TREE BASED MODELS

Random Forest

INTRODUCTION

- *Random Forest* can be used for
  - Classification
  - Regression
- A *Random Forest* model is an ensamble model consisting of many (hundreds or thousands) of slightly different *Decision Trees*.
- The majority predictions (majority vote) from all *Decision Trees* becomes the prediction of the *Random Forest* for classification»
- The average predictions from all *Decision Trees* becomes the prediction of the *Random Forest* for regression»

THE IDEA BEHIND RANDOM FOREST

The idea that a combination of weak predictors can lead to a strong prediction is analogous to the wisdom of crowds phenomenon described in Galton 1907:

Visitors at a stock and poultry exhibition in England submitted guesses about the weight of an ox. Although most of the visitors were off with their predictions, surprisingly the mean of all predictions was very close to the real weight of the ox.»

HOW TO ENSURE THAT DECISION TREES ARE (SLIGHTLY) DIFFERENT

- **Random Subspace Method:** Each decision tree uses only a random selection of the predictors.
  - As a rule of thumb: If we have m predictors, each decision tree uses only $\sqrt{(}M)$ predictors (rounded down if needed).
  - In the case here, where we have 7 predictors, each decision tree uses only 2 randomly chosen predictors ($\sqrt{(}7) = 2.65$ rounded down to 2)

- **Bagging:** Each decision tree is presented with slightly different training data: We draw from the original training data with replacement to step-wise generate a new training dataset (**Bootstrapping**).
  - The new training dataset has the same size as the original dataset.
  - Generate a bootstrap dataset for every decision tree.

EXAMPLE FOR BAGGING WITH BOOTSTAPPING

Creating 5 Bootstrap Training Datasets `DataTrain1`, `DataTrain2`, `DataTrain3`, `DataTrain4`, `DataTrain5` from original dataset `DataTrain`.

Original Training Data:

▶ Code

ID	name	Sex	Age	Weight
1	Adam	male	22	145
2	Berta	female	54	180
3	Carlos	male	32	167
4	Dora	female	21	197
5	Ernst	male	32	221
6	Fiola	female	87	146
7	Gert	male	18	230

1st Bootstrap Sample:

▶ Code

ID	name	Sex	Age	Weight
1	Adam	male	22	145
2	Berta	female	54	180
4	Dora	female	21	197
6	Fiola	female	87	146
6	Fiola	female	87	146
3	Carlos	male	32	167
4	Dora	female	21	197

2nd Bootstrap sample:

▶ Code

ID	name	Sex	Age	Weight
2	Berta	female	54	180
4	Dora	female	21	197
1	Adam	male	22	145
1	Adam	male	22	145
4	Dora	female	21	197
4	Dora	female	21	197

ID	name	Sex	Age	Weight
2	Berta	female	54	180

3rd Bootstrap sample:

▶ Code

ID	name	Sex	Age	Weight
7	Gert	male	18	230
6	Fiola	female	87	146
5	Ernst	male	32	221
4	Dora	female	21	197
7	Gert	male	18	230
7	Gert	male	18	230
7	Gert	male	18	230

4th Bootstrap sample:

▶ Code

ID	name	Sex	Age	Weight
6	Fiola	female	87	146

ID	name	Sex	Age	Weight
2	Berta	female	54	180
6	Fiola	female	87	146
2	Berta	female	54	180
7	Gert	male	18	230
5	Ernst	male	32	221
6	Fiola	female	87	146

5th Bootstrap sample:

- Code

ID	name	Sex	Age	Weight
2	Berta	female	54	180
6	Fiola	female	87	146
1	Adam	male	22	145
4	Dora	female	21	197
1	Adam	male	22	145
7	Gert	male	18	230
2	Berta	female	54	180

REAL WORLD DATA WITH A DECISION TREE

Predicting vaccination rates in the U.S. based on data from September 2021:

- Outcome variable: Percentage of fully vaccinated (two shots) people ($PercVacFull$).
- Data from 2,630 continental U.S. counties.»

REAL WORLD DATA WITH A DECISION TREE – PREDICTOR VARIABLES

- Race/Ethnicity:
  - Counties' proportion African Americans ($PercBlack$),
  - Counties' proportion Asian Americans ($PercAsian$), and
  - Counties' proportion Hispanics ($PercHisp$)
- Political Affiliation (Presidential election 2020):
  - Counties' proportion Republican votes ($PercRep$)
- Age Groups in Counties:
  - Counties' proportion young adults (20-25 years); $PercYoung25$
  - Counties' proportion older adults (65 years and older); $PercOld65$
- Income related:
  - Proportion of households receiving food stamps ($PercFoodSt$)»

LOADING THE DATA AND ASSIGNING TRAINING AND TESTING DATA

▶ Code

County	State	PercVacFull	PercRep	PercAsian	PercBlack	PercHisp	PercYoung2
Baldwin	AL	0.504	0.7506689	0.0092	0.0917	0.0456	0.063418
Barbour	AL	0.416	0.4911710	0.0048	0.4744	0.0436	0.073464
Chambers	AL	0.315	0.7063935	0.0112	0.3956	0.0238	0.064653
Cherokee	AL	0.318	0.8319460	0.0025	0.0460	0.0159	0.052545
Choctaw	AL	0.648	0.7445596	0.0013	0.4255	0.0041	0.061241
Cleburne	AL	0.319	0.8402859	0.0002	0.0275	0.0246	0.059486

CREATING MODEL DESIGN, RECIPE, AND FITTED WORKFLOW (DEFAULT HYPER-PARAMETERS)

- Code

```
Default value for mtry is: 2
```

```
══ Workflow [trained] ══════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: rand_forest()

── Preprocessor ────────────────────────────────────────────────────────────────
0 Recipe Steps

── Model ───────────────────────────────────────────────────────────────────────
Ranger result

Call:
 ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~DefaultMtry,      x), num.trees =
~500, min.node.size = min_rows(~5, x), num.threads = 1,      verbose = FALSE, seed =
sample.int(10^5, 1))

Type:                             Regression
```

METRICS FOR DEFAULT RANDOM FOREST MODEL

- Code

```
# A tibble: 3 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard      0.109
2 rsq     standard      0.450
3 mae     standard      0.0771
```

TUNING THE RANDOM FOREST MODEL

On the next slide you find a link to an R script that you can use to tune the Random Forest model.

Try to beat the predictive quality of the *default* model we used here.

TUNING THE RANDOM FOREST MODEL

```
ModelDesignRandForTree=rand_forest(trees=500, min_n=tune(), mtry = tune()) %>%
                       set_engine("ranger") %>%
                       set_mode("regression")
```

What to consider for the tuning grid?

- `trees=500`: No need to tune because a high number of trees cannot lead to overlearning. This is because the Random Forest prediction is calculated from the average of the trees. However, a high number of trees slows the training (`trees=500' was set; feel free to change it).
- `mtry=tune()` is a hyper-parameter that should be tuned. It determines the number of predictors randomly considered for each decision tree. Recommended is $\sqrt{(7)} = 2.65$. Maybe you try 2, 3, 5.
- `min_n=tune()` is a hyper-parameter that should be tuned. It determines the minimum number of observations inside a node required for a split into child nodes. If this number is not reached the node becomes a terminal node. Therefore, `min_n=tune()` indirectly determines the tree depth. Default is 5. Maybe you try 3, 5, 10, 30.

Below is a link for an R script that you can use to tune the *Random Forest* model.