STAT511-RF

Arif

November 22, 2017

Random Forest Classification

Next, we used Forest classification - a non-linear, decision tree-based ensemble learning algorithm, was used to examine the tundra wildfire-climate relationship. Random Forest classification method as it yields relatively better classifcation results among all tree-based methods. As opposed to growing single decision tree (as in CART), random forest grows multiple trees, with having each split to consider only a subset of all predictors.Then it takes average of all trees to make final tree. In this way, random forest can reduce amount of potetial correlation between trees and thereby helps reduce the variance of the final tree.

Before fitting Random Forest model, we randomly partitioned the wildfire-climate dataset into training (75%) and test (25%) data samples. Therefore, out of total 940 'yes-fire' and 'no-fire' observations (spatial grids), 705 were randomly designated as training and 235 as test observations. The fitted model on training sample grids (27 climate variables were used for splitting at each tree node: best mtry parameter with least out of bag error) was then used to predict wildfire occurrence from the test sample grids.

After splitting datset, we used tuneRF function to find the optimal numbers of variables to try (mtry) splitting on at each node. We found mtry = 13 produces least out of the box (OBB) error (14.18%), that means, 13 out of 84 predictors should be considered for each split.

```
firedata = read.csv("Annual_forClassification.csv")

firedata$Y = as.factor(firedata$Y)

# Divide into training and test
n <- dim(firedata)[1]
p <- dim(firedata)[2]

set.seed(2017)
test <- sample(n, round(n/4))
train <- (1:n)[-test]
fire.train <- firedata[train,]
fire.test <- firedata[test,]

#-------------------------------------------------------------------------------
## Tune the RF: Finding optimul numbers of variables for splitting on each node
#-------------------------------------------------------------------------------

bestmtry <- tuneRF(fire.train[-1], fire.train$Y, ntreeTry=100,
                   stepFactor=1.5,improve=0.01, dobest=FALSE, plot = F)
```

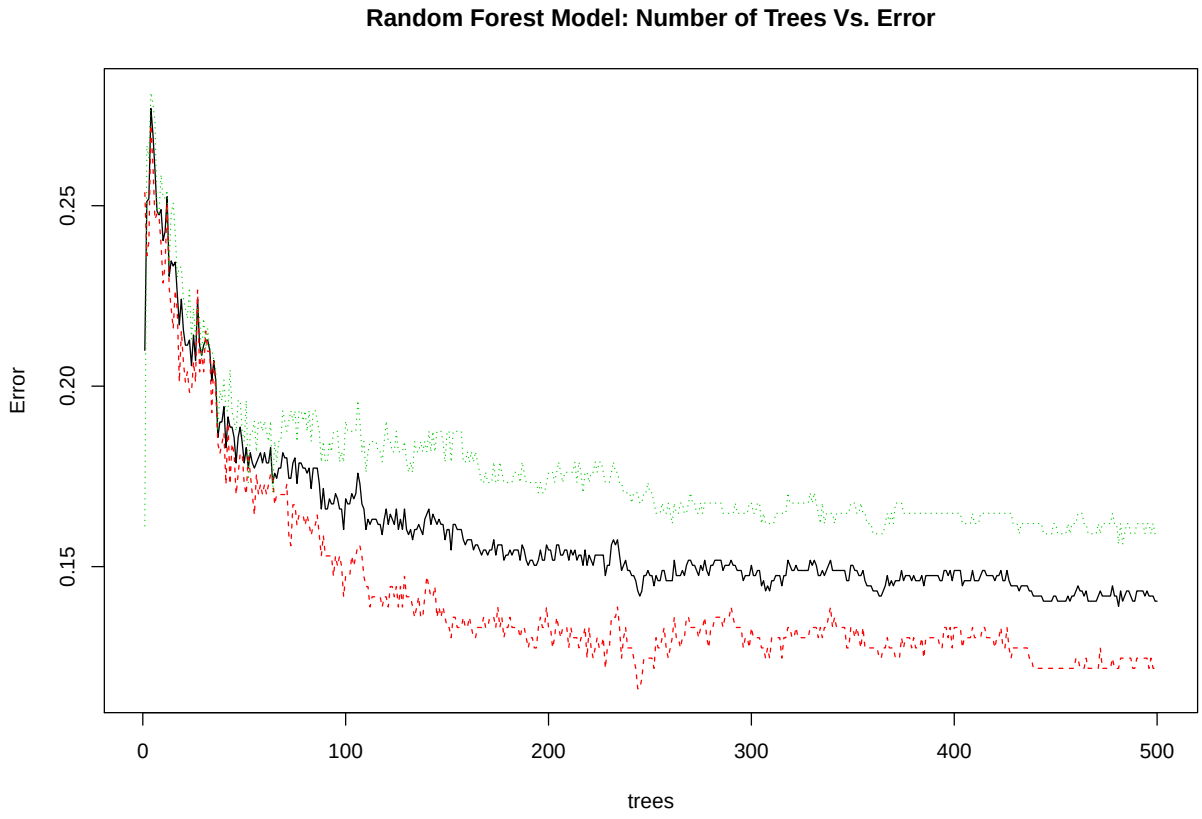
First we used training dataset for model fitting and prediction. In doing so, in the Random Forest model algorithm, we specified Yes(1)/No(0) fire as response versus all climate varibales (total 84) as predictors. We also specify mtry = 13 as found previously from searching for optimal value. The number of trees to grow is 500. We have tried ntree= 100, 1000, 1500, and 2000 trees. These different number of trees didn't cause any siginificant differences in model performance.

```
rf.fire = randomForest(Y~., data = fire.train, mtry=13, ntree=500, importance = TRUE)
```

