



HYPERPARAMETER TUNING IN R

Hyperparameter tuning in caret

Dr. Shirin Glander
Data Scientist

Voter dataset from US 2016 election

- Split into training and test set

```
library(tidyverse)
glimpse(voters_train_data)

Observations: 6,692
Variables: 42
$ turnout16_2016      <chr> "Did not vote", "Did not vote", "Did not vote", "Di
$ RIGGED_SYSTEM_1_2016 <int> 2, 2, 3, 2, 2, 3, 3, 1, 2, 3, 4, 4, 4, 3, 1, 2, 2,
$ RIGGED_SYSTEM_2_2016 <int> 3, 3, 2, 2, 3, 3, 2, 2, 1, 2, 4, 2, 3, 2, 3, 4, 3,
$ RIGGED_SYSTEM_3_2016 <int> 1, 1, 3, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 2, 1, 1, 1,
$ RIGGED_SYSTEM_4_2016 <int> 2, 1, 2, 2, 2, 2, 2, 2, 1, 3, 3, 1, 3, 3, 1, 3, 3,
$ RIGGED_SYSTEM_5_2016 <int> 1, 2, 2, 2, 2, 3, 1, 1, 2, 3, 2, 2, 1, 3, 1, 1, 2,
$ RIGGED_SYSTEM_6_2016 <int> 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 1, 3, 1, 3, 1, 1, 1,
$ track_2016          <int> 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 1, 2, 1, 1, 2, 2,
$ persfinretro_2016    <int> 2, 2, 2, 2, 1, 2, 2, 2, 3, 2, 3, 2, 2, 2, 2, 3, 3,
$ econtrend_2016       <int> 2, 2, 2, 3, 1, 2, 2, 2, 3, 2, 4, 1, 1, 2, 2, 2, 3,
$ Americatrend_2016    <int> 2, 3, 1, 1, 3, 3, 2, 2, 1, 2, 3, 1, 1, 2, 3, 3, 3,
$ futuretrend_2016     <int> 3, 3, 3, 4, 4, 3, 2, 2, 3, 2, 4, 1, 1, 3, 3, 3, 3,
$ wealth_2016          <int> 2, 2, 1, 2, 2, 8, 2, 8, 8, 2, 2, 2, 2, 2, 1, 2, 2,
...
```

Let's train another model with caret

- Stochastic Gradient Boosting

```
library(caret)
library(tictoc)

fitControl <- trainControl(method = "repeatedcv",
                           number = 3,
                           repeats = 5)

tic()
set.seed(42)
gbm_model_voters <- train(turnout16_2016 ~ .,
                          data = voters_train_data,
                          method = "gbm",
                          trControl = fitControl,
                          verbose = FALSE)

toc()

32.934 sec elapsed
```

Let's train another model with caret

```
gbm_model_voters
```

```
Stochastic Gradient Boosting
```

```
...
```

```
Resampling results across tuning parameters:
```

interaction.depth	n.trees	Accuracy	Kappa
1	50	0.9604603	-0.0001774346
...			

```
Tuning parameter 'shrinkage' was held constant at a value of 0.1
```

```
Tuning parameter 'n.minobsinnode' was held constant at a value of 10
```

```
Accuracy was used to select the optimal model using the largest value.
```

```
The final values used for the model were n.trees = 50, interaction.depth = 1, sh
```

Cartesian grid search with caret

- Define a **Cartesian grid** of hyperparameters:

```
man_grid <- expand.grid(n.trees = c(100, 200, 250),
                      interaction.depth = c(1, 4, 6),
                      shrinkage = 0.1,
                      n.minobsinnode = 10)

fitControl <- trainControl(method = "repeatedcv",
                          number = 3,
                          repeats = 5)

tic()
set.seed(42)
gbm_model_voters_grid <- train(turnout16_2016 ~ .,
                              data = voters_train_data,
                              method = "gbm",
                              trControl = fitControl,
                              verbose = FALSE,
                              tuneGrid = man_grid)

toc()

85.745 sec elapsed
```

