# Random Forest model for groundwater arsenic in Punjab

Yusuf Jameel

12/10/2020

In this code, I am using random forest model to predict groundwater arsenic in the Indus Valley Basin where groundwater arsenic is pervasive and posses a serious health issue.

## Import the libraries

```
setwd("/Users/yusufjameel/Dropbox/Arsenic_MIT/Arsenic_prediction_indus_valley")
library(rsample)      # data splitting
library(randomForest) # basic implementation
library(ranger)       # a faster implementation of randomForest
library(caret)        # an aggregator package for performing many machine learning models
library(h2o)          # an extremely fast java-based platform
library(tidyverse)
library(ggExtra)
library(raster)
library(tibble)
library(sf)
library(rnaturalearthdata)
library(tidyverse)
library(rnaturalearthhires)
library(tmap)
library(scatterpie)
library(sp)
library(rgdal)
library(geosphere)
library(ggplot2)
library(cutpointr)
```

## Import the data

Read the csv file that contains 30,000 arsenic measurements Reassign the kit categories their mean concentration

```
As_Indus = read.csv('Indus_Arsenic_kit_vars.csv', as.is =T)

As_Indus$As__ppb = floor(As_Indus$As__ppb)

As_Indus$Recalibrated_As[As_Indus$As__ppb == 0] = 6
As_Indus$Recalibrated_As[As_Indus$As__ppb == 10] = 36
```

```
As_Indus$Recalibrated_As[As_Indus$As__ppb == 25] = 28
As_Indus$Recalibrated_As[As_Indus$As__ppb == 50] = 46
As_Indus$Recalibrated_As[As_Indus$As__ppb == 100] = 50
As_Indus$Recalibrated_As[As_Indus$As__ppb == 200] = 81
As_Indus$Recalibrated_As[As_Indus$As__ppb == 300] = 116
As_Indus$Recalibrated_As[As_Indus$As__ppb == 500] = 258
As_Indus$Recalibrated_As[As_Indus$As__ppb == 1000] = 719
As_Indus= As_Indus[complete.cases(As_Indus), ]
As_Indus$As_cat[As_Indus$Recalibrated_As < 10] = 0
As_Indus$As_cat[As_Indus$Recalibrated_As >= 10] = 1
As_Indus$As_cat = as.factor(As_Indus$As_cat)
```

## Plot the arsenic data

We will cluster the data as there are several thousands points In this chunk we will get the data ready to be plotted Since there are 30,000+ measurements over a small region, we will cluster the data and then plot the proportion of wells in each village that have low arsenic concentration (cyan) and high arsenic concentration (red). Each cluster represents all the points in a 10 km radius.

```
# convert data to a SpatialPointsDataFrame object
xy <- SpatialPointsDataFrame(
  matrix(c(As_Indus$Long,As_Indus$Lat), ncol=2),
  data.frame(ID=seq(1:length(As_Indus$Lat))),
  proj4string=CRS("+proj=longlat +ellps=WGS84 +datum=WGS84"))

# use the distm function to generate a geodesic distance matrix in meters
mdist <- distm(xy)

# cluster all points using a hierarchical clustering approach
hc <- hclust(as.dist(mdist), method="complete")

# define the distance threshold, in this case 10000 m (i.e. 10 km)
d=10000

# define clusters based on a tree "height" cutoff "d" and add them to the SpDataFrame
xy$clust <- cutree(hc, h=d)

As_Indus$Cluster = xy$clust
wells_per_cluster =As_Indus %>%
  count(Cluster)

wells_lat =As_Indus %>%
  group_by(Cluster) %>%
  summarise(mean_lat = mean(Lat, na.rm = TRUE))


wells_long =As_Indus %>%
  group_by(Cluster) %>%
  summarise(mean_long = mean(Long, na.rm = TRUE))

As_Indus_negative = As_Indus[which(As_Indus$As_cat == 0),]
```

```
wells_negative = As_Indus_negative %>%
  group_by(Cluster) %>%
  summarise(mean_zero = n())

As_clust = data.frame(cbind(wells_long$mean_long,
                            wells_lat$mean_lat, wells_per_cluster$n, wells_per_cluster$Cluster))
colnames(As_clust) = c("Long", "Lat", "Count", "Cluster")

As_Indus_clust  = merge (As_clust, wells_negative, by.x = "Cluster", by.y = "Cluster", all =T)

As_Indus_clust= As_Indus_clust[complete.cases(As_Indus_clust), ]

As_Indus_clust$less_10 = As_Indus_clust$mean_zero/As_Indus_clust$Count
As_Indus_clust$greater_10 = 1 - As_Indus_clust$less_10
```

