# Project Draft

## Introduction

To safeguard agricultural productivity and maintain soil health in Telangana, our research team is conducting an analysis of irrigation water quality across various mandals within the state. Leveraging data obtained from the "Telangana Open Data Portal", we aim to identify potential correlations between water quality parameters and their implications for irrigation practices. This initiative seeks to provide actionable insights for farmers, agricultural scientists, and policymakers to optimize water resource management strategies. From previous pre-processing of the data, we found that the following features might have a bigger impact on our models.

Data Overview:

- **Mandal**: Administrative division in Telangana where the water sample was collected, serving as the primary geographical unit for our analysis.
- **Village**: Specific villages within each mandal from which samples were collected, offering granular data points for our study.
- **RSC** (Residual Sodium Carbonate): Indicates the excess of carbonate and bicarbonate ions over calcium and magnesium ions in water, crucial for evaluating the risk of soil sodicity.
- **SAR** (Sodium Adsorption Ratio): Measures the sodium hazard to crops and soil, a key factor in assessing water's suitability for irrigation.
- **Na** (Sodium content): The concentration of sodium in the water, which influences both SAR and the overall salinity hazard.
- **E.C** (Electrical Conductivity): Reflects the water's salinity level, directly impacting plant water uptake and soil structure.
- **TDS** (Total Dissolved Solids): The overall concentration of dissolved substances in water, indicating potential salinity issues.
- **HCO3** (Bicarbonate level): Concentration of bicarbonate in water, affecting soil pH and the precipitation of calcium and magnesium.

- **pH**: The acidity or alkalinity of the water, with significant implications for nutrient availability and microbial activity in the soil.
- **Classification.1:** Safety classification of the water (Safe P.S., Marginal MR, Unsafe U.S.). This variable is our response variable and has been changed to a binary response by changing MR to U.S.

The primary objective of this study is to establish a comprehensive understanding of irrigation water quality within Telangana's agricultural landscapes. By analyzing the relationships between the above water quality indicators and their collective impact on soil and plant health, we aim to categorize water sources into suitability classes for irrigation. This classification will help in formulating guidelines for water use in agriculture, thereby mitigating the risks associated with inappropriate water sources.

## Random Forest Model: (Brayan Gutierrez, Tie Wang, Ulises Ramirez)

To avoid the overfitting, high variance, and poor generalization to unseen data that a single decision tree is prone to, we decided to go straight to the random forest ensemble method. However, with deciding to use the random forest model we are sacrificing the interpretability that was present in a single decision tree, computational resources, and we gain the requirement of hyperparameter tuning that is not a major issue in a single decision tree. Like the previous model, our response variable is the `Classification.1` variable with the rest of the variables being our predictors. Below is the formula for our random forest model:

$$\text{Classification.1} \sim \text{Mandal} + \text{Village} + \text{RSC} + \text{SAR} + \text{Na} + \text{E.C} + \text{TDS} + \text{HCO3} + \text{pH}$$

First, we wanted to see how the base random forest with no tuned hyperparameters performs on our data. The data was read, the NA observations were omitted, and the response variable was changed to a binary response. Furthermore, since more than one category was present in the `Mandal` and `Village` predictors and for this model to function properly, these predictors were one-hot-encoded using the `dummyVars()` function from the caret library.

```r
library(tree)
```

```
Warning: package 'tree' was built under R version 4.3.3
```

```r
library(randomForest)
```

```
Warning: package 'randomForest' was built under R version 4.3.3
```

```
randomForest 4.7-1.1

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