Cartesian grid search with caret

```
gbm_model_voters_grid
```

```
Stochastic Gradient Boosting
```

```
...
```

```
Resampling results across tuning parameters:
```

interaction.depth	n.trees	Accuracy	Kappa
1	100	0.9603108	0.000912769
...			

```
Tuning parameter 'shrinkage' was held constant at a value of 0.1
```

```
Tuning parameter 'n.minobsinnode' was held constant at a value of 10
```

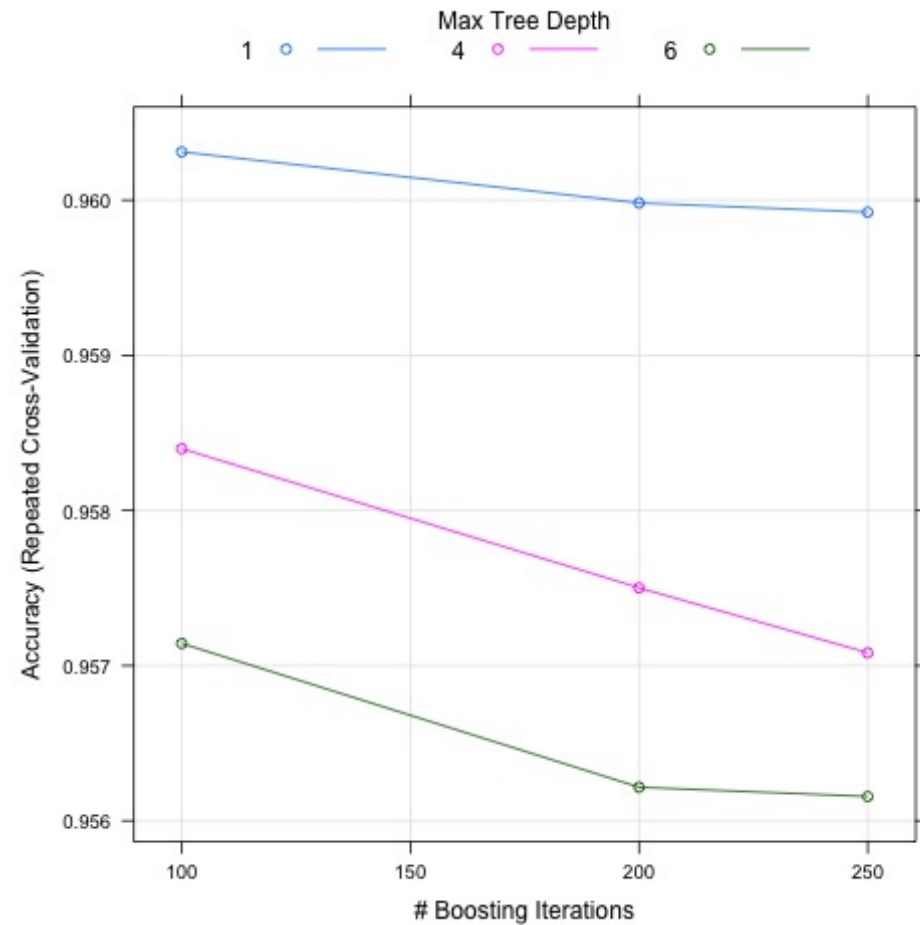
```
Accuracy was used to select the optimal model using the largest value.
```

```
The final values used for the model were n.trees = 100,
```