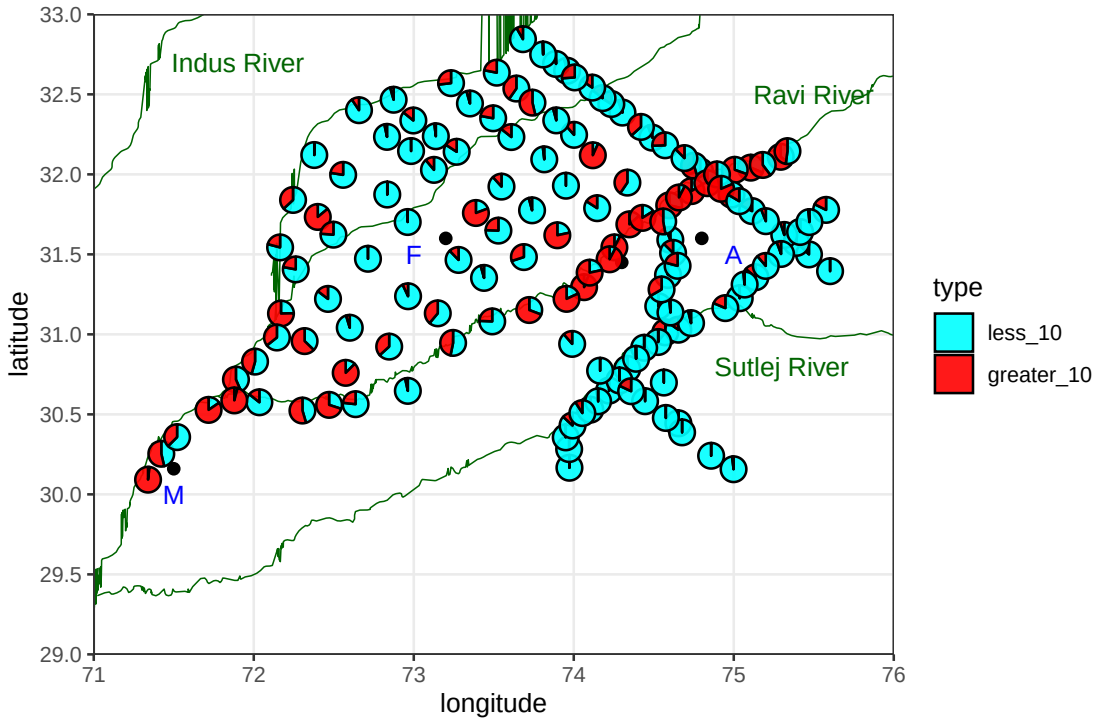
First we obtain shape file of the rivers in the basin. We also include major cities (black circle). After that we will plot the data.

```
india = rnaturalearth::ne_countries(country = "india", returnclass = "sf", scale = "medium")
river_punjab<- readOGR(dsn ="./world_rivers_dSe/world_rivers_dSe.shp")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "/Users/yusufjameel/Dropbox/Arsenic_MIT/Arsenic_prediction_Indus_valley/world_rivers_dSe/wor
## with 4080 features
## It has 8 fields
```

```
river_punjab_df <- fortify(river_punjab)

indus =    subset(river_punjab, river_punjab$Name1 == "Indus")
ravi =     subset(river_punjab, river_punjab$Name1 == "Ravi")
chenab =   subset(river_punjab, river_punjab$Name1 == "Chenab")
sutlej =   subset(river_punjab, river_punjab$Name1 == "Sutlej")
jhelum =   subset(river_punjab, river_punjab$Name1 == "Jhelum")

sites <- data.frame(longitude = c(74.30,71.50,73.2,74.8,73.1,76.8),
                    latitude = c(31.45,30.16,31.6,31.6,33.5,30.7)) ### major cities


ggplot() + theme_bw() +
  geom_point(data = sites, aes(x = longitude, y = latitude), size = 2,
             shape = 21, fill = "black") +
  coord_sf(xlim = c(71, 76), ylim = c(29, 33), expand = FALSE)+
  geom_line(data = indus,
            aes(x = long, y = lat), col = 'dark green', size = 0.3)+
  geom_line(data = ravi,
            aes(x = long, y = lat), col = 'dark green', size = 0.3)+
  geom_line(data = jhelum,
            aes(x = long, y = lat), col = 'dark green', size = 0.3)+
  geom_line(data = chenab,
            aes(x = long, y = lat), col = 'dark green', size = 0.3)+
  geom_line(data = sutlej,
            aes(x = long, y = lat), col = 'dark green', size = 0.3)+
```

```
geom_scatterpie(aes(x=Long, y=Lat, group=Cluster, r=0.08),
                data=As_Indus_clust, cols=c("less_10", "greater_10"),
                color="black", alpha=.9) +
scale_fill_manual(values= c("cyan", "red"))+
  annotate(geom="text", x=71.5, y=30, label="M",
             color="Blue")+
annotate(geom="text", x=75, y=31.5, label="A",
             color="Blue")+
  annotate(geom="text", x=73, y=31.5, label="F",
             color="Blue")+
annotate(geom="text", x=71.9, y=32.7, label="Indus River",
             color="dark green")+
annotate(geom="text", x=75.3, y=30.8, label="Sutlej River",
             color="dark green")+
  annotate(geom="text", x=75.5, y=32.5, label="Ravi River",
             color="dark green")
```




