_type_class<-factor(test$extraction_type_class,levels=c('gravity','handpump','motorp
test$funder<-factor(test$funder,levels=c('danida','dhv','district council','dwsp','government of ta
#test
```

In [70]:

```
test$status_group<-0
nrow(test)
#It doesn't matter what value of status_group here.
#I add this column just aim to make test set similar to the dataset we dealt with.To avoid error.
for (i in 1:nrow(test)){
    if(i%%3==1){
        test[i,12]<-"functional needs repair"
    }else{if(i%%3==2){
        test[i,12]<-"functional"

    }else{if(i%%3==0){
        test[i,12]<-"non functional"
    }}
       }
}

test$status_group<-factor(test$status_group,levels=c('functional','functional needs repair','non fu

dataset<-test %>%
  mutate_if(is.numeric, scale)
dataset
```

14850

With the model trained in Chapter 2.3, make prediction with random forest.

In [71]:

```
predict_rf <- predict(rf_model, dataset)
```

In [72]:

```
#length(predict_rf)
tmp<-as.data.frame(predict_rf)
status_group<-as.character(tmp$predict_rf)
result<-cbind(id,status_group)
result
```

A matrix: 14850 × 2 of type chr

id	status_group
50785	non functional
51630	functional
17168	functional
45559	non functional
49871	functional
52449	functional
24806	non functional
28965	non functional
36301	non functional
54122	functional
419	functional
45750	non functional
653	non functional
14017	functional
44607	functional
40228	functional
27714	functional
28785	functional
28330	functional
18532	non functional
69961	functional
55083	non functional
8691	non functional
30331	non functional
70970	functional
61136	functional
28799	non functional
46825	non functional