From plotted model we can see that, albeit little fluctuations after about 250 trees there isn't much changes in terms of error. Black solid line is for overall OOB error, green line represents classification error rate for "yes" fire, whereas red line represents classification error rate for "no" fire cases.

After printing the model we can see that number of variables tried at each split to be 13 and an OOB estimate of error rate 14.04%. The training model fits the training data fairly well. Out of total 366 "no fire" observations, 56 of them were missclassified as "yes fire" observations. On the other hand, out of total 339 "yes fire" observations, 43 of them were misclassified as "no fire" observations.

```
plot(rf.fire, main = "Random Forest Model: Number of Trees Vs. Error")
```

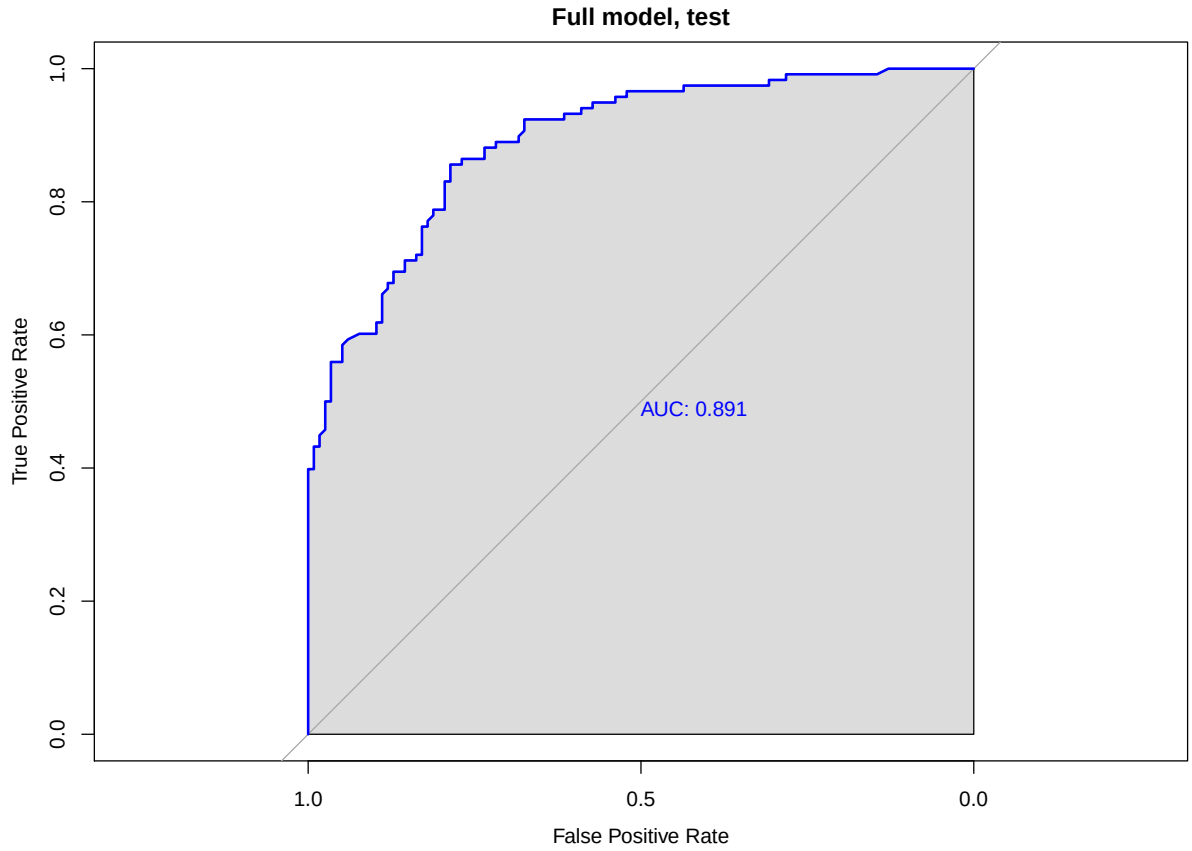



```
rf.fire
```

```
##
## Call:
##  randomForest(formula = Y ~ ., data = fire.train, mtry = 13, ntree = 500,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##         OOB estimate of  error rate: 14.04%
## Confusion matrix:
##     0   1 class.error
## 0 310  43   0.1218130
## 1  56 296   0.1590909
```