We see that most of the wells are uncontaminated and the contaminated wells lie on the eastern and southern part of the state. Ravi River has the largest proportion of the contaminated wells (high proportion of red - greater than 10 ppb of arsenic) for each cluster.

In the figure A is Amritsar, F is Faisalabad and M is Multan.

## Data cleaning

Check the class of the variables. We need to do this as many categorical variables are treated as numeric/integer.

```
str(As_Indus)
```

Convert factors that are defaulted as numeric to factor.

## Reassign arsenic category

Arsenic concentration less than 10 ppb is assigned a category of 0 Arsenic concentration greater than or equal to 10 ppb is assigned a category of 1

```
As_Indus$As_cat[As_Indus$Recalibrated_As < 10] = 0
As_Indus$As_cat[As_Indus$Recalibrated_As >= 10] = 1
As_Indus$As_cat = as.factor(As_Indus$As_cat)

As_Indus= As_Indus[complete.cases(As_Indus), ] ### only keep the complete cases
```

## Data rearrangement

since the predictor names are large, we reassign names to them: all the predictors are assigned the names var1 to var56 in alphabetical order.

```
all_data_select = As_Indus[,c(68, 7:8, 12:67)]

for (i in 1:56){
   colnames(all_data_select)[i+3] = paste0("var", i)
}
```

## Import all the raster across the entire punjab basin

We will import the rasters needed to predict the probability of arsenic in the Indus River Basin (we still haven't ran the model, however it is needed as all the data levels for factors are sometimes not included in the training dataset). So we will add all the factor levles present in the rasters to the training dataset.