44718	functional needs repair
37350	non functional
...	...
52228	non functional

id	status_group
70038	functional
25901	non functional
21131	functional
26580	non functional
66059	functional
32944	functional
13686	functional
8471	non functional
19620	functional
74162	non functional
37994	functional
71151	functional
45017	functional
12592	functional needs repair
58693	functional
57539	functional
71252	non functional
7869	functional
57316	functional
59757	functional
64579	functional
57731	functional needs repair
65541	functional
68174	functional
39307	non functional
18990	functional
28749	functional
33492	functional
68707	non functional

And finally, store the result in .csv file.

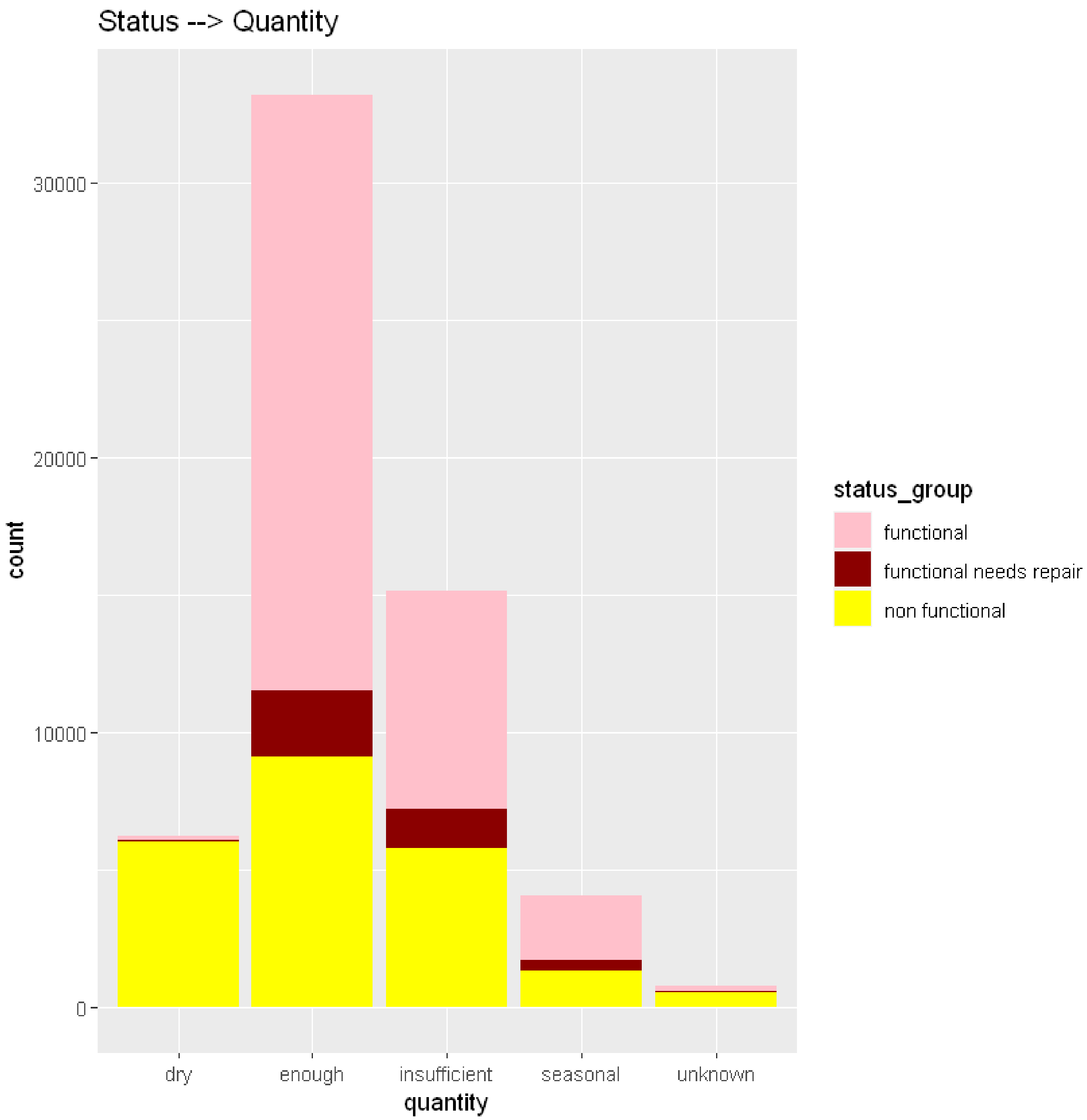
You can find it under the file path of the project.

In [73]:

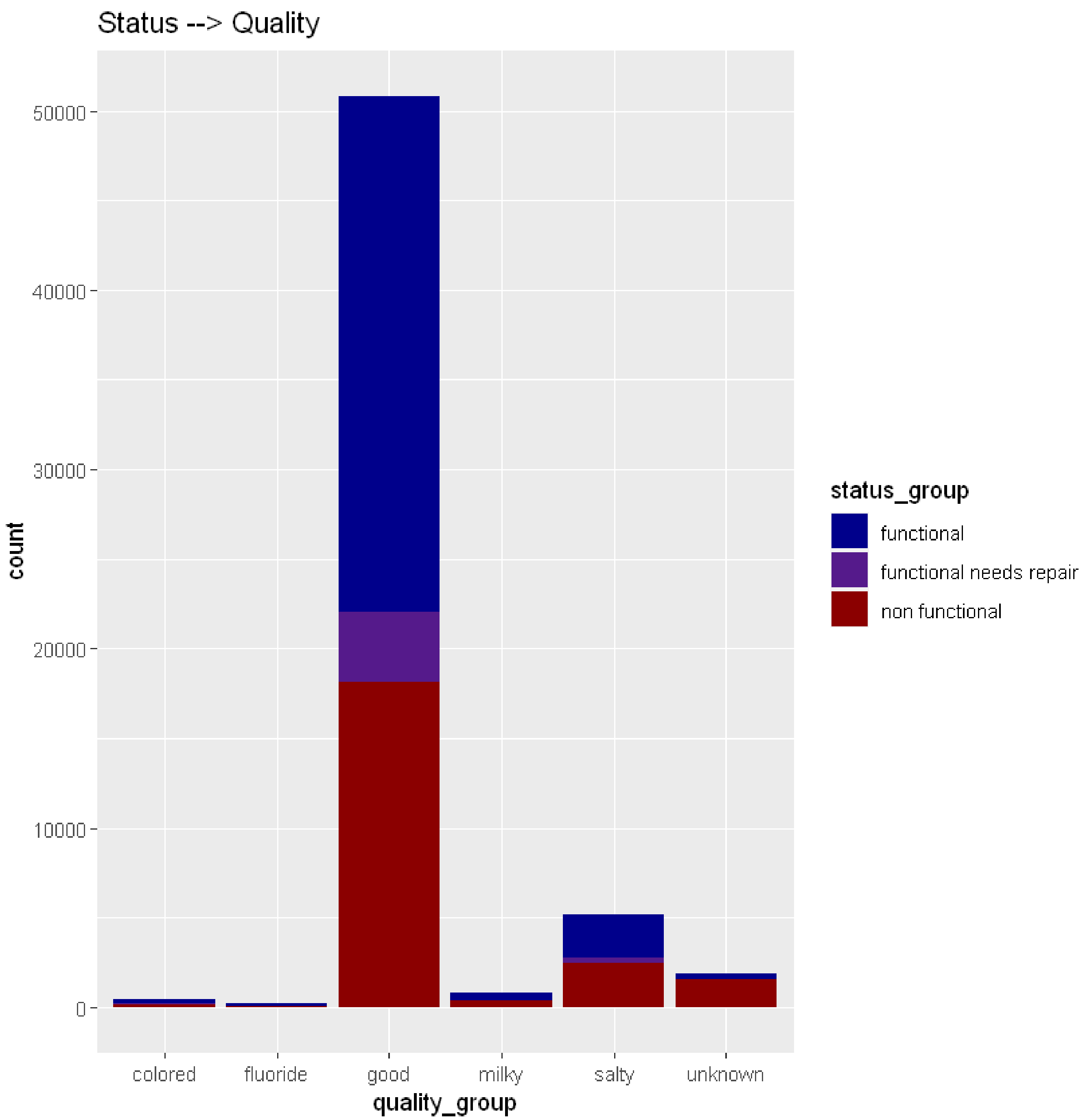
```