```r
library(caret)
```

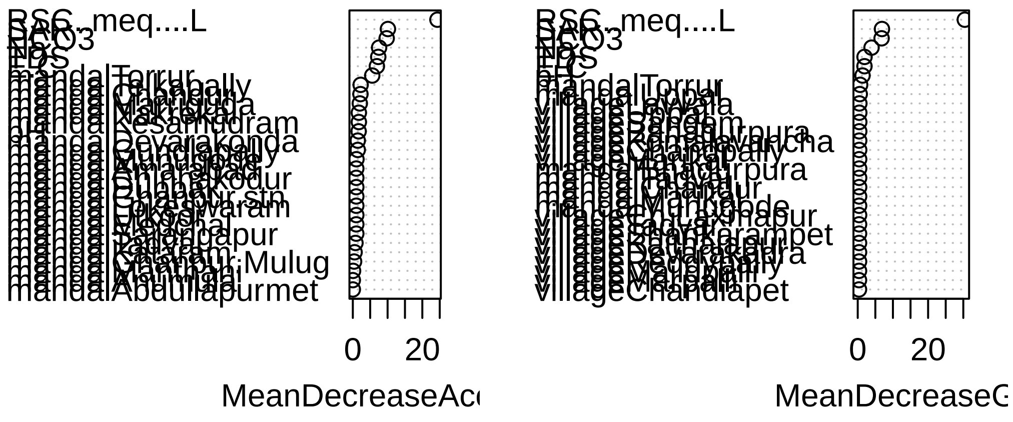
```
Loading required package: ggplot2


Attaching package: 'ggplot2'

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

    margin

Loading required package: lattice
```

```r
ground_water_quality_2022_post <- read.csv("ground_water_quality_2022_post.csv", header =

ground_water_quality_2022_post = na.omit(ground_water_quality_2022_post)

# Omitting CO3 and Season since they're the same
ground_water_quality_2022_post$CO3 = NULL
ground_water_quality_2022_post$season = NULL

# Selecting the most useful features
gw_df = ground_water_quality_2022_post[,c(23, 21, 16, 9, 10, 11, 8, 3, 4)]

# One-hot-encoding Mandal and Village
dummy = dummyVars(" ~ .", data = gw_df)
dum_df = data.frame(predict(dummy, newdata = gw_df))

# Changing Marginal Safe (MR) to Unsafe (U.S.)
Classification.1 = ground_water_quality_2022_post[,24]
final_gw_df = data.frame(dum_df, Classification.1)
final_gw_df$Classification.1 = gsub("MR", "U.S.", final_gw_df$Classification.1)
final_gw_df$Classification.1 = as.factor(final_gw_df$Classification.1)
```

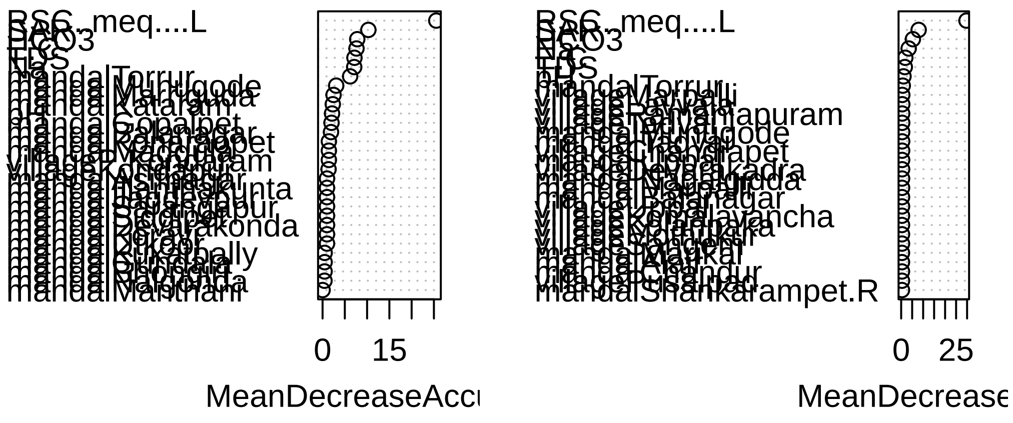
To obtain the testing error for a classification random forest model, we would need a confusion matrix. Since were are testing this model 10 times with different 80% training/20% testing sets, we decided to store all of the confusion matrices for these different splits in a list called `conf_mat`. The model was trained on the 10 different training sets and then tested on 10 different testing sets in a for loop as the `cv.glm()` function would only work on a regression random forest model. The seed used in our model was set to 1. Additionally, we also plotted the variable importance for each training/testing split. The code for this can be seen below:

```r
conf_mat = list()

set.seed(1)

for (i in 1:10) {

  # Splitting data
  sample = sample(1:nrow(final_gw_df), 0.8 * nrow(final_gw_df))
  training = final_gw_df[sample,]
  testing = final_gw_df[-sample,]

  # Random Forest Model
  rf_gw = randomForest(Classification.1 ~., data = training, proximity = T, importance = T

  # Importance Plots
  importance(rf_gw)
  varImpPlot(rf_gw, main = paste('Ground Water Random Forest Split', i, sep = ' '))

  test.pred = predict(rf_gw, newdata = testing)

  # Confusion Matrix
  conf_mat[[i]] = table(testing$Classification.1, test.pred)

}
```

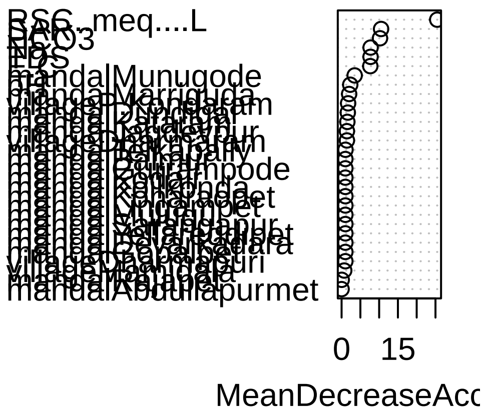
Ground Water Random Forest Split 1



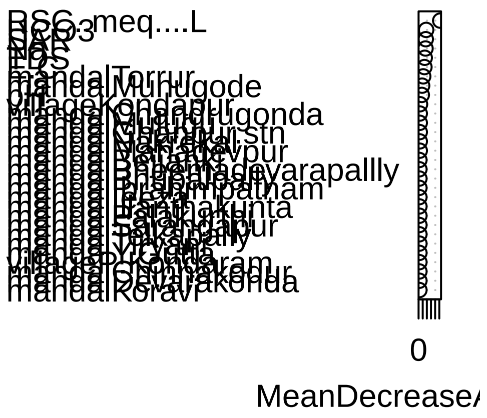
Ground Water Random Forest Split 2



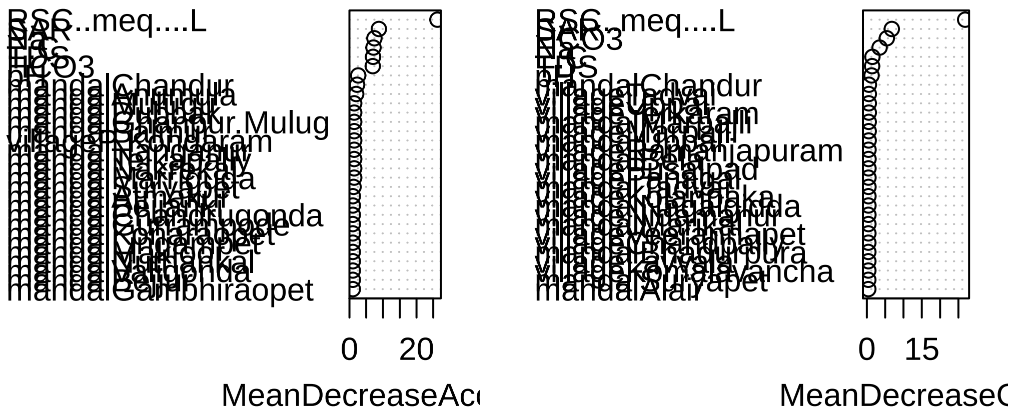
## Ground Water Random Forest Split 3



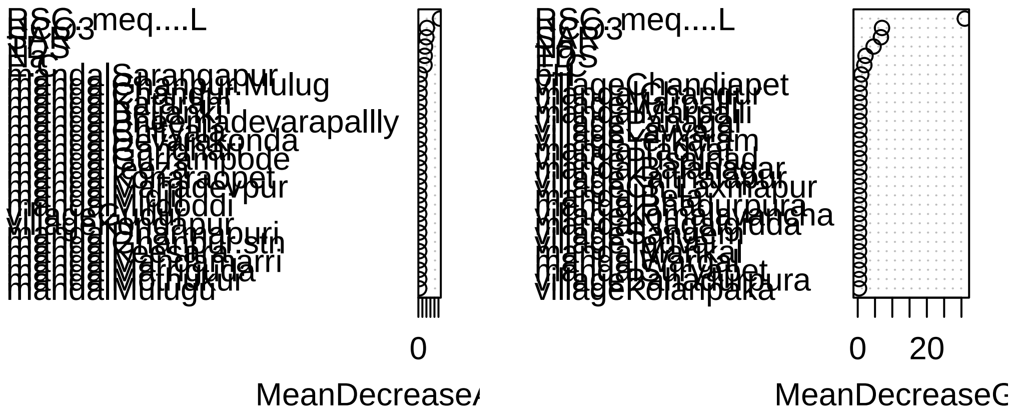
## Ground Water Random Forest Split 4



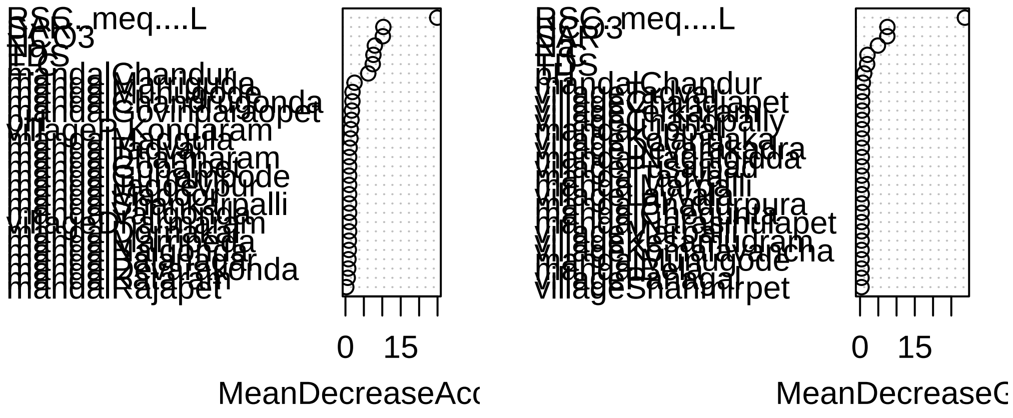