Read the rasters

```
rastlist <- list.files(path = "./Punjab", pattern='.tif$', all.files=TRUE,
                       full.names=TRUE)
rast = lapply(rastlist, raster)

for (i in 1:56){
   names(rast[[i]]) = paste0("var",i)
}
```

## Stack the rasters

After stacking and assigning projection convert the variables which are assigned numeric by default into categorical variables

```
all_stack <- raster::stack(rast)

dataframe_raster <- raster::as.data.frame(all_stack, centroids = T, xy = T)
dataframe_raster <- dataframe_raster %>%
   drop_na()

colnames(dataframe_raster)[1] = "Long"
colnames(dataframe_raster)[2] = "Lat"

wgs.84 <- "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0" ##assign projection

dataframe_raster = dataframe_raster[,c(2,1,3:58)] ### rearrange the rasters
dataframe_raster$var16 = as.factor(dataframe_raster$var16)
dataframe_raster$var18 = as.factor(dataframe_raster$var18)
dataframe_raster$var28 = as.factor(dataframe_raster$var28)
dataframe_raster$var29 = as.factor(dataframe_raster$var29)
dataframe_raster$var43 = as.factor(dataframe_raster$var43)
dataframe_raster$var44 = as.factor(dataframe_raster$var44)
```

## Add all the factor levels to the training dataset

Add all the levels from raster to the test and training data: in some cases training data have only few levels - this will cause an error when we use the random forest model to predict the probability of arsenic exceeding 10 ppb using the predictors.

```
all_data_select$var41 = as.numeric(all_data_select$var41)
levels(all_data_select$var16) <- levels(dataframe_raster$var16)
levels(all_data_select$var18) <- levels(dataframe_raster$var18)
levels(all_data_select$var28) <- levels(dataframe_raster$var28)
levels(all_data_select$var29) <- levels(dataframe_raster$var29)
levels(all_data_select$var43) <- levels(dataframe_raster$var43)
levels(all_data_select$var44) <- levels(dataframe_raster$var44)
```

## Split the data in test and training

The data set was then randomly split into training (70%) and testing (30%) data sets while maintaining the same ratio of low and high values as in the full data set.

```
set.seed(123)
trainIndex <- createDataPartition(all_data_select$As_cat, p = .7,
                                  list = FALSE,
                                  times = 1)

iso_train <- all_data_select[trainIndex,]
iso_test <- all_data_select[-trainIndex,]
```

## Run random forest

The number of decision trees was fixed at 500, one-third of sample data was not included in each decision tree [out-of-bag (OOB) samples] by the bagging process. The randomly selected variables 'mtry' made available at each node was fixed at 8 (square root of the total number of variables).

As our data distribution shows a considerable class imbalance with more lower values (i.e., As <10 ppb) than higher values (As >10 ppb), the training dataset was randomly downsampled for each tree generation. We chose 2000 samples of arsenic greater than and lower than 10 ppb respectively at each node. In this way, the information from all measurements was incorporated in the model, while also avoiding the problem of overemphasizing low concentrations.

