Untitled

Luke Wertis

2022-10-18

Read in the data

This dataset contains information extracted from the LiDAR dataset and the aerial imagery data. All variables are being looked at initially to try and predict the type of tree being observed.

```
Data <- fread("C:\\Users\\wertisml\\Documents\\Hemlocks\\Scripts_and_Tools\\tree_crowns\\Final\\csv\\Tr

names(Data)
```

```
##  [1] "treeID"      "Z"           "npoints"     "cnvhll_"     "zmax"
##  [6] "zmean"       "zsd"         "zskew"       "zkurt"       "zentrpy"
## [11] "pzbvzmn"     "pzabov2"     "zq5"         "zq10"        "zq15"
## [16] "zq20"        "zq25"        "zq30"        "zq35"        "zq40"
## [21] "zq45"        "zq50"        "zq55"        "zq60"        "zq65"
## [26] "zq70"        "zq75"        "zq80"        "zq85"        "zq90"
## [31] "zq95"        "zpcum1"      "zpcum2"      "zpcum3"      "zpcum4"
## [36] "zpcum5"      "zpcum6"      "zpcum7"      "zpcum8"      "zpcum9"
## [41] "itot"        "imax"        "imean"       "isd"         "iskew"
## [46] "ikurt"       "ipgrond"     "ipcmz10"     "ipcmz30"     "ipcmz50"
## [51] "ipcmz70"     "ipcmz90"     "p1th"        "p2th"        "p3th"
## [56] "p4th"        "p5th"        "pground"     "n"           "area"
## [61] "egn_lrg"     "egn_mdm"     "egn_sml"     "curvatr"     "linerty"
## [66] "planrty"     "sphrcty"     "anstrpy"     "hrzntlt"     "treID_3"
## [71] "imean_1"     "isd_1"       "iskew_1"     "ikurt_1"     "ipcmz05"
## [76] "ipc10_1"     "ipcmz15"     "ipcmz20"     "ipcmz25"     "ipc30_1"
## [81] "ipcmz35"     "ipcmz40"     "ipcmz45"     "ipc50_1"     "ipcmz55"
## [86] "ipcmz60"     "ipcmz65"     "ipc70_1"     "ipcmz75"     "ipcmz80"
## [91] "ipcmz85"     "ipc90_1"     "ipcmz95"     "ipcm100"     "R"
## [96] "G"           "B"           "Canopy_Base" "Tree_Type"
```

```
n.cores <- parallel::detectCores() - 1
```

Prepare the data

zentropy has many NA values within its column and for that reason it is being removed from the analysis. Additionally, for a random forest classification to work, the outcome variable, Tree_Type, needs to be a factor.

The data is then partined using a 80/20 train/test split. The train will be used for tuning the random forest model and the test will be left out of the training and will be used at the end to test how well the model performs when predicting on new data.

```r
# Fix the names of the data if needed
Data <- Data %>%
  dplyr::select(-zentrpy)

# Turn the outcome varialbe into a factor
Data$Tree_Type <- as.factor(Data$Tree_Type)

# Remove the NA rows
Data <- Data[complete.cases(Data),]

# Split the data for training and testing
set.seed(24)
trainIndex <- createDataPartition(Data$Tree_Type,
                                  p = .8, #(80/20 split of the data)
                                  list = FALSE,
                                  times = 1)

Train <- as.data.frame(Data[ trainIndex,])
Test  <- as.data.frame(Data[-trainIndex,])
```

Model Tuning

a grid is created to create multiple combinations of models (250) for the random forest to test. The best model will have the lowest prediction error.

Additionally, this model is run in parallel to allow for faster computation. This is not necessary for this model at the moment, but if a lot of new information was added it would be useful to already have set up.

```r
myGrid <- expand.grid(num.trees = seq(1, 2001, 500),
                      mtry = seq(1, 10, 1),
                      min.node.size = seq(1, 21, 5))

#create and register cluster
my.cluster <- parallel::makeCluster(n.cores)
doParallel::registerDoParallel(cl = my.cluster)

#fitting each rf model with different hyperparameters
set.seed(24)
prediction.error <- foreach(num.trees = myGrid$num.trees,
                            mtry = myGrid$mtry,
                            min.node.size = myGrid$min.node.size,
                            .combine = 'c',
                            .packages = "ranger") %dopar% {

  #fit model
  m.i <- ranger::ranger(data = Train,
                        dependent.variable.name = "Tree_Type",
                        classification = TRUE,
                        num.trees = num.trees,
                        mtry = mtry,
                        min.node.size = min.node.size)

  #returning prediction error as percentage
  return(m.i$prediction.error * 100)

}
```

Optimal Hyperparameters

From all the different models tested on the outcome with the lowest prediction error will be selected and those parameters will be used in the final model.

```r
myGrid$prediction.error <- prediction.error

best.hyperparameters <- myGrid %>%
  dplyr::arrange(prediction.error) %>%
  dplyr::slice(1)
```

Fit the model

The best hyperparameters are then added into a new grid for mtry and min.node.size. 10 fold-cv is set up for the final testing of the model using the caret package and lastly this is run in the train function to determine the best fitting.

```r
grid_tune <- expand.grid(.mtry = best.hyperparameters$mtry,
                         .splitrule = "gini",
                         .min.node.size = best.hyperparameters$min.node.size)

train_control <- trainControl(method = "cv",
                              number = 10,
                              #sampling = "smote", #if there is an imbalance in the classes
                              verboseIter = TRUE,
                              classProbs = TRUE,
                              allowParallel = TRUE)


set.seed(24)
rf_tune <- caret::train(x = Train[,-98],
                         y = Train[, 98],
                         trControl = train_control,
                         tuneGrid = grid_tune,
                         num.trees = best.hyperparameters$num.trees,
                         method = "ranger",
                         metric = "Accuracy",
                         verbose = TRUE)
```

```
## Aggregating results
## Fitting final model on full training set
```

model prediction

using the best random forest hyperparameters a new model is created, is is then used with the Test data from earlier to predict the tree classification from the unkown data set.

Using the confussion matrix we can see how well the model performed on predicting the tree type based on the data provided to it.

```
probabilities = predict(rf_tune, Test)

confusionMatrix(probabilities, Test$Tree_Type, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Deciduous Hemlock Pine
##   Deciduous        19       0    2
##   Hemlock           0       8    0
##   Pine              1       0    5
##
## Overall Statistics
##
##                Accuracy : 0.9143
##                  95% CI : (0.7694, 0.982)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 9.733e-06
##
##                   Kappa : 0.8498
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Deciduous Class: Hemlock Class: Pine
## Sensitivity                    0.9500         1.0000      0.7143
## Specificity                    0.8667         1.0000      0.9643
## Pos Pred Value                 0.9048         1.0000      0.8333
## Neg Pred Value                 0.9286         1.0000      0.9310
## Precision                      0.9048         1.0000      0.8333
## Recall                         0.9500         1.0000      0.7143
## F1                             0.9268         1.0000      0.7692
## Prevalence                     0.5714         0.2286      0.2000
## Detection Rate                 0.5429         0.2286      0.1429
## Detection Prevalence           0.6000         0.2286      0.1714
## Balanced Accuracy              0.9083         1.0000      0.8393
```