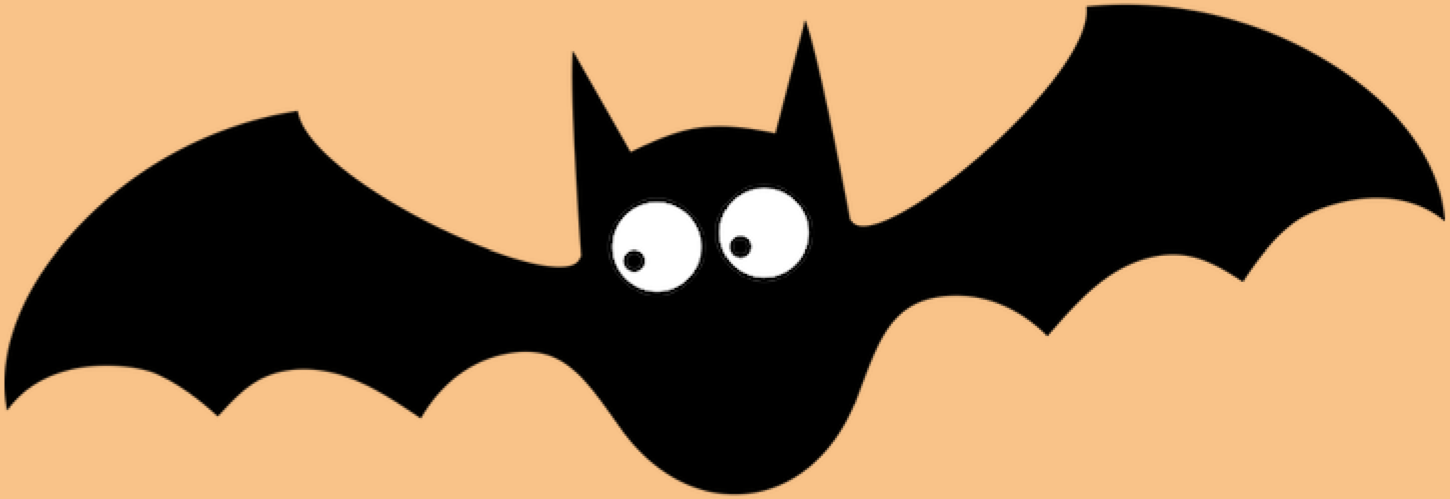


Hyperparameters Tuning - Random Forest

Son Nguyen

September 28, 2020

So far for pred. modeling.

① $\boxed{\text{TRAIN}}$
- ① Decision Tree "rpart"
- ② Random Forest "randomForest"

② $\boxed{\text{TRAIN}}$ + $\boxed{\text{TUNE}}$
- → Decision Tree
- → Random Forest

using caret

③ Model comparison

Data

What other hyperparameters I can tune?

- Go to https://topepo.github.io/caret/available-models.html
- Then search for the name of the method.
- OR

```
getModelInfo('rpart2')$rpart$parameters
```

```
##    parameter   class          label
## 1  maxdepth numeric Max Tree Depth
```

rf rf

Random Forest

- What hyperparameters can we tune?

```
getModelInfo('rf')$rf$parameters
```

```
##   parameter   class                         label
## 1      mtry numeric #Randomly Selected Predictors
```

- As shown, `mtry` is the only tuning parameter that method `rf` offers.