## Ground Water Random Forest Split 5



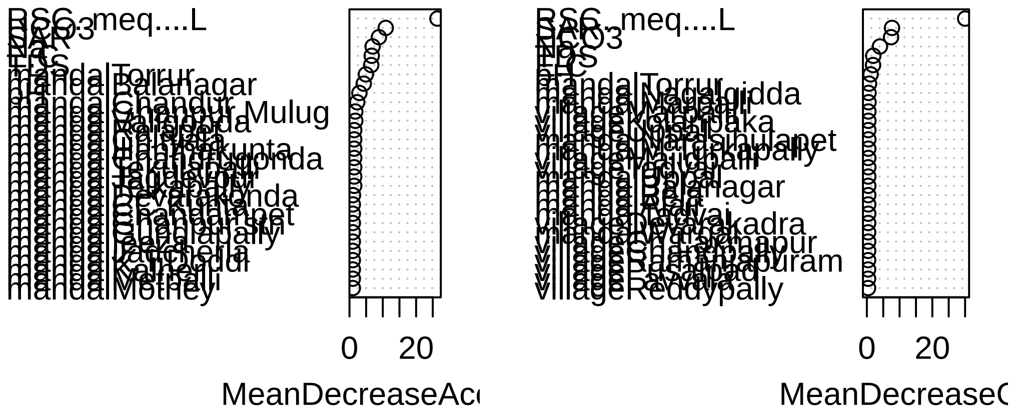
## Ground Water Random Forest Split 6

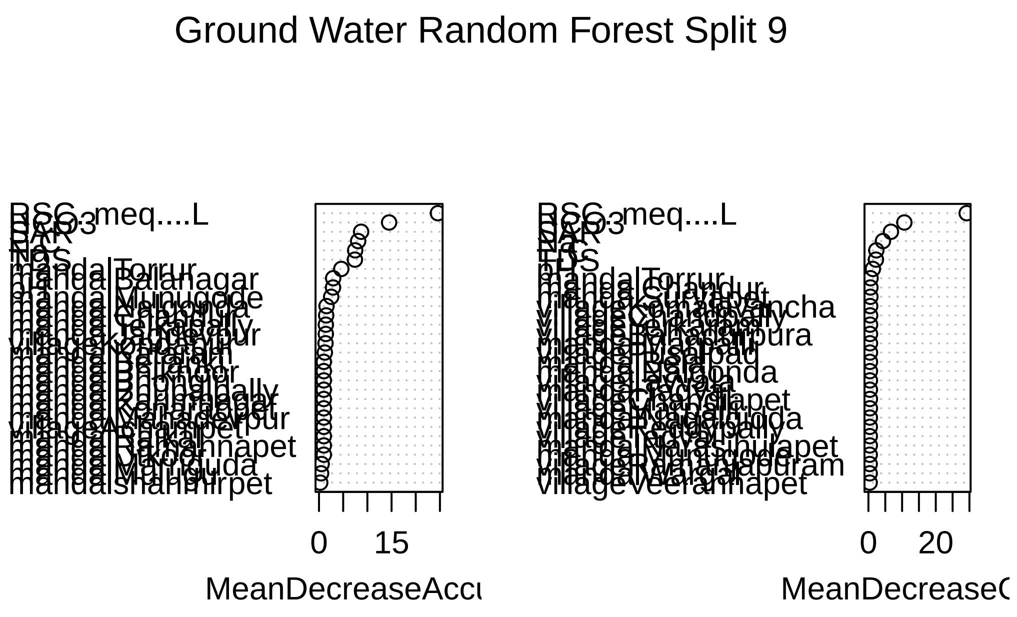


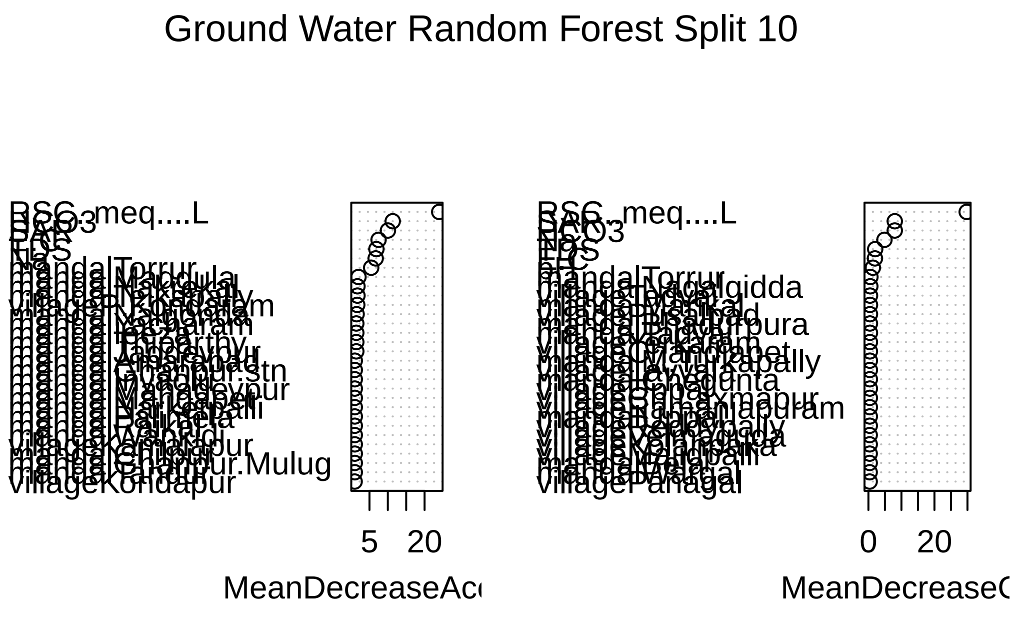
Ground Water Random Forest Split 7



Ground Water Random Forest Split 8







With the confusion matrices for each training/testing split now saved, we used these matrices to now calculate the testing error of each split.

