## 04/25 (Benchmark National Model I)

Finish the first benchmark model with a single random forest model. It needs at least 50GB memory and takes about 6 hours to fit (150 Trees, 12 Mtry, Extratree split rule). Overall, the model looks good, the relative importance is shown below:

Table

Description automatically generated with medium confidence

The most important detector-independent predictors are RI, temperature, and soil temperature. It seems that the age of surficial materials and the thickness of surficial material matters. The rank of importance generally makes sense.

Table

Description automatically generated with low confidence

Regarding the performance, the overall R2 looks promising. The average of 9-11 is around 0.90 with a RMSE of 0.31. Fair enough. The next step is to see whether increasing the number of trees and change mtry can really determine the performance of the model.

## 04/25 (Round I of parameter tuning)

When the ntree is 100 and mtry is 12, the performance is:

Text

Description automatically generated

A little worse than the benchmark model (n=150, mtry=12). The rank of relative importance also changes a bit to:

Graphical user interface, text

Description automatically generated

When the ntree is 100 and mtry is 8, the performance is:

Text

Description automatically generated Text

Description automatically generated

Significantly worse than the (n=100, mtry=12) model.

When the ntree is 100 and mtry is 10, the performance is:

Text

Description automatically generated

Significantly worse than the (n=100, mtry=12) model.

When the mtry increases to 14 (n=100,mtry=14), the performance is:

Text

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Almost identical to the performance of (n=100,mtry=12), suggesting increasing mtry has minimal marginal benefits.

When the mtry increases to 16 (ntree=100, mtry=16), the performance is:

Text, letter

Description automatically generated Text, letter

Description automatically generated

Also almost identical to the performance of (n=100, mtry=16), supporting that increasing mtry has minimal marginal benefits if kept ntree unchanged. But we do find that model with larger mtry tends to lower the importance of these proxy variables such as month, x, y. This makes sense because these proxy variables always conglomerated information from several variables, such as meteorological factors. By enlarge mtry, more actually important predictors are involved in splitting the nodes, making them more important. As a result, the mtry should be set at a reasonable value. Also, increasing mtry “only” elongate the fitting time a bit, not to a similar extent of ntree.

But it’s also worth noting that increasing the number of trees obviously elongate the model training.

When ntree is 200 and mtry is 10, the performance is

Text

Description automatically generated with low confidence

Better than the (n=100, mtry=10) model, suggesting that increasing the ntree helps.

Here’s the performance of the (ntree=200,mtry=12) model:

Text

Description automatically generated Text

Description automatically generated

A bit better than the benchmark model, very promising.

Here’s the performance of the (ntree=200, mtry=14) model:

Text

Description automatically generated with medium confidence Graphical user interface, text

Description automatically generated

A little better than the (ntree=200, mtry=12) model. Overall, these two models are quite similar.

Here’ the performance of the (mtree=200, mtry=16) model:

Text

Description automatically generated Graphical user interface, text

Description automatically generated with medium confidence

A little better again. And the importance of the proxy predictors falls further.

Here’s the performance of the (mtree=300, mtry=10) model:

Text

Description automatically generated Graphical user interface, text

Description automatically generated with medium confidence

Suggesting that increasing number of trees alone only marginally increase the model.

Here’s the performance of the (ntree=300, mtry=14) model:

Text, letter

Description automatically generated Graphical user interface, text

Description automatically generated

Here’s the performance of the (ntree=300, mtry=16) model:

Text

Description automatically generated Text

Description automatically generated

The performance itself looks promising as it’s significantly better than the (ntree=100, mtry=8) model. It seems that the current model, which takes ~20 hours to fit, hasn’t reached the best theoretical performance. Increasing both parameters seem to work a bit, but the margin is getting thinner.

## 04/26 (Round II of parameter tuning)

Here’s the performance of the (ntree=400, mtry=18) model:

Text, letter

Description automatically generated Graphical user interface, text

Description automatically generated

Further increasing the ntree and mtry seems not as helpful as before.

Here’s the performance of the (ntree=400, mtry=20) model:

Text, letter

Description automatically generated Graphical user interface, text

Description automatically generated

Further increasing mtry over 16 seems harm the overall performance a bit. The trend is conformed to seeing the performance of the (ntree=400, mtry=22) model:

Text

Description automatically generated

and the performance of the (mtree=400, mtry=24) model:

Text

Description automatically generated

## 04/30 Round II of parameter tuning

Here’s the performance of the (mtree=600, mtry=18) model:

Text, letter

Description automatically generatedGraphical user interface, text

Description automatically generated

Seems to be the best model up to now. Further increasing mtry also harm the performance of the model. The performance of the (mtree=600, mtry=20) model:

Text

Description automatically generated Graphical user interface, text

Description automatically generated

Not as good as the best one, but the difference is trivial.

## 05/01 Add Bedrock-related variables

After including another 65 bedrock-related variables in the model, fitting is becoming increasingly memory-intensive. The peak memory usage of a (mtree=100, mtry=8) model is ~ 50 GB. The parameter of the best model is subject to change.

After adding the bedrock-related covariates, the performance of the (mtree=100, mtry=8) model actually decreased significantly to:

Text

Description automatically generated Graphical user interface, text

Description automatically generated

Increasing the mtry to 12, the performance of (mtree=100, mtry=12) is

Text

Description automatically generated

Much better than the mtry=8 model.

Increasing the mtry to 16, the performance of (mtree=100, mtry=16) looks similart to the previous model without adding the bedrock-related covariates:

Text

Description automatically generated Graphical user interface, text

Description automatically generated

Increasing the mtry to 20, the performance of (ntree=100, mtry=20) look better

Text

Description automatically generated Graphical user interface, text

Description automatically generated with medium confidence

Hope this pattern can continue as we gradually increase the mtry to 24. It seems that shale bedrock play an important role in the spatial variation of radon. However, the performance of (ntree=100, mtry=24) model is

Text

Description automatically generated Text

Description automatically generated

Only slightly better than the previous model, suggesting that adding mtry doesn’t contribute a lot if the ntree is not increased accordingly.

When the ntree increases to 500, the performance of (ntree=500, mtry=12) is

Text

Description automatically generated

Better than the (ntree=100, mtry=12) model, suggesting that both ntree and mtry should be increased. By the way, all results reported originally by caret/ranger are out-of-bag (OOB) performance. If I use the predict() function to make prediction on the training data, the R2 is generally much greater than the OOB R2.