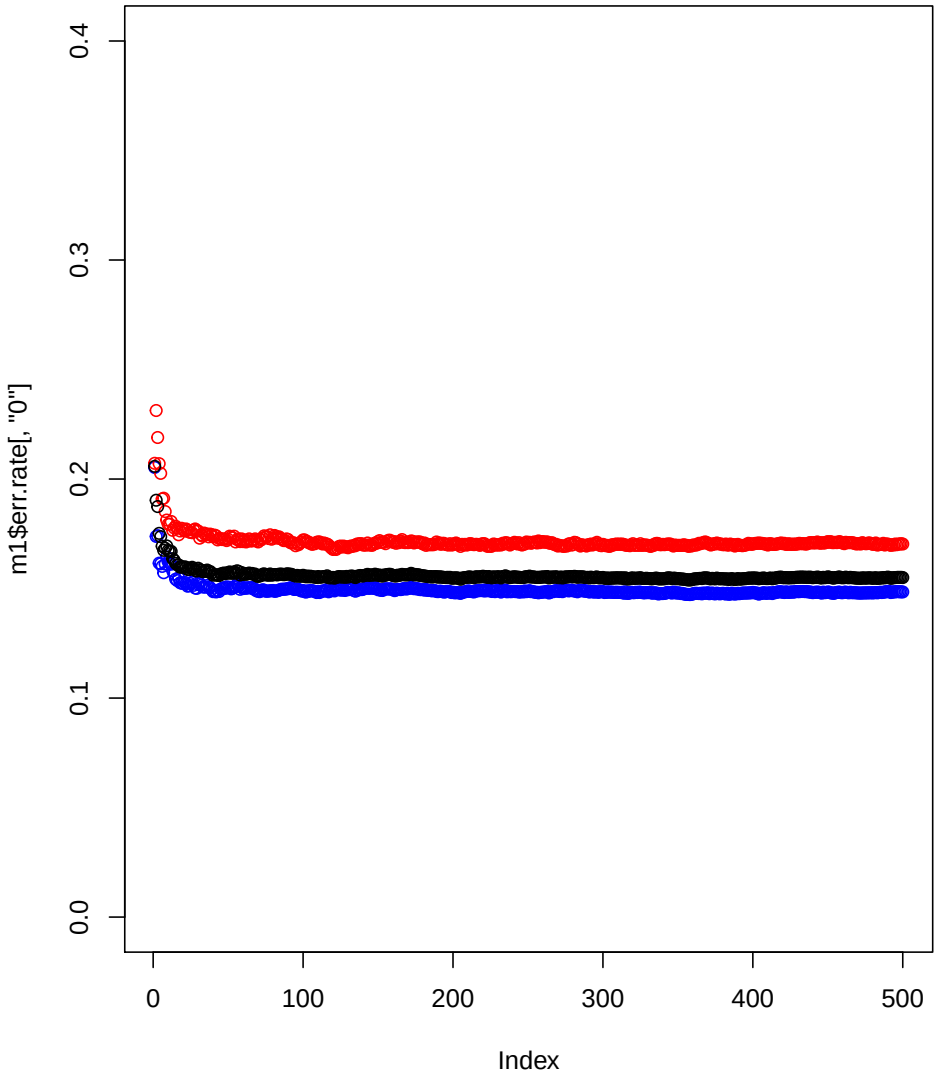
```
#mtry = 8 (square root of the total number of variables)
  m1 <- randomForest(
    formula = As_cat ~ .,
    data    = iso_train,
    localImp = T, na.action=na.exclude, ntree = 500, mtry =8, sampsize = c(2000,2000),
    replace=TRUE, importance = TRUE)
```