```r
# Test Error Calculations
test_errors = rep(0, 10)

# Number 1
conf_mat[[1]]
```

```
    test.pred
     P.S. U.S.
P.S.  152    0
U.S.    4    8
```

```r
test_errors[1] = sum(0,4)/sum(152,0,4,8)

# Number 2
conf_mat[[2]]
```

```
    test.pred
     P.S. U.S.
P.S.  150    0
U.S.    1   13
```

```r
test_errors[2] = sum(0,1)/sum(150,0,1,13)

# Number 3
conf_mat[[3]]
```

```
    test.pred
     P.S. U.S.
P.S.  151    0
U.S.    1   12
```

```r
test_errors[3] = sum(0,1)/sum(151,0,1,12)

# Number 4
conf_mat[[4]]
```

```
    test.pred
     P.S. U.S.
P.S.  153    0
U.S.    1   10
```

```r
test_errors[4] = sum(0,1)/sum(153,0,1,10)

# Number 5
conf_mat[[5]]
```

```
    test.pred
     P.S. U.S.
P.S.  148    0
U.S.    1   15
```

```r
test_errors[5] = sum(0,1)/sum(148,0,1,15)

# Number 6
conf_mat[[6]]
```

```
    test.pred
     P.S. U.S.
P.S.  154    0
U.S.    0   10
```

```r
test_errors[6] = sum(0,0)/sum(154,0,0,10)

# Number 7
conf_mat[[7]]
```

```
    test.pred
     P.S. U.S.
P.S.  154    0
U.S.    2    8
```

```r
test_errors[7] = sum(0,2)/sum(154,0,2,8)

# Number 8
conf_mat[[8]]
```

```
    test.pred
     P.S. U.S.
P.S.  153    0
U.S.    0   11
```

```r
test_errors[8] = sum(0,0)/sum(153,0,0,11)

# Number 9
conf_mat[[9]]
```

```
       test.pred
        P.S. U.S.
  P.S.   155    0
  U.S.     2    7
```

```r
test_errors[9] = sum(0,2)/sum(155,0,2,7)

# Number 10
conf_mat[[10]]
```

```
       test.pred
        P.S. U.S.
  P.S.   156    0
  U.S.     2    6
```

```r
test_errors[10] = sum(0,2)/sum(156,0,2,6)

test_errors
```

```
 [1] 0.024390244 0.006097561 0.006097561 0.006097561 0.006097561 0.000000000
 [7] 0.012195122 0.000000000 0.012195122 0.012195122
```

```r
(avg_test_error = sum(test_errors) / 10)
```

```
[1] 0.008536585
```

We found that the base random forest model with 10 different training/testing splits had a test error between 0%-2.4%. With an average testing error of 0.8%. This result is very promising, however, *(write something about less U.S. observations).* Next, we then moved on to tuning the hyperparameters of the random forest model on our data…