Random Forest

- Tuning `mtry` using Cross-Validation

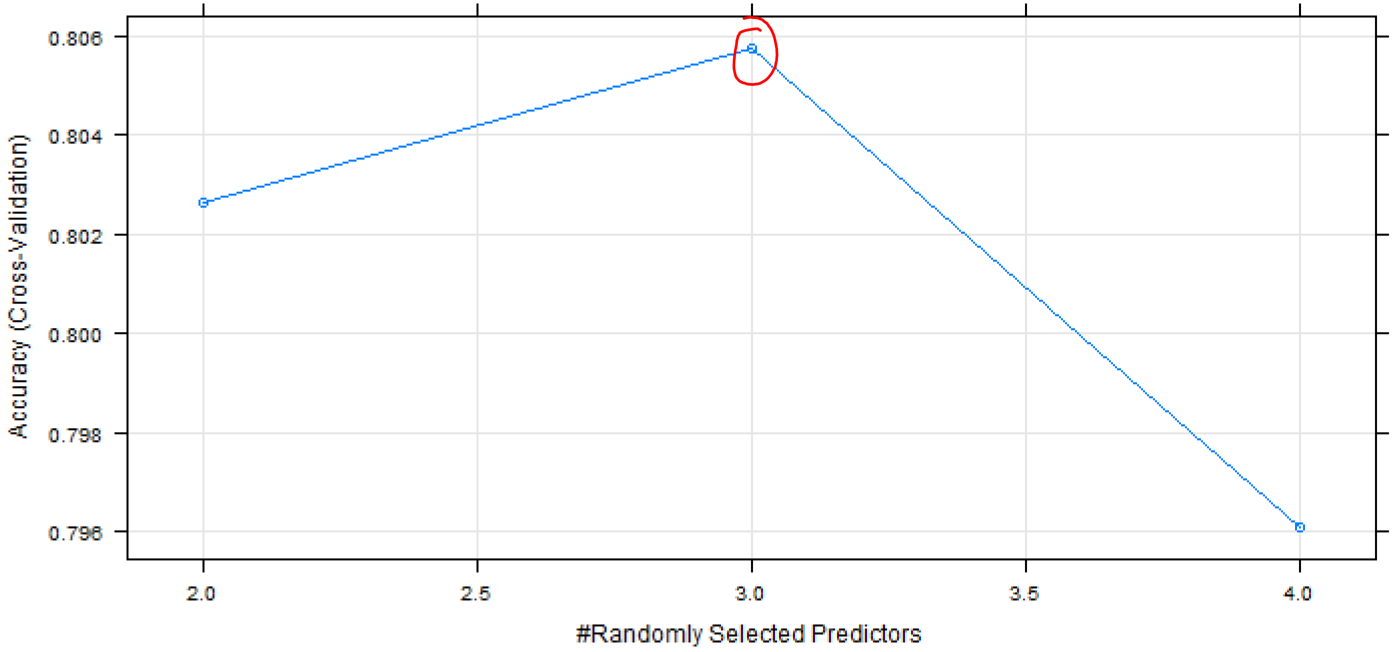
```
# Decide the range of the maxdepth to search for the best
tuneGrid = expand.grid(mtry = 2:4)        2, 3, 4

# Tell caret to do approach 3, i.e. Cross-Validation
trControl = trainControl(method = "cv",
                         number = 10)


# Do approach 3
forest_cv <- train(target~., data=df_train,
                                method = "rf",
                                trControl = trControl,
                                tuneGrid = tuneGrid)
```

Random Forest

```
plot(forest_cv)
```



Random Forest with Ranger

- If you want to tune other hyperparameter, you may want to look at different method for random forest.
- Random Forest can be implemented by method `ranger`
- Let's check to see what parameters we can tune with `ranger` method

```
getModelInfo('ranger')$ranger$parameters
```

```
##       parameter     class                         label
## 1          mtry   numeric #Randomly Selected Predictors
## 2     splitrule character                Splitting Rule
## 3 min.node.size   numeric             Minimal Node Size
```

- As shown, `ranger` offer to tune three hyperparameters for random forest: `mtry`, `splitrule` and `min.node.size`.

Tune

- Tune `mtry`, `splitrule` and `min.node.size`.

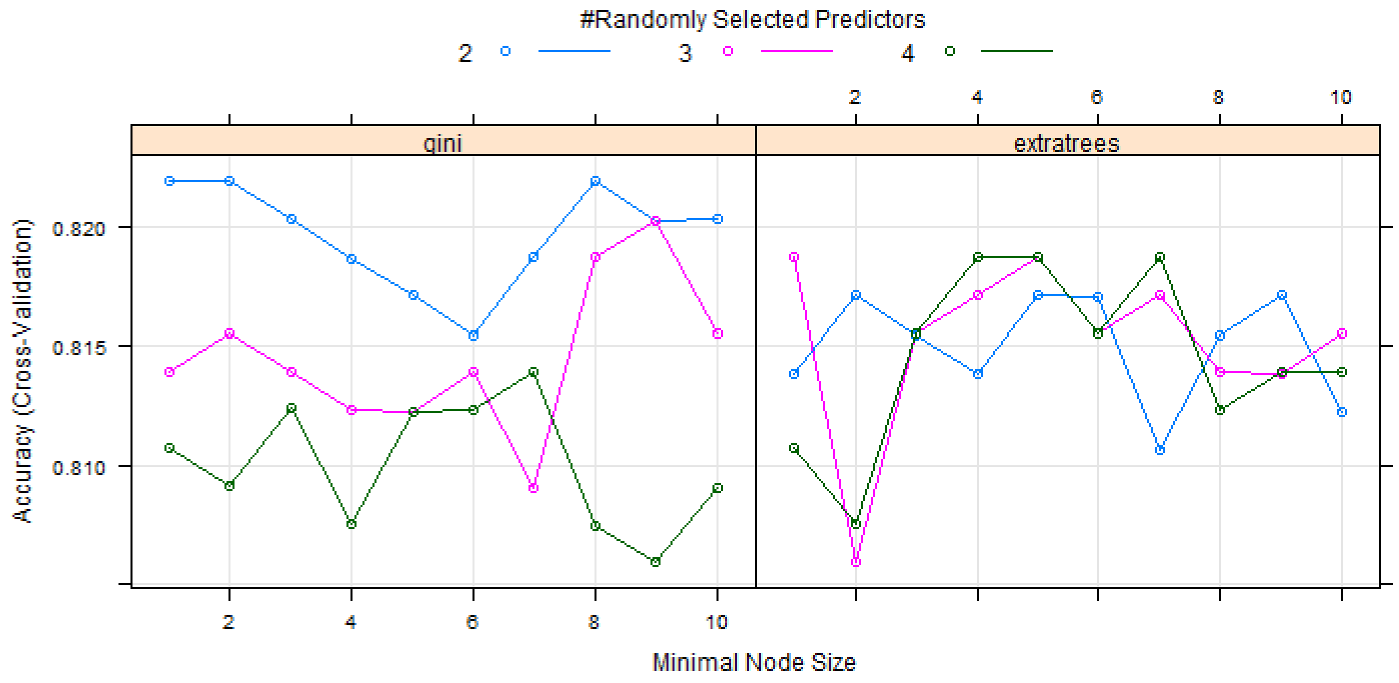
```
trControl = trainControl(method = "cv",
                         number = 10)

tuneGrid = expand.grid(mtry = 2:4,
                       splitrule = c('gini', 'extratrees'),
                       min.node.size = c(1:10))

# Do approach 3
ranger_cv ← train(target~., data=df_train,
                    method = "ranger",
                    trControl = trControl,
                    tuneGrid = tuneGrid)
```