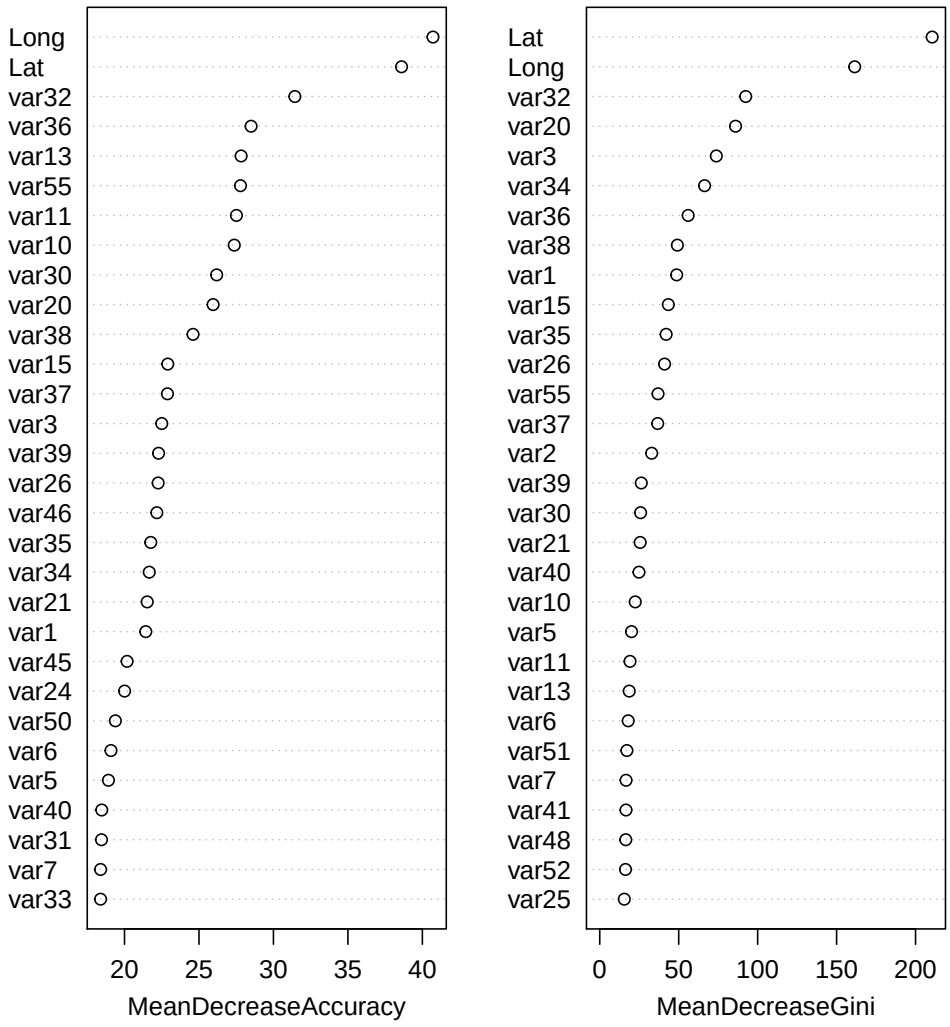
## Run diagnostic

Let's look at the OOB eroor and variable importnce plot

```
plot(m1$err.rate[,"0"], col = "blue", ylim = c(0,0.4))
  points(m1$err.rate[,"1"], col = "red")
  points(m1$err.rate[,"OOB"], col = "black")
  cat('\n\n')
```

```
  varImpPlot(m1) ### plot the variable importance
```


m1$err.rate[, "0"]
0.0
0.1
0.2
0.3
0.4
0
100
200
300
400
500
Index




## Predict for the test dataset

We will use the model output to classify probability of arsenic exceeding 10 ppb (category = 1) for the test dataset and compare with results with the actual arsenic category for each well.

```
pred_rf <- predict(m1, iso_test, type = "prob")
```

## Confusion matrix and other statistics for the test datastet

This can be done using a library, however I have done it manually here. The confusion matiex and other statistics will provide information on how good the model performs.

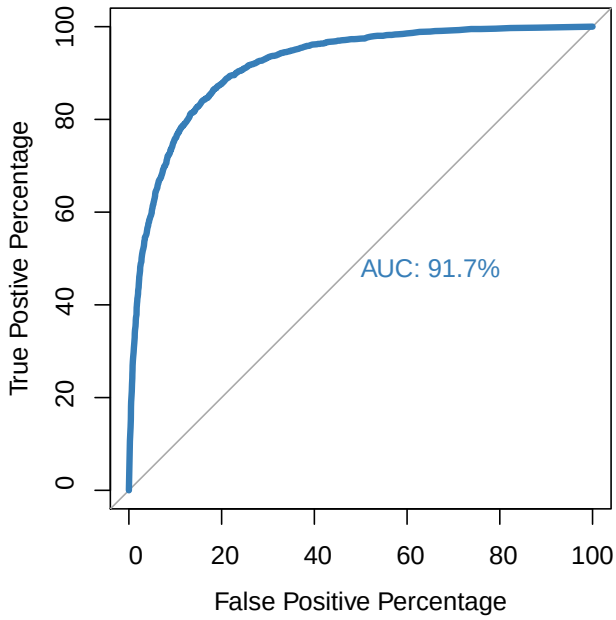
```
  model.df = data.frame(iso_test$As_cat, pred_rf[,2])
colnames(model.df) <- c("true", "predicted")
par(pty = "s")
pROC::roc(model.df$true, model.df$predicted, plot=TRUE,
          legacy.axes=TRUE, percent=TRUE,
          xlab="False Positive Percentage", ylab="True Postive Percentage",
          col="#377eb8", lwd=4, print.auc=TRUE)
```