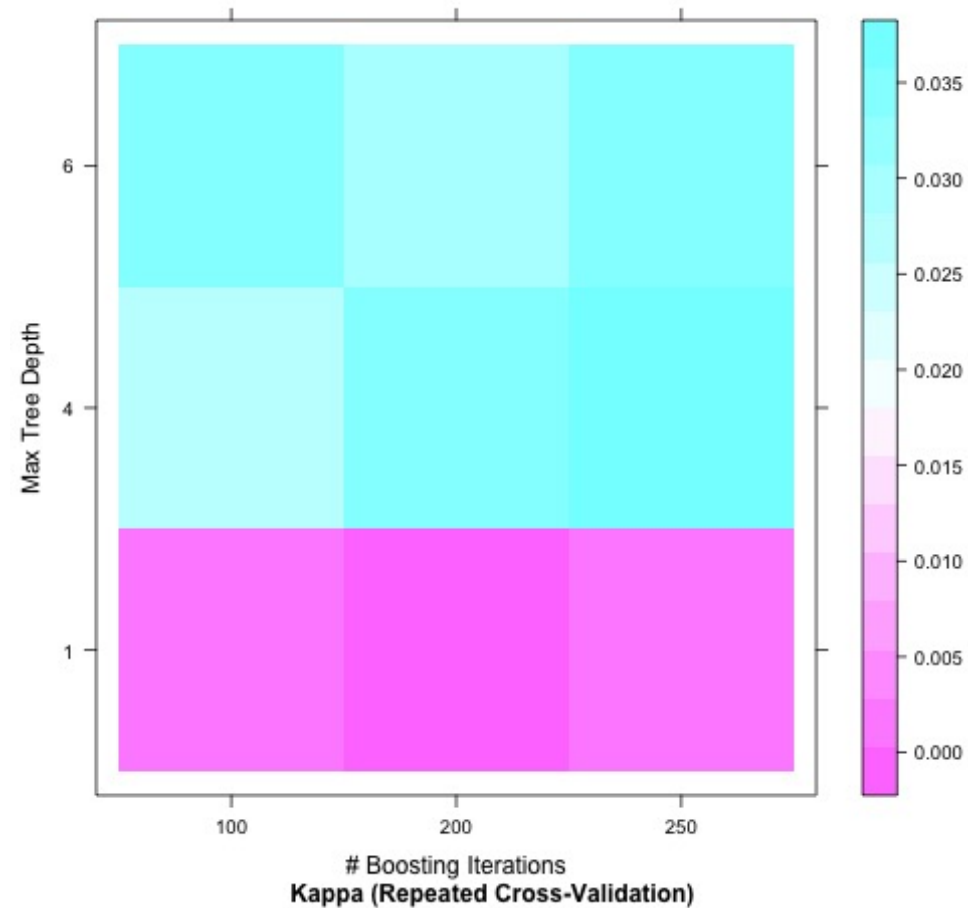
```
interaction.depth = 1, shrinkage = 0.1 and n.minobsinnode = 10.
```

Plot hyperparameter models

```
plot(gbm_model_voters_grid)
```



```
plot(gbm_model_voters_grid,  
     metric = "Kappa",  
     plotType = "level")
```





HYPERPARAMETER TUNING IN R

Test it out for yourself!



HYPERPARAMETER TUNING IN R

Hyperparameter tuning with Grid vs. Random Search

Dr. Shirin Glander
Data Scientist

Grid search continued

```
man_grid <- expand.grid(n.trees = c(100, 200, 250),
                       interaction.depth = c(1, 4, 6),
                       shrinkage = 0.1,
                       n.minobsinnode = 10)

fitControl <- trainControl(method = "repeatedcv",
                           number = 3,
                           repeats = 5,
                           search = "grid")

tic()
set.seed(42)
gbm_model_voters_grid <- train(turnout16_2016 ~ .,
                              data = voters_train_data,
                              method = "gbm",
                              trControl = fitControl,
                              verbose= FALSE,
                              tuneGrid = man_grid)

toc()

85.745 sec elapsed
```

Grid Search with hyperparameter ranges

```
big_grid <- expand.grid(n.trees = seq(from = 10, to = 300, by = 50),  
                      interaction.depth = seq(from = 1, to = 10,  
                                             length.out = 6),  
                      shrinkage = 0.1,  
                      n.minobsinnode = 10)
```

```
big_grid  
  n.trees interaction.depth shrinkage n.minobsinnode  
1      10             1.0      0.1          10  
2      60             1.0      0.1          10  
3     110             1.0      0.1          10  
4     160             1.0      0.1          10  
5     210             1.0      0.1          10  
6     260             1.0      0.1          10  
7      10             2.8      0.1          10  
8      60             2.8      0.1          10  
9     110             2.8      0.1          10  
10     160             2.8      0.1          10  
11     210             2.8      0.1          10  
12     260             2.8      0.1          10  
13      10             4.6      0.1          10  
...  
36     260            10.0      0.1          10
```