Tune

```
plot(ranger_cv)
```

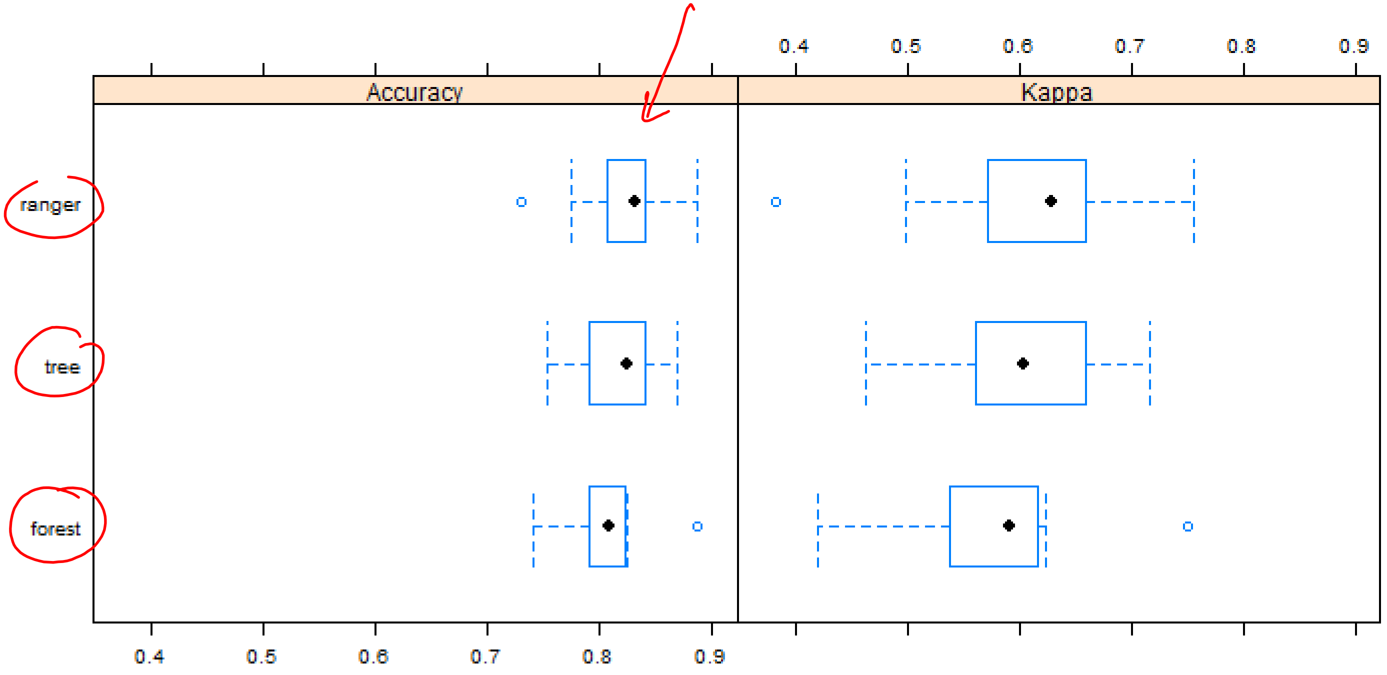


Model Comparison

- So far we have three models:

```
# Compare models
results ← resamples(list(forest = forest_cv,
                         ranger = ranger_cv,
                         tree= tree_approach3))
bwplot(results)
```

best



Final Model

- The comparison shows that random forest using `ranger` package is the best.

- Let's evaluate this model on the test data

```
pred <- predict(ranger_cv, df_test)

cm <- confusionMatrix(data = pred, reference = df_test$target, positive = "1")

cm$overall[1]
```

```
##  Accuracy
## 0.8458647
```