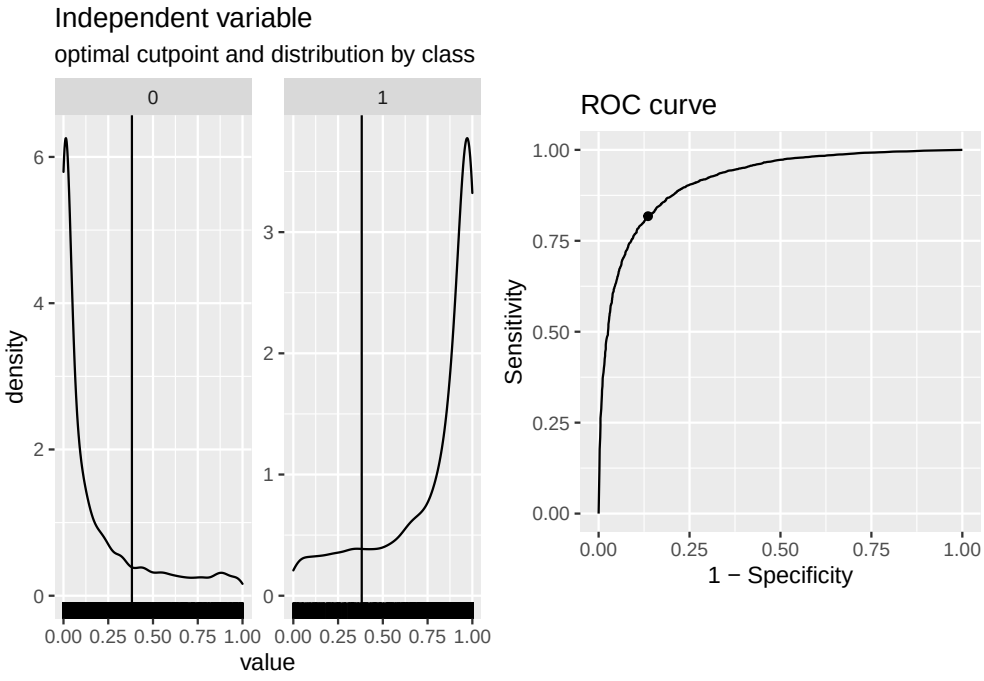
```
##
## Call:
## roc.default(response = model.df$true, predictor = model.df$predicted,     percent = TRUE, plot = TRU
##
## Data: model.df$predicted in 6394 controls (model.df$true 0) < 2745 cases (model.df$true 1).
## Area under the curve: 91.73%
```

```
##calcuate the optimale cutoff point
  cp <- cutpointr(model.df, predicted, true,
              method = maximize_metric, metric = sum_sens_spec, pos_class = 0)

plot(cp)

 cutoff = cp$optimal_cutpoint
 model.df$As_model_cat =  ifelse(model.df$predicted >= cutoff , 1, 0)

 model.df$As_model_cat = as.factor(model.df$As_model_cat)

 confusionMatrix(model.df$As_model_cat, model.df$true)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5222  372
##          1 1172 2373
##
##                Accuracy : 0.8311
##                  95% CI : (0.8232, 0.8387)
##     No Information Rate : 0.6996
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6289
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8167
##             Specificity : 0.8645
##          Pos Pred Value : 0.9335
##          Neg Pred Value : 0.6694
##              Prevalence : 0.6996
##          Detection Rate : 0.5714
##    Detection Prevalence : 0.6121
##       Balanced Accuracy : 0.8406
##
```

```
##         'Positive' Class : 0
##
```




The model performs pretty good. The accuracy is 85% with high sensitivity, however the specificity of the model is a bit low.