Grid Search with many hyperparameter options

```
big_grid <- expand.grid(n.trees = seq(from = 10, to = 300, by = 50),
                      interaction.depth = seq(from = 1, to = 10,
                                              length.out = 6),
                      shrinkage = 0.1,
                      n.minobsinnode = 10)

fitControl <- trainControl(method = "repeatedcv",
                          number = 3,
                          repeats = 5,
                          search = "grid")

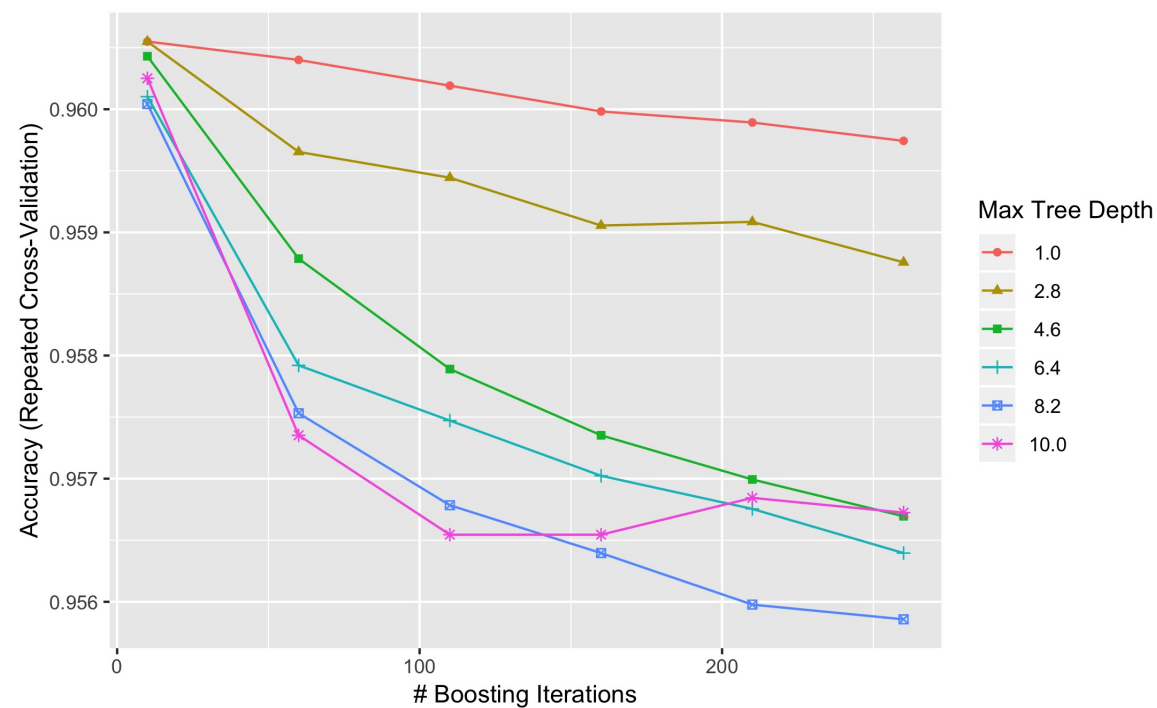
tic()
set.seed(42)
gbm_model_voters_big_grid <- train(turnout16_2016 ~ .,
                                  data = voters_train_data,
                                  method = "gbm",
                                  trControl = fitControl,
                                  verbose = FALSE,
                                  tuneGrid = big_grid)

toc()

240.698 sec elapsed
```

Cartesian grid vs random search

```
ggplot(gbm_model_voters_big_grid)
```



- Grid search can get **slow** and **computationally expensive** very quickly!
- Therefore, in reality, we often use **random search**.