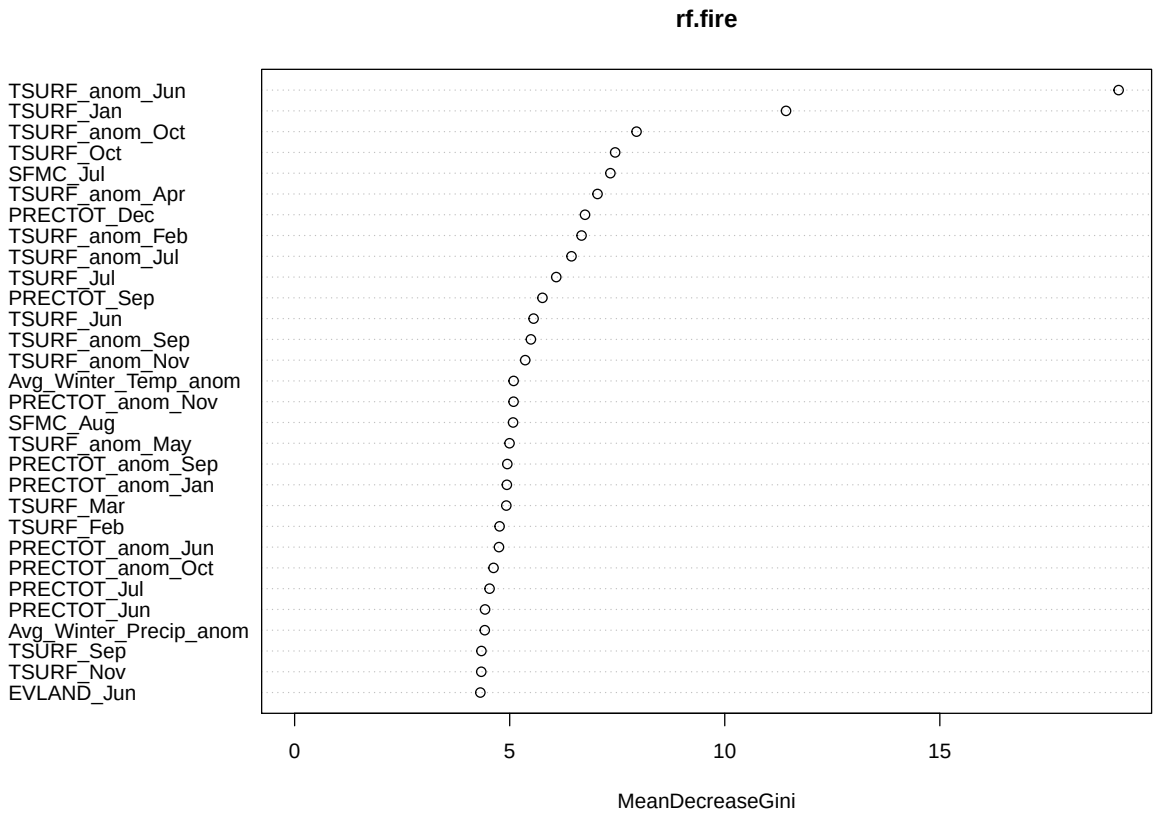
```r
yhat.rf.train = predict(rf.fire, type = "prob", newdata = fire.train)[,2]
# train.pred = prediction(yhat.rf.train, fire.train$Y)
# train.pred.perf = performance(train.pred, "tpr", "fpr")
# plot(train.pred.perf, main = "ROC Curve for Random Forest", col = 2, lwd = 2)
# abline(a=0,b=1,lwd=2,lty=2,col="gray")
```

Next, we used the fitted model on training dataset to predict "yes fire" and "no fire" events in test dataset. Although



The variable importance plot:

The variable importance plot shows which variables had the greatest impact in the classification model. Here I printed Mean Decreasing Gini - which is a measure of how each variable contributes to the homogeneity of the nodes and leaves in the resulting random forest. Each time a particular variable is used to split a node, the Gini coefficient for the child nodes are calculated and compared to that of the original node. The changes in Gini are summed for each variable and normalized at the end of the calculation.



We see that variables that resulted in nodes with higher purity have a higher decrease in Gini coefficient. The June surface temperature anomalies ("TSURF_anom_Jun" varibale in the figure above) were by far the most important variable in determining a yes-wildfire vs. no-wildfire event. Other relatively important climate variables were surface temperature anomaly of summer (July, October) and winter (February). Also, July surface mositure content may be another important variable. However, it seems overall surface temperature values have been very important drivers for tundra wildfire occurences in 2001-2015.