## Predict Arsenic for entire Punjab based on random forest model

```
rf_output <- predict(m1, dataframe_raster, type = "prob")

#########get the probability of arsenic >10 ppb for punjab

rf_model_Punjab <- tibble(longitude = dataframe_raster$Long,
                          latitude = dataframe_raster$Lat,
                          arsenic = rf_output[,2])

 rf_model_raster <- raster::rasterFromXYZ(rf_model_Punjab)
 proj4string(rf_model_raster) = wgs.84
```

## Plot the prediction map

Step 1: Get the rivers in the basin, major cities and the state boundaries

```
#######add some cites for references
   long = c(74.30,71.50,73.2,74.8,73.1,76.8)
   lat = c(31.45,30.16,31.6,31.6,33.5,30.7)
   city = c(1,1,1,1,1,1)
   cities = data.frame(long,lat,city)
   cities_points <- cities %>%
     st_as_sf(coords = c('long', 'lat'), crs = wgs.84)

   ############################get state and country boundry
   states_ind = rnaturalearth::ne_states(country = "india")
   Punjab_ind = subset(states_ind, states_ind$name %in% c("Punjab"))
   states_pak = rnaturalearth::ne_states(country = "pakistan")
   Punjab_pak = subset(states_pak, states_pak$name %in% c("Punjab"))
   ab <- aggregate(rbind(Punjab_pak,Punjab_ind))
   ab ### only the state of punjab in India and Pakistan

   ############################get rivers in the basin
   river_punjab<- st_read(dsn ="./world_rivers_dSe/world_rivers_dSe.shp")
   indus =   subset(river_punjab , river_punjab$Name1 == "Indus")
   ravi =   subset(river_punjab , river_punjab$Name1 == "Ravi")
   chenab =   subset(river_punjab, river_punjab$Name1 == "Chenab")
   sutlej =   subset(river_punjab, river_punjab$Name1 == "Sutlej")
   jhelum =   subset(river_punjab, river_punjab$Name1 == "Jhelum")
```

Step -2 Clip the model output to Punjab

```
punjab_crop <- crop(rf_model_raster, ab)
punjab_crop <- mask(punjab_crop, ab)
st_crs(river_punjab)==st_crs(punjab_crop)
```

Plot the map -Step 3

```
tm_shape(punjab_crop) +
    tm_raster(n=8, palette="-RdBu", midpoint = 0.5) +
     tm_shape(cities_points) + tm_dots(size = 0.25, shape=20) +
     tm_shape( ab) + tm_polygons(col = "gray99",border.col = "black", lwd = 0.75,alpha = 0.1,legend.sho
     tm_layout(main.title  = "Random forest prediction for As > 10 ppb", legend.bg.color = "white", leg
              legend.frame=F, legend.outside = F, bg.color = "gray80") +
     tm_shape(indus) + tm_lines(col = "darkgreen",  alpha = 0.7, lwd = 1, lty = "solid") +
     tm_shape(ravi) + tm_lines(col = "darkgreen", alpha = 0.7, lwd = 1, lty = "solid") +
     tm_shape(chenab) + tm_lines(col = "darkgreen",  alpha = 0.7, lwd = 1, lty = "solid") +
     tm_shape(sutlej) + tm_lines(col = "darkgreen",  alpha = 0.7, lwd = 1, lty = "solid") +
     tm_shape(jhelum) + tm_lines(col = "darkgreen",  alpha = 0.7, lwd = 1, lty = "solid") +
     tm_grid(lwd = 0.25, labels.size = 1, n.x = 4, n.y =4)
```