Random Search in caret

- Define *random* search in `trainControl` function

```
library(caret)
fitControl <- trainControl(method = "repeatedcv",
                           number = 3,
                           repeats = 5,
                           search = "random")
```

- Set *tuneLength*

```
tic()
set.seed(42)
gbm_model_voters_random <- train(turnout16_2016 ~ .,
                                data = voters_train_data,
                                method = "gbm",
                                trControl = fitControl,
                                verbose = FALSE,
                                tuneLength = 5)

toc()

46.432 sec elapsed
```

Random Search in caret

```
gbm_model_voters_random
```

```
Stochastic Gradient Boosting
```

```
...
```

```
Resampling results across tuning parameters:
```

shrinkage	interaction.depth	n.minobsinnode	n.trees	Accuracy	Kappa
0.08841129	4	6	4396	0.9670737	-0.00853312
0.09255042	2	7	540	0.9630635	-0.01329168
0.14484962	3	21	3154	0.9570179	-0.01397025
0.34935098	10	10	2566	0.9610734	-0.01572681
0.43341085	1	13	2094	0.9460727	-0.02479105

```
Accuracy was used to select the optimal model using the largest value.
```

```
The final values used for the model were n.trees = 4396,  
interaction.depth = 4, shrinkage = 0.08841129 and n.minobsinnode = 6.
```

- **Beware:** in `caret` random search can **NOT** be combined with grid search!



HYPERPARAMETER TUNING IN R

Let's get coding!



HYPERPARAMETER TUNING IN R

Adaptive Resampling

Dr. Shirin Glander
Data Scientist

What is Adaptive Resampling?

Grid Search

- **All** hyperparameter combinations are computed.

Random Search

- Random **subsets** of hyperparameter combinations are computed.

=> Evaluation of best combination is done **at the end**.

Adaptive Resampling

- Hyperparameter combinations are **resampled** with values near combinations that performed well.
- Adaptive Resampling is, therefore, **faster and more efficient!**

"Futility Analysis in the Cross-Validation of Machine Learning Models." Max Kuhn; ARXIV 2014

Adaptive Resampling in caret

```
trainControl(method = "adaptive_cv" + search = "random" + adaptive =
```

- *min*: minimum number of resamples per hyperparameter
- *alpha*: confidence level for removing hyperparameters
- *method*: "gls" for linear model or "BT" for Bradley-Terry
- *complete*: if TRUE generates full resampling set

```
fitControl <- trainControl(method = "adaptive_cv",  
                           adaptive = list(min = 2, alpha = 0.05,  
                                           method = "gls", complete = TRUE),  
                           search = "random")
```

Adaptive Resampling in caret

- `trainControl()` + `tuneLength = x`

```
fitControl <- trainControl(method = "adaptive_cv",
                           number = 3,
                           repeats = 3,
                           adaptive = list(min = 2,
                                             alpha = 0.05,
                                             method = "gls",
                                             complete = TRUE),
                           search = "random")

tic()
set.seed(42)
gbm_model_voters_adaptive <- train(turnout16_2016 ~ .,
                                   data = voters_train_data,
                                   method = "gbm",
                                   trControl = fitControl,
                                   verbose = FALSE,
                                   tuneLength = 7)

toc()
1239.837 sec elapsed
```

Adaptive Resampling

```
gbm_model_voters_adaptive
```

```
...
```

Resampling results across tuning parameters:

shrinkage	interaction.depth	n.minobsinnode	n.trees	Accuracy	Kappa
0.07137493	5	6	4152	0.9564654	0.02856571
0.08408739	5	14	674	0.9547185	0.02098853
0.28552325	8	15	3209	0.9568141	0.03024238
0.33663932	10	13	2595	0.9571130	0.04250979
0.54251480	3	24	3683	0.9482171	0.03568586
0.56406870	7	25	4685	0.9549898	0.05284333
0.58695763	8	24	1431	0.9520286	0.02742592

Accuracy was used to select the optimal model using the largest value.

The final values used for the model were `n.trees = 2595`,
`interaction.depth = 10`, `shrinkage = 0.3366393` and `n.minobsinnode = 13`.



HYPERPARAMETER TUNING IN R

Let's get coding!