write.csv(result, file = "output.csv", row.names = FALSE)
```

Conclusions

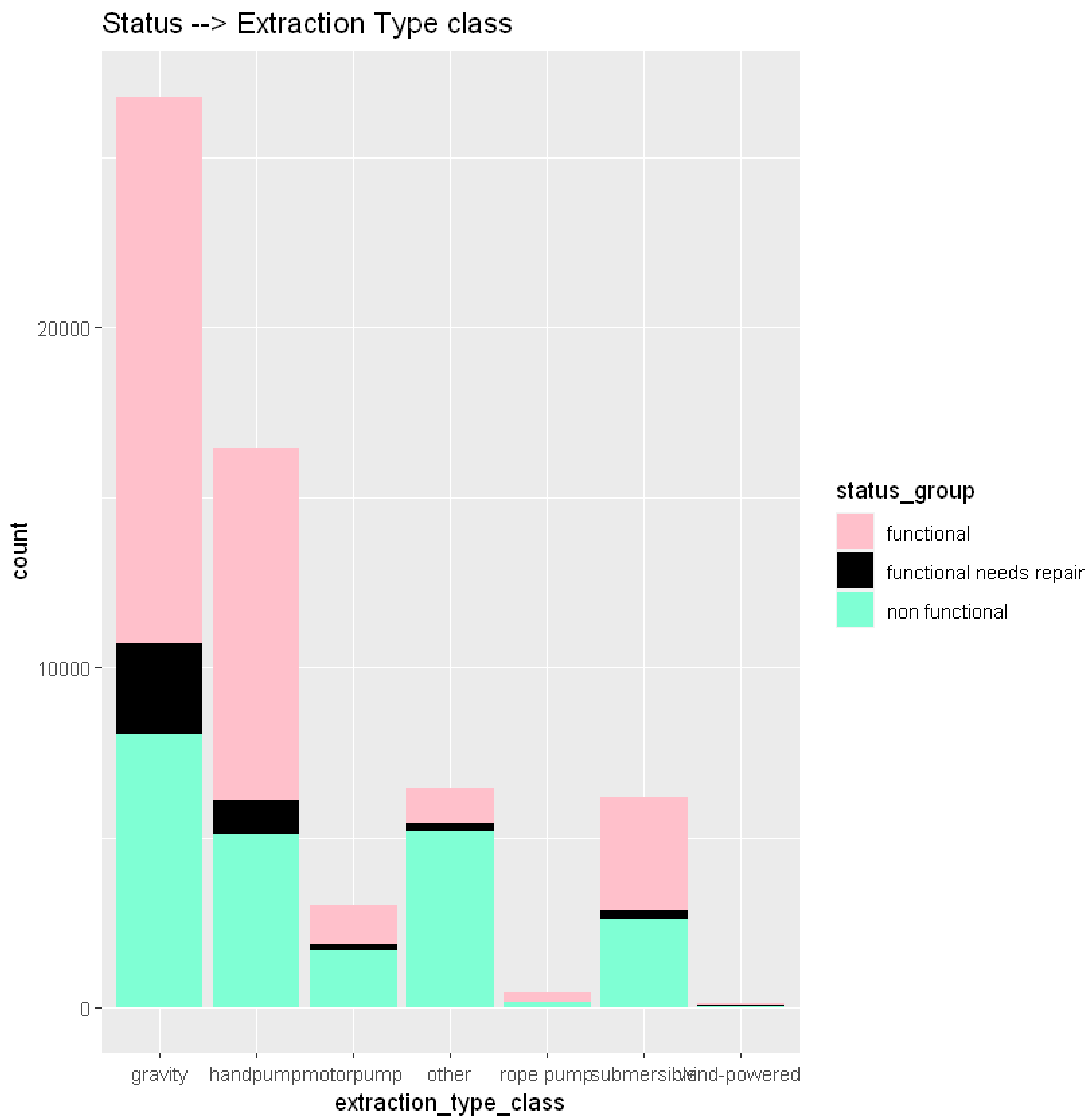
1- Our Dataset in brief:



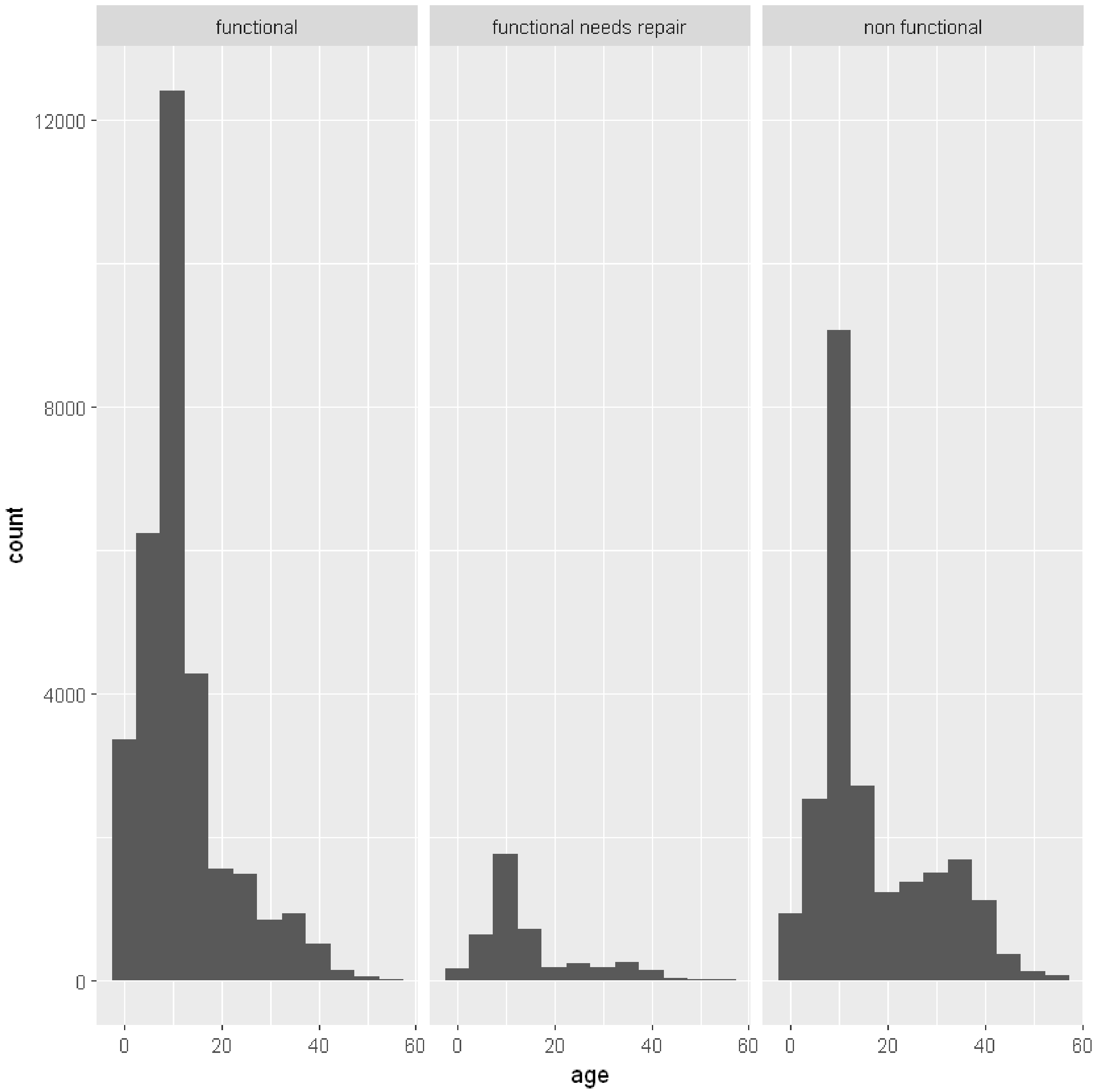
The diagram depicts that more than one-third(more than 20000) of the pumps in our dataset have enough water and at the same time, they are functional, while in contrast, the majority of dry pumps are not functional.



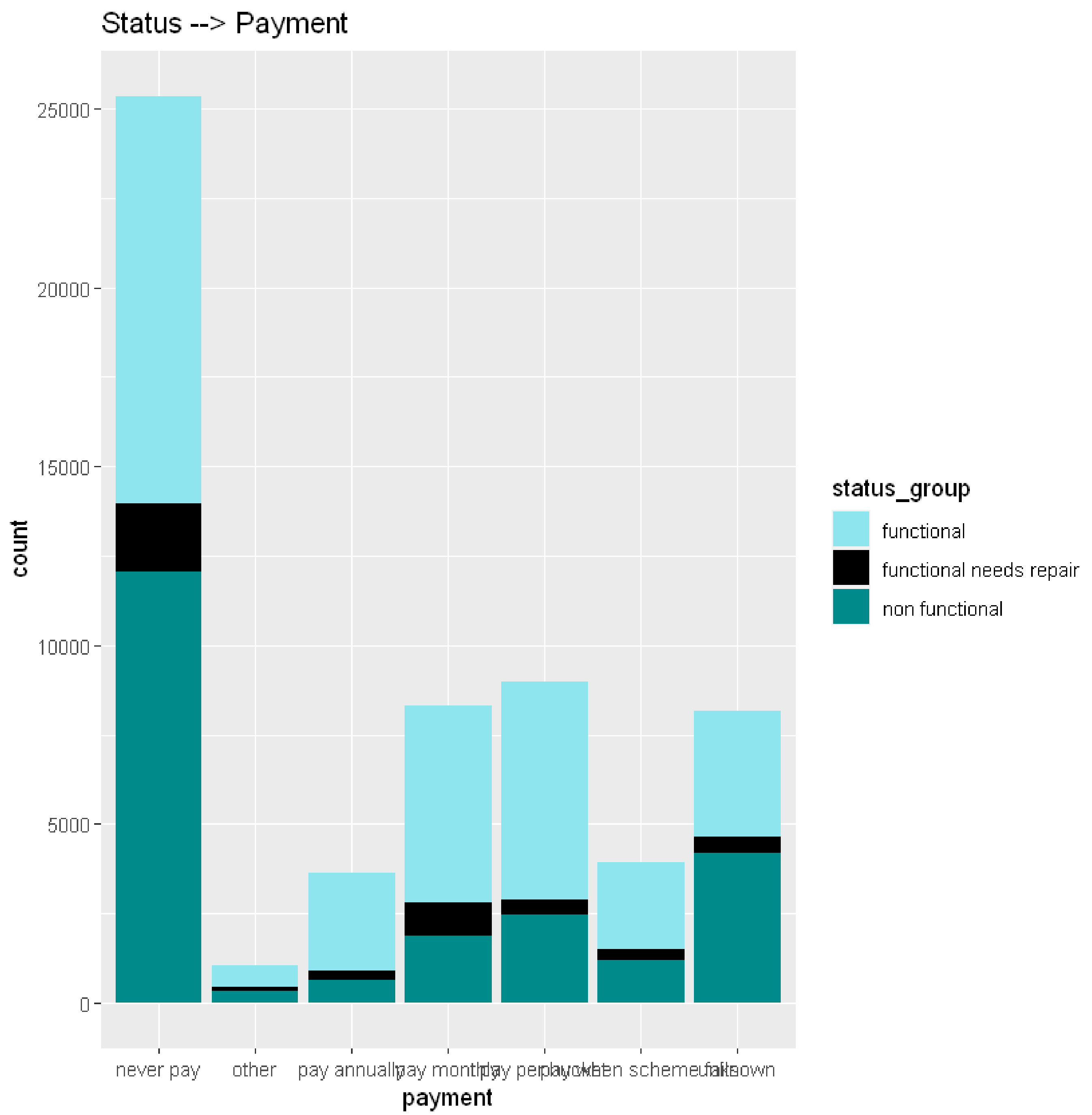
This plot led us to conclude that around half of the pumps are provided with good quality water, meanwhile, they are functional. Indeed the majority of pumps have good quality water. In contrast, most pumps with unknown water quality are not functional.
From another perspective,the soft water raises the probability of functionality of pumps, while salty water makes about an equal probability of functional and non-functional.



The most frequent type of extraction of water is "gravity",and more than half pumps in this group are functional. on the contrary, the majority of the "other" type are nonfuctional



The diagram explains that most of the pumps constructed in recent twenty years. And under twenty-year-old pumps fluctuate between functional and non-functional.



In fact, most of the pumps are never paid for the water they supply, and in this category around half of them are non-functional

2-Used Models and Their Scores

</tr>

#	Model	Classification Accuracy	F-Score	Macro-F1 Score	Time

#	Model	Classification Accuracy	F-Score	Macro-F1 Score	Time
1	Decision Trees	0.670286	0.782278	0.429795	1.029108 secs
2	KNN Model	0.670286	0.787944	0.570179	0.595443 secs
3	Random Forest	0.787710	0.833603	0.640276	2.525385 mins
Alternative	xgboost	0.769276	0.817931	0.624586	25.71637 secs

The best model to predict the functionality of the pumps was the Random Forest with accurately 0.787710 of the time and It's F-Score is higher than its accuracy :)

3- Discussions:

The prediction on class “Functional needs repair” performs worse accuracy than other classes.
One possible reason is that the data recorded from pumps needs repair may appears like other two kinds of pumps.Makes it difficult to classify it from other two categories through existing data and methods.

As it is shown in chapter 2.1, **each feature has a different influence on the model**.Which means their importance is not the same.
Which means we may remove some features that contribute a little to the model if there are too